Using Mobile Phones to Nurture Social Networks

Using Mobile Phones to Nurture Social Networks

Novel technologies for nurturing a person's contacts use mobile phones to recommend new friends or track a friendship's health. Such monitoring can also help people monitor their own emotions.

Daniele Quercia
Massachusetts Institute of Technology

Jonathan Ellis and Licia Capra
University College London

Adolescent depression is a typical response to the normal maturation process, but just because it's common doesn't mean it won't adversely affect a young person's school performance or impair family relationships. One of the most effective ways of countering depression is strong social support from friends,[1] and with today's technology, it's easier than ever before to maintain contact with this support network. Adolescents no longer have to sit in front of their PCs to talk to friends—rather, they constantly communicate with each other via mobile social networking applications.

Some sociologists initially equated a permanent online presence with social isolation, believing that young people who spent all their time transfixed by mobile phone screens weren't making connections with the outside world. In the early 2000s, however, other sociologists started questioning this viewpoint, conducting small-scale studies and showing that people who go online regularly often have higher levels of face-to-face interactions as well.[2] A recent large-scale sociological study found that mobile phone owners and social networking website users are more likely to belong to a local voluntary group, such as a neighborhood association, sports league, youth group, church, or social club.[3]

To help adolescents effectively grow and nurture their social relations, we engineered a new technology for mobile phones that silently keeps track of users' collocations, as well as the frequency of voice calls and text messages. Two new algorithmic frameworks then process this data, with the following goals:

- *To effectively find social contacts based on encounters*. We present a framework that reveals friends' relationships based on collocation data. The framework runs either on a social-networking site or on the mobile phone itself. The latter choice is appealing to privacy-conscious users. Simulation studies conducted on real data demonstrate the ability of a set of algorithms to reveal friends' relationships.
- *To nurture online and offline contacts*. We also present a framework that detects patterns in both physical encounters (collocations) and social activity (phone calls and text messages), as well as deviations from such patterns. If a user suddenly becomes less sociable, the engine sends an alert through a numeric code or avatar. We're also studying the degree to which this inference engine can predict user mood simply based on activity.

We've implemented and deployed both frameworks on BlackBerry mobile phones with negligible computational and communication overhead, thus confirming how this technology can run silently on modern mobile phones.

Finding Social Contacts: FriendSensing

FriendSensing is a framework that lets new members of social networking websites discover their friends automatically.[4] It also helps

Published by the IEEE CS ■ 1536-1268/10/$26.00 © 2010 IEEE

Related Work in Social Relationship Discovery

Various approaches exist that aim to automatically discover social relations. They mainly differ in the information they process: social networking profiles, e-mails, or data from portable devices.

Social Networking Profiles

Jilin Chen and colleagues proposed four algorithms for suggesting people on Beehive (an IBM internal social networking website).[1] The algorithms are different combinations of two basic ideas. The first idea is to match people by common interests (for example, those who blog on similar topics or share the same role within IBM), and the second idea is to match people based on their social connections (for example, connecting friends of friends). However, the two ideas rely solely on profile content, and the researchers concede that this restriction should be dropped in future implementations.

E-mails

Thomas Karagiannis and Milan Vojnovic gathered the e-mails exchanged by their company's research labs' more than 100,000 employees.[2] They then represented their data as a graph whose nodes were employees and whose links were e-mail exchanges. Then, to recommend new e-mail addresses for contact lists, they connected friends-of-friends relationships.

Portable Device Data

On mobile phone data, Ankur Gupta and colleagues showed that it's possible to identify and recommend popular hangouts.[3] More recently, Danny Wyatt and colleagues built a framework with which collar devices could capture audio readings and automatically suggest to their users who they might know.[4] Using audio sensors, the collar devices record face-to-face conversations and, based on conversation length, infer who's likely to befriend whom. The inference is made possible by knowing global properties (the clustering coefficients) of the users' social networks. Under this assumption, the promise is that you could accurately reconstruct the whole social network.

Our Work

FriendSensing takes a different, more ubiquitous approach, whereby friends are recommended starting from readily available information (proximity data from mobile phones),[5] requiring no a priori knowledge about users and their social ties. In the Reality Mining project, Nathan Eagle and colleagues demonstrated that social ties are likely to exist between individuals who behave similarly.[6,7] In FriendSensing, we built on that work and analyzed how different prediction strategies of social ties would perform. SensingHappiness uses sentiment analysis techniques proposed by Peter Sheridan Dodds and Christopher M. Danforth.[8] To determine whether a small group of people is happy or sad, psychologists hand out questionnaires or conduct interviews. In their article, Dodds and Danforth argued that people are more honest in personal writings than during formal psychological tests. For this reason, they crawled 2.4 million blogs, scanned the texts for more than 1,000 emotionally charged words that a 1999 psychology study had ranked on a scale from 1 (miserable) to 9 (ecstatic), and calculated an average happiness score for each blog.

REFERENCES

1. J. Chen et al., "Make New Friends, but Keep the Old: Recommending People on Social Networking Sites," *Proc. Int'l Conf. Human Factors in Computing Systems* (CHI), ACM Press, 2009, pp. 201–210.
2. T. Karagiannis and M. Vojnovic, "Behavioral Profiles for Advanced Email Features," *Proc. 18th Int'l World Wide Web Conf.*, ACM Press, 2009, pp. 711–720.
3. A. Gupta et al., "Automatic Identification of Informal Social Groups and Places for Geo-Social Recommendations," *Int'l J. Mobile Network Design and Innovation*, vol. 2, no. 3/4, 2007, pp. 159–171.
4. D. Wyatt, T. Choudhury, and J. Bilmes, "Learning Hidden Curved Exponential Random Graph Models to Infer-Face-to-Face Interaction Networks from Situated Speech Data," *Proc. 23rd Int'l Conf. Artificial Intelligence* (AAAI), AAAI Press, 2008, pp. 732–738.
5. D. Quercia and L. Capra, "FriendSensing: Recommending Friends Using Mobile Phones," *Proc. Int'l Conf Recommender Systems* (RecSys), ACM Press, 2009, pp. 273–276.
6. N. Eagle, A.S. Pentland, and D. Lazer, "Inferring Friendship Network Structure by Using Mobile Phone Data," *Proc. Nat'l Academy of Sciences*, vol. 106, no. 36, 2009, pp. 15274–15278.
7. N. Eagle and A. Pentland, "Eigenbehaviors: Identifying Structure in Routine," *Behavioral Ecology and Sociobiology*, vol. 63, 2009, pp. 1057–1066.
8. P. Dodds and C. Danforth, "Measuring the Happiness of Large-Scale Written Expression: Songs, Blogs, and Presidents," *J. Happiness Studies*, July 2009, www.springerlink.com/content/757723154j4w726k.

existing members elicit new social relationships. Essentially, FriendSensing automatically creates personalized recommendations of the other people the user might know in two steps:

- *Logging encounters*. Using short-range radio technologies readily available on almost all modern mobile phones (such as Bluetooth), each device transparently records encounters with collocated people. More precisely, each phone *A* keeps track of how many times it has met phone *B* and how much time it's spent being collocated with

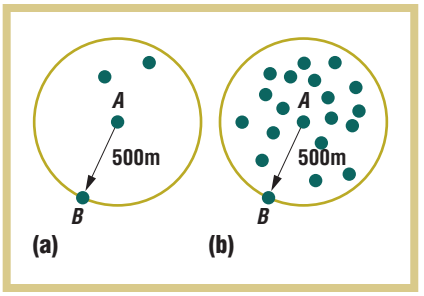


Figure 1. Friendship probability. If *A* and *B* live 500 meters apart, they could be (a) next-door neighbors in the countryside or (b) total strangers in central London.

it. Here, we make the assumption that a mobile phone is a personal device not shared among various people. Moreover, we assume it's possible to link devices (such as a phone's Bluetooth ID number) to users' identities on social networking websites (as pioneered by the Cityware project[5]).

- *Recommending friends*. Both the social networking site and the mobile device can process the collocation records to elicit relevant encounters and arrange them into a weighted social network; similarly, either the site or the device can then traverse this network to compute personalized lists of people each user might know. FriendSensing doesn't prescribe where the processing of proximity records or the navigation of the inferred social network should occur—it can be done on a social networking site or mobile device.

We now focus on the algorithms for proximity processing and network navigation in general terms; we defer a discussion about the privacy implications of different architectural deployments until later in this article.

Processing Encounters

Once FriendSensing collects the collocation logs, it must then filter out irrelevant encounters—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 geographic proximity, based on the intuition that friendship probability increases with geographic proximity—the closer two individuals are, the likelier they are to be friends. In other words, we can model 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 David Liben-Nowell and his colleagues later demonstrated, the absolute value of geographic distance alone is insufficient to model friendship.[6] To see how, imagine A and B live 500 meters apart: at the very same distance, A and B would likely be next-door neighbors in the countryside (Figure 1a), but complete strangers in central London (Figure 1b). This suggests that we must also consider population density. Liben-Nowell and colleagues did so in a simple way: they replaced the absolute distance $dist(A, B)$ with a ranked distance, $p(A \rightarrow B) \propto 1/(rankDist_A(B))$.[6]

The denominator is A's rank of B, which is the number of people who are closer to A than B is; it's expressed as (we add "+1" to avoid division by zero):

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

In other words, the probability of A befriending B depends on the number of people within distance $dist(A, B)$. The denser 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. Liben-Nowell and his colleagues successfully evaluated this model with a half-million profiles collected from the LiveJournal blogging website, suggesting that geography is a good friendship predictor. However, geographic information isn't widely available on mobile phones, and should localization technology such as GPS become a commodity, it would still fail to capture indoor encounters. We thus need to reformulate the problem based on "mobile phone proximity."

Using mobile phones, we can keep track of how many times A has met B (frequency $freq(A, B)$) and how long each encounter lasted (duration $dur(A, B)$). We can now 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 geographic information, we can't consider frequency or duration alone to compute friendship probabilities because they're both distributed nonuniformly. Indeed, individuals have skewed mobility patterns—this is true not only for college students[7] but also for conference attendees and hundreds of thousands of other mobile users.[8] Rather than using absolute frequency and duration values, we can instead take their rank. From frequency, the friendship probability becomes

$$p(A \rightarrow B) \propto \frac{1}{rankFreq_A(B)}, \quad (1)$$

where $rankFreq_A(B) = |\{C : freq(A, C) > freq(A, B)\}| + 1$. Consequently, the probability of A befriending B depends on the number of people who have met A more frequently than B has.

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

$$p(A \rightarrow B) \propto \frac{1}{rankDur_A(B)}, \quad (2)$$

where $rankDur_A(B) = |\{C : dur(A, C) > dur(A, B)\}| + 1$. Again, the probability of A befriending B depends not on $freq(A, B)$ or $dur(A, B)$ itself but on the number of people who have met A more

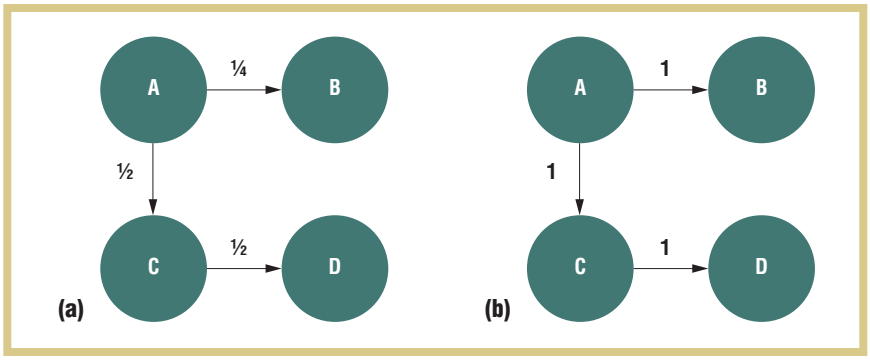


Figure 2. Encounter networks. The link weights are (a) friendship probabilities or (b) ranks.

frequently or for a longer duration than B has.

We can then compute friendship probabilities and ranking from proximity logs and build a weighted social networks of encounters: each mobile device would be represented as a node, and a link would be 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 (that is, A's ranking of B, which we can compute from the friendship probability itself). Let's now look more closely at when to opt for probabilities versus ranks.

Computing Recommendation Lists

Based on A's network of encounters, FriendSensing generates a personalized list of people A might know—that is, it predicts which of A's encounters are likely to be A's friends.

In social network literature, this problem is called *link prediction*, and researchers have proposed different methods to tackle it.[9] Most of these methods assign a score to a pair of nodes (A, B) following one of two possible strategies.

Markov chain algorithms. For the Markov chain class of algorithms, we compute the score between a pair of nodes A and B as the fraction of time spent at B by a random walk in the network originating at A. In this type of network, weights reflect the connection strength between node pairs. Therefore, we adopt friendship probabilities $p(A \rightarrow B)$ as link weights instead (see Figure 2a). The algorithms in this class then convert the network in a first-order Markov chain, the idea being that after starting at node A (the *prior node*), the walk can 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.
- *K-MarkovChain.* This algorithm unfolds as PageRank with prior does, expect for the walk which is now of fixed length K.
- *Hyperlink-Induced Topic Search (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.

Once FriendSensing computes the scores for a walk originating in A, it then uses them to build A's personalized recommendation list.

Shortest path. The score between a pair of nodes A and B is the weighted length of the shortest path between them. The intuition behind this approach is that social networks are "small worlds" (individuals connected by short chains), 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—here, weights reflecting how unlikely it is for two nodes to befriend each other. Because 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 (see Figure 2b). The path length is then weighted in the sense that it's the sum of the weights along the shortest path.

Evaluation

FriendSensing's goal is to recommend new social contacts. To ascertain FriendSensing's effectiveness at meeting this goal, we set up a simulation driven by real data collected as part of the Reality Mining project at the Massachusetts Institute of Technology. The MIT traces contain collocation information from 96 subjects (staff and students) at the MIT campus during the 2004–2005 academic year. We gave them Bluetooth-enabled Nokia 6600 phones and collected collocation information (roughly a 10-meter range) via frequent (five-minute) Bluetooth device discoveries. Note that the users in this data set are young adults rather than "youths": in particular, 30 users were incoming freshmen, 20 were incoming masters students, and the remaining were older students and staff. However, we expect the results obtained to hold equally for mobility scenarios of adolescents; in fact, as existing analysis demonstrates, such traces share many unifying features with other mobility traces.[8]

Besides providing mobility traces, the MIT data set also implicitly includes information about the users' social networks. In fact, it logs both the text messages sent and the phone calls made by each phone during the study. Using this information, we extracted a

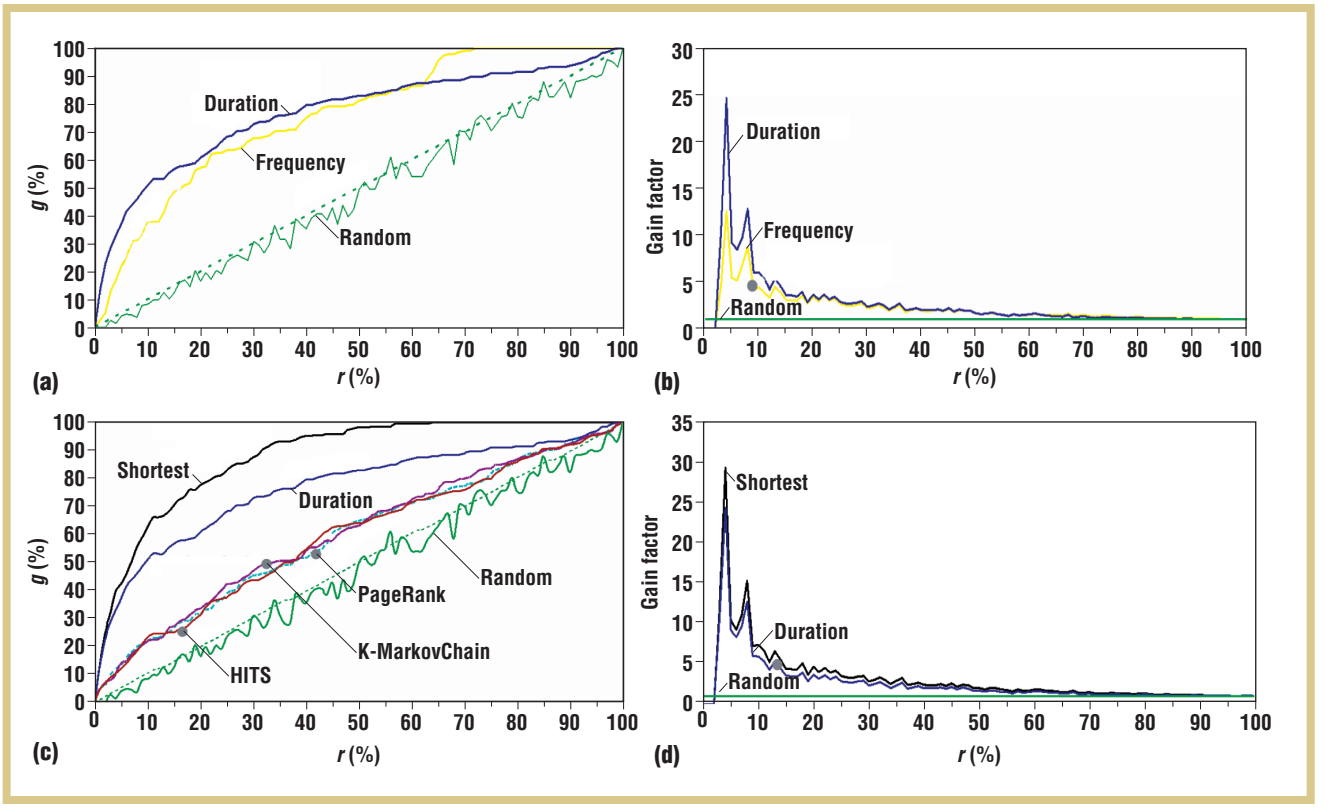


Figure 3. Evaluation results. (a) Predicted ties *g* versus recommended people *r* for two algorithms based on frequency and duration of encounters, (b) gain factor versus recommended people *r* for the two algorithms, (c) predicted ties *g* versus recommended people *r* for six algorithms, and (d) gain factor versus recommended people *r* for the six algorithms.

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 participants' actual social networks and computed the fraction of network ties that FriendSensing correctly predicted. We refer to this fraction as "good recommendations" *g* and study how *g* varies while we increase the percentage *r* of people recommended to each user from 0 to 100 percent.

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

Frequency vs. duration. In the first set of experiments, we aimed to compare the effectiveness of frequency as a collocation processing strategy, as opposed to duration. We did so by disabling any link propagation strategy and using the ranking produced in the frequency/duration collocation processing strategies locally. This is equivalent to running FriendSensing on someone's mobile device without reporting his or her proximity log to a social networking website (where we could execute the full FriendSensing approach, including link propagation). Figure 3a plots *g* (good recommendations) versus *r* (recommended people) for the two basic 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) because the more people recommended, the likelier it is that FriendSensing will get some of them right. At the extreme of $r = 100$ percent (all users recommended to each user), *g* reaches 100 percent (for all strategies). The two remaining strategies perform significantly better than random, but duration discovers friends faster than frequency. To see which strategy performed better over another, we compared frequency and duration against the random one. Essentially, we defined the gain factor over random as

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

where $g_{strategy}$ is the fraction of good recommendations for a given strategy (e.g., duration, frequency) and g_{random} is the fraction of good recommendations for the random strategy. A factor of one means the strategy performs no better than random (no gain), and a factor of two means that the strategy

performs two times better than random. Figure 3b shows that duration gains more than frequency, especially for the first 20 percent of people recommended. As expected, the effectiveness of frequency and duration dies off to a point where their gains flatten toward random. This is because after recommending the most friends possible, any strategy we use will only have a few friends left to recommend (and those will be hard to predict).

Duration and "link prediction." Our second set of experiments compared the four different link prediction strategies we presented earlier—shortest path, PageRank, HITS, and K-Markov-Chain. We did experiments on a social network of encounters built from duration and frequency information. Because the results we obtained with duration were consistently better than those obtained with frequency, we report results for the former case only. Figure 3c plots g versus r for all four strategies. It also plots 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 collocation information for propagation processing. PageRank, HITS, and K-Markov-Chain performed equally well and only showed small differences due to confidence in the results. In fact, the results were similar and came from the common use of Markov chains by all three algorithms. We found that we'd be better off using only duration information rather than combining it with the three algorithms. This isn't necessarily bad news because it suggests that by relying only on his or her own proximity information, a user gets quality recommendations and retains control of his or her own private data. In line with the literature, shortest path performs best. Indeed, Figure 3d shows that it gains more than duration, and it does so consistently. This is because, unlike duration, shortest path can suggest to user A those friends who belong to A's social circle but haven't yet met A.

Cultivating Social Contacts: SensingHappiness

Once users find social contacts with FriendSensing, they then need to cultivate them. The idea we're currently exploring is whether by logging not only encounters but also calls and text messages, mobile phones can help users cultivate their contacts.

Awareness of Friendship Health

As research suggests,[7] a misalignment exists between individuals' perception of how much time they spend with their friends and how much time they actually spend in their proximity. In particular, perception seems to grossly overestimate reality. If users suddenly become notably less sociable with one of their friends, an application running on their phones could make them aware that perhaps they're neglecting a friendship. An avatar or numeric code could represent a friendship's health, in line with current work on affective computing.[10] Studies confirm the importance of social interaction in the process of friendship formation and dissolution;[11] moreover, they highlight how people have a tendency to abandon asymmetric relationships ("I call you, but you never call me"). Of course, users might purposely neglect some relationships, so a fade-and-forget function could be applied to friends as a user ceases interacting with them (the function archives people with whom a user stops interacting). In so doing, users don't need to "unfriend" anyone; rather, ignoring undesirable relations for a specific time period would suffice, which also alleviates traumatic social rejections. Being unfriended often results, deliberately or accidentally, in upset feelings.[12]

User Mood and Phone Activity

In addition to helping us evaluate the FriendSensing framework, the Reality Mining project at MIT offered insight into mood prediction. In their analysis of the Reality Mining data, Nathan Eagle, Sandy Pentland, and David Lazer[7] compared the behavioral data from mobile phones (mobility, calls, and text messages) with self-report survey data from the participants themselves. The researchers found that they could predict job satisfaction, for example, solely from behavioral data. More specifically, they found that "having friends, especially ones to whom you were near at work, predicted satisfaction with the work group, and calling friends while at work was associated with lack of satisfaction with the work group."[7] The researchers concluded that visible behavioral data (activity with mobile phones) offers an insight into invisible cognitive constructs such as mood and satisfaction.

Using this research as a springboard, we've started a new project called SensingHappiness, in which we analyze data crawled from the microblogging service Twitter to test whether a visible activity level offers any insight into the inferred mood of its users. Twitter lets users post messages ("tweets") of up to 140 characters and supports a variety of communicative practices, including public republishing of something other users have written ("retweets") in the

The researchers concluded that visible behavioral data (activity with mobile phones) offers an insight into invisible cognitive constructs such as mood and satisfaction.

attempt to spread whatever word they feel like spreading.

From this data, we'll test whether deviation from "usual activity" can predict a Twitter user's mood. We'll first select the (re)tweets that express sentiments ("I feel ...," "I'm feeling ...") and classify each of these (re)tweets according to the mood they express (such as happiness or sadness) using existing sentiment analysis algorithms.[13] We'll then look at each user's (re)tweets before and after expressing the sentiment and study whether someone's activity level (in terms of number of tweets and retweets) deviates from his or her "normal" activity. By predicting mood, SensingHappiness can make users more aware of their emotions and, consequently, take action or pay particular attention to a resulting behavior.

The current implementation of the FriendSensing and SensingHappiness technology follows the thick-client model: all the logging and processing happen on the mobile phone.

Implementing FriendSensing and SensingHappiness on a BlackBerry

To bring our research out of the lab and into the real world, we implemented FriendSensing and SensingHappiness on the BlackBerry platform (specifically, the BlackBerry Pearl 8120). In selecting a target mobile platform, we had the following constraints: it had to allow applications to seamlessly run in the background; it had to have an API that's openly accessible through open source languages; and finally, the API had to support Bluetooth access. Although Google now offers a full Bluetooth API in Android 2.0, at the time of development, there was no support for it; the iPhone doesn't allow background applications to run, and any Windows Mobile application would require development to be done in Microsoft's proprietary .NET languages. Consequently, we went with BlackBerry, but a second implementation for Android 2.0-powered devices is under way.

Here's how it works: after installing FriendSensing on a BlackBerry, a user is prompted to create a profile by typing his or her name and e-mail address. The phone sends these two pieces of information, along with its Bluetooth ID and phone number, to a central server over GPRS (General Packet Radio Service). The server handles registration and user profile distribution (in its current version, the server simply consists of a PHP front end and MySQL back end, but these functionalities will be integrated into a Facebook application in the next release). After the user registration step, three software blocks run on the phone (see Figure 4):

- Every 10 minutes, the Bluetooth Manager initiates a scan of its proximity, resulting in a list of Bluetooth IDs and device-friendly names of the phones in proximity.
- For each phone, the Peer Manager collects the number of collocations (frequency) and duration (expressed in time units) and also flags the device as either new (if the phone hasn't been in range before) or seen before (if the phone has been in range in the past). Since it's necessary to filter out at an early stage those phones that have only been in range for short periods of time, we consider a phone belonging to a potential person of interest and begin its long-term tracking only if the phone has been in range for more than 20 minutes.
- Every seven days, the Recommendation Engine produces a list of recommended social contacts, based on collocation frequency or duration; it then downloads the profiles of these people from the server and shows the list on the phone's screen. The user either confirms or rejects each of the recommendations, and the Peer Manager accordingly switches the flags on the corresponding phones from seen before to either confirmed or rejected, to avoid repeated recommendations. Confirmed social contacts will have SensingHappiness enabled and be added on the server's user profile.

Note that both the scan interval (10 minutes) and the aggregation interval (seven days) are tunable parameters. By setting the scan interval to 10 minutes, casual encounters that last only a few minutes are discarded as not informative of actual social relationships. By setting the aggregation interval to seven days, we hope to capture the user's routine, which typically revolves around weekly schedules.

Once a phone is confirmed as a friend, SensingHappiness keeps track of the duration of phone calls and the number of text messages to and from that phone. By doing so, it gauges the friendship strength on more direct forms of communication than simple collocation. After an observation period during which this data is logged (currently set to two weeks, to enable repetition of weekly user behaviors), we can learn activity patterns and their deviations. In the current implementation, such changes manifest themselves when users visualize their social networks: healthy connections are represented as thick edges, and neglected ones as increasingly thinner lines.

The current implementation of the FriendSensing and SensingHappiness technology follows the thick-client model: all the logging and processing happen on the mobile phone. This choice is suitable for privacy-conscious

users who prefer to retain full control over their social activity data. However, an alternative deployment could follow the thin client-thick server model instead: the application on the mobile phone simply collects data and then transfers it to the server for processing and inferencing. This deployment's main advantage is the possibility of transparently updating (and improving) the server's reasoning engine without having to reinstall a new application version on the phone. The next release of the FriendSensing technology will follow this model, with the reasoning taking place server side in a Facebook application and the logging functionality ported to a wider variety of mobile phones.

The storage overhead a mobile phone that uses FriendSensing/SensingHappiness would experience depends on the number of managed profiles: for each one, a phone only needs to store the Bluetooth ID and profile name along with four counters (encounter frequency and duration, number of text messages sent, and duration of phone calls made) and one flag (set to seen before, new, confirmed, or rejected). Even in metropolitan cities, where a phone could keep track of thousands of other devices before they're rejected, the amount of storage used is negligible compared to modern mobiles phones' capacity.

Network communication is considered by far the most severe battery-draining factor on mobile devices, whereas computation has been shown to cause negligible battery consumption.[14] In our technology's current version, with all the processing happening client side, communication is kept to a minimum: once a week, the Bluetooth IDs of recommended friends are sent to the server, and their profiles are pushed back. To use the technology, users will have to leave Bluetooth enabled; as various studies demonstrate, many people already do this,[15,16] so we're primarily interested in determining the impact that frequent Bluetooth scanning would have on battery life. So far, we've run a comparative study with two Bluetooth-enabled BlackBerry Pearl 8210 phones, carried around by the same user for a week, with only one phone running the FriendSensing technology (performing Bluetooth scans every 10 minutes and recording scan results) in both crowded and empty areas; we disabled all other functionalities (such as calling and texting). The handset running FriendSensing had its battery depleted after five days, whereas the phone without FriendSensing (but with Bluetooth still turned on) had roughly 40 percent of its battery capacity remaining at the same point. Although this clearly shows that frequent Bluetooth scanning can have a fairly significant effect on battery life, the phone with FriendSensing still managed to last for five days (not bad for modern mobile devices, especially one running a resource-intensive application). It's important to remember that users can tune the scan interval to achieve a desired trade-off between accuracy of friend predictions and battery life. When moving to the technology's thin-client version, users must transfer all the logged data to the server for processing to conserve battery power. To avoid drainage, users can transfer aggregated data to the server instead of raw data; users can also control when uploads occur (for example, only when at home, when the device is being charged).

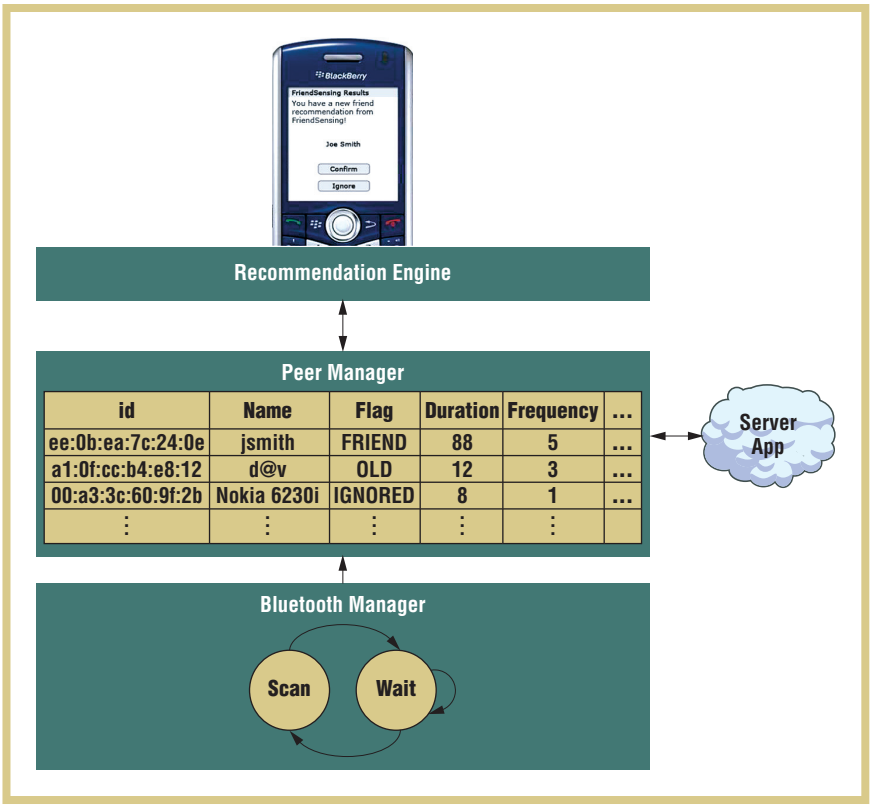
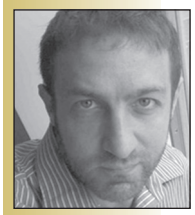


Figure 4. Schematic representation of the FriendSensing implementation. It consists of three software blocks: Bluetooth Manager, Peer Manager, and Recommendation Engine.

We're still experimenting with bringing our research into real-world devices. To this end, our project's next phase exploits PhoneGap, an open source development tool for

the AUTHORS



Daniele Quercia is a postdoctoral research associate at the Massachusetts Institute of Technology. His research interests include computational social science, social computing, and mobile social networking. Quercia has a PhD in computer science from University College London. Contact him at daniele.quercia@gmail.com.

Jonathan Ellis is a third year undergraduate studying MEng Computer Science at University College London. His research interests include mobile computing, social networks, and pervasive computing. Contact him at j.ellis@cs.ucl.ac.uk.

Licia Capra is a senior lecturer in the Department of Computer Science at University College London. Her research interests include mobile systems, pervasive computing, and social networks. Capra has a PhD in computer science from University College London. Contact her at l.capra@cs.ucl.ac.uk.

quickly building mobile applications with JavaScript. The server-side component is supplemented by a Facebook application. Once this cross-platform implementation is completed, we'll run a user study with 12 students enrolled in the same school. The goal will be to understand how adolescents perceive the technology—useful, unobtrusive, tedious, or intrusive—and whether it ultimately helps them nurture their social contacts. P

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments; Sebastian Mueller for his constructive feedback; and Lorenzo Vaiuso, Neal Lathia, and Viktor Presenti for their support in the FriendSensing project.

REFERENCES

1. S. Bhatia and S. Bhatia, "Adolescents' Social Environment and Depression: Social Networks, Extracurricular Activity, and Family Relationship Influences," *J. Clinical Psychology in Medical Settings*, vol. 16, no. 4, 2009, pp. 346–354.
2. K. Hampton and B. Wellman, "Neighboring in Netville: How the Internet Supports Community and Social Capital in a Wired Suburb," *City & Community*, vol. 2, no. 4, 2003, pp. 277–311.
3. K. Hampton et al., "Social Isolation and New Technology," *Pew Internet Report*, Nov. 2009.
4 D. Quercia and L. Capra, "FriendSensing: Recommending Friends Using Mobile Phones," *Proc. Int'l Conf Recommender Systems* (RecSys), ACM Press, 2009, pp. 273–276.
5. V. Kostakosand and E. O'Neill, "Cityware: Urban Computing to Bridge Online and Real-World Social Networks," *Urban Informatics: The Practice and Promise of the Real-Time City*, IGI Global, 2008, pp. 195–204.
6. D. Liben-Nowell et al., "Geographic Routing in Social Networks," *J. Nat'l Academy of Sciences*, vol. 102, no. 33, 2005, pp. 11623–11628.
7. N. Eagle, A.S. Pentland, and D. Lazer, "Inferring Friendship Network Structure by Using Mobile Phone Data," *Proc. Nat'l Academy of Sciences*, vol. 106, no. 36, 2009, pp. 15274–15278.
8. P. Hui and J. Crowcroft, "Human Mobility Models and Opportunistic Communications System Design," *Philosophical Trans. Royal Soc. London*, vol. 366, no. 1872, 2008, pp. 2005–2016.
9. D. Liben-Nowell and J. Kleinberg, "The Link-Prediction Problem for Social Networks," *J. Am. Soc. Information Science and Tech.*, vol. 58, no. 7, 2007, pp. 1019–1031.
10. P. Andre et al., "Experience in Social Affective Applications: Methodologies and Case Study," *Proc. alt.chi*, ACM Press, 2010, pp. 2775–2764.
11. G.G. Van De Bunt et al., "Friendship Networks through Time: An Actor-Oriented Dynamic Statistical Network Model," *Computational & Mathematical Organization Theory*, vol. 5, no. 2, 1999, pp. 167–192.
12. D. Boyd, "Friends, Friendsters, and MySpace Top 8: Writing Community into Being on Social Network Sites," *First Monday*, vol. 11, no. 12, 2006; www.firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/issue/view/206.
13. P. Dodds and C. Danforth, "Measuring the Happiness of Large-Scale Written Expression: Songs, Blogs, and Presidents," *J. Happiness Studies*, July 2009; www.springerlink.com/content/757723154j4w726k.
14. E. Miluzzo et al., "Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the Cenceme Application," *Proc. 6th Int'l Conf. Embedded Network Sensor Systems*, ACM Press, 2008, pp. 337–350.
15. L. McNamara, C. Mascolo, and L. Capra, "Media Sharing Based on Colocation Prediction in Urban Transport," *Proc. 14th ACM Int'l Conf. Mobile Computing and Networking* (MobiCom), ACM Press, 2008, pp. 58–69.
16. E. O'Neill et al., "Instrumenting the City: Developing Methods for Observing and Understanding the Digital Cityscape," *Proc. Int'l Conf. Ubiquitous Computing* (Ubicomp), ACM Press, 2006, pp. 315–332.