Social Data Mining on Smartphones

Semester Thesis

Ajita Gupta

July 7, 2011

Advisor: **Sacha Trifunovic**
Supervisor: **Prof. Dr. Bernhard Plattner**

Computer Engineering and Networks Laboratory, ETH Zurich

Abstract

Smartphones have become increasingly popular and will continue to do so in the near future. Today's applications intend to improve user comfort by enhancing social and collaborative interactions, which allow users to communicate, share and exchange personal viewpoints and experiences. Designing such applications poses an ample challenge, since it entails thorough understanding of **(1) social contacts**, **(2) social interactions** as well as **(3) communication patterns** in order to provide a more *context-aware* service, which is optimized for each individual user.

In this project, we report **SocialMine** as an auxiliary tool to gather versatile smartphone datasets related to the social demeanor. **SocialMine** is a backend Android application, which collects various types of metadata from social applications: Phone- and SMS logs, Contact- and Collocation details along with a large of amount of information available on Facebook (Friends, Hobbies, Message Threads, Wallposts and Likes among many others). The deployment of **SocialMine** allowed us to compare and contrast the contact, interaction and communication patterns graphs for a fixed group of users. We report the substratal evaluation results of a very first deployment conducted at the *Communication Systems Group* at *ETH Zurich*.

Acknowledgements

I would like to express my deepest gratitude to everyone who has accompanied me throughout the course of this project in the past five months.

At the outset, I would like to Prof. Dr. Bernhard Plattner for giving me another wonderful opportunity to work in his group and providing me with a **Nexus One**, the Google Android mobile phone, which has been indispensable for the completion of this project.

I am grateful to Sacha Trifunovic for his guidance, help and patience. He has been an excellent advisor and has readily engaged with me in valuable academic, as well as insightful offbeat conversations.

I have been fortunate to work with Onur Mat, an ETH colleague, who has obligingly assisted me at various occasions. He has always been available for a second professional opinion. Thanks for bearing with me.

Special thanks to the members of an Android forum called *android-hilfe.de*. They were a constant source of inspiration. This project builds upon their technical expertise and input.

My heartfelt thanks to all the participants of the **Alpha Test**, who agreed to share their personal data for analysis purpose and provided us with constructive feedback and suggestions.

Finally, I would like to thank my family for their unflinching support and cooperation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Social platforms, networks and collaboration tools like YouTube, Flickr, Facebook or Google Docs, which allow users to interact, share and express their minds have attained celebrity status in the recent years.

Smartphones in particular, confront us with a colossal pool of possibilities to connect with the people around us: (Video-) games to play, events to participate in, groups to join, photos to share and every conceivable discussion to contribute to. The rapid proliferation of these devices creates bounteous opportunity for novel applications, as well as for extending the realm of existing ones.

Designing of user-focused applications is challenging, as it requires knowledge about various aspects of human behavior. Such, so-called, *context-aware* applications must exploit heterogeneity and behavior patterns in order to make educated decisions according to user preference (without his explicit involvement) and optimize his experience.

This is the point where Social Data Mining Systems step into the picture. Recorded user activities are analyzed and used to deduce the structure (e.g. immediate proximity) and strength of social and mobility ties (e.g. close friend) and classify relations accordingly (family member, friend, coworker, acquaintance, stranger). This classification is imperative for filtering relevant contacts (e.g. by leveraging them).

Thus, by automating the optimization process we minimize user effort and, as a direct consequence, maximize user satisfaction.

1.2 Problem Statement

The goal of this project is to build an efficient Social Data Mining System, which performs an analysis based on the following three criterias:

1. **Social Contacts**
   "How many people is the user connected to (over Contacts List, Facebook Friends)? What can we infer from the intersection region of the two sets?"

2. **Social Interactions**
   "What is the respective medium of communication (Calls, SMS, Face-to-Face Meetings or Facebook)? Are there any overlappings? If yes, how are they related to the tie strength and relationships of users?"

3. **Communication Patterns**
   "What is the frequency and duration of these communication events? What do these patterns tell us?"

To achieve this goal, we developed a backend based Android application called **SocialMine**. A first deployment, an **Alpha Test**, was performed among the members of the *CS Group* for a period of seven days. Data was statistically evaluated in order to provide plausible answers to the points raised above.

Chapter 2

Related Work

In this chapter I will briefly review two studies conducted in the area of data mining. I will first give an overview of the work and then outline the significant differences to situate the approach we have taken in our project.

2.1 Stumbl

Recent work in social data mining focuses on collection of relevant datasets on Facebook. As discussed by Hossmann et al. [1], the aim is to extract the contact, social and activity network of Facebook users in order to make out clear pattern structures between mobility, social connections and communication. This provides invaluable information for algorithm design and in-depth research of operational opportunistic networks.

The implementation is carried out with a Facebook Application called **Stumbl**, which uses a newfangled methodology combining automated data collection as well as periodic user input.

Stumbl addresses the following three issues:

1. **How is the type of the social tie (friend, family, colleague, acquaintance) related to the context, duration and frequency of meetings?**
   "Meetings with family members are long and frequent. Get-togethers with friends are occasional, but lengthy. As expected, encounters with work colleagues are brief and occur quite often."

2. **How are different relationships related to the number of communication events on Facebook?**
   "Friends and family have the highest rate of communication. Colleagues exhibit much lower percentages, whereas acquaintances drop down to a meagre 0.3 events per pair."

3. **Are we more or less likely to communicate with people to whom we have strong mobility ties with or with remote friends, whom we do not run across on a regular basis?**
   "On average, the number of communication events with day-to-day, so-called Stumbl friends, is upto 10 times higher than with normal Facebook friends."

We take the results of the **Stumbl** experiment into account and move one step ahead. In our project, we are not limited to only contrasting Facebook friends amongst each other, but we also include the user's phone and message contacts, their respective communication channel as well as collocation data in our analysis. Furthermore, we also add another Facebook interaction type to our analysis: *Facebook Messages*. This allows us to capture all sorts of interaction patterns (not just face-to-face encounters) of the user's contact circle on and off Facebook.

However, we do not distinguish between different relationship types (Facebook friends, family members, classmates, acquaintances). This classification can be used for a more fine-tuned analysis of social relations and might be an advantageous addition worth considering (see *Chapter 8: Future Work*).

Another significant (positive) difference to the **Stumbl** approach is the fact that user effort is nonrecurring and negligible, since data collection does not depend on his input. This also implies perfect accuracy and correctness of data we acquire, since a meeting or a phone call cannot be *"forgotten to report"* by the sending functionality of our system.

2.2 Device Analyzer

The **Device Analyzer** is an Android application developed by researchers at the University of Cambridge[1], which collects usage statistics and periodically uploads them to a central server in the background while people use their phones.

The goal is to use gathered data sets for the improvement of future smart phones, extract patterns and trends. Besides, personal statistics and recommendations are provided (e.g. the best phone plan based on the user's historical usage or interesting apps based on the ones currently used).

Extracted data includes

1. **Basic Data**: Phone Usage Statistics and Network Info
2. **Applications**: Running Apps, CPU/Memory Usage and Executed Tasks
3. **Hashed GSM Cell IDs**
4. **User Location**

[1] `http://deviceanalyzer.cl.cam.ac.uk`

By downloading the **Device Analyzer** app from the Android market, more than 2100 participants have already contributed to this research project.

Our approach attaches great weight to social data, which is acquired from Facebook, whereas the **Device Analyzer** focuses on general, but richer and rather fine-grained information related to smartphone usage patterns.

Providing the user with his personal collected raw data and live statistics is a stimulating enticement to attract more users for test runs and should be included in prospective design optimizations.

Chapter 3

Design

In this chapter, I will provide an introduction to a Social Data Mining System - the paramount outcome of my project.

3.1 Goal

The system I have designed is optimized to gather user data related to social behavior on Android phones. The intention is to find a connection between the communication patterns of users and their social relations. Apart from showing such correlations, user classification (friend, coworker, family member, acquaintance) can be performed by extracting collocation data and social interactions from various modules, such as the **SMS-Collector**, the **Location-Tracker**, the **Call-Logger**, the **Contacts-Fetcher** or the **Facebook-Miner**, which are part of an Android application (named) **SocialMine**.

3.2 General Setting

Figure 3.1 gives a graphical overview of the setup of our system.
The interaction and cooperation of **SocialMine**, as the local (phone-based) data extraction unit, and the **Server**, as the centralized data storage system, represent the steady backbone of our analysis. User data is invariably collected and stored internally on the phone. Once the connection with the Server is established, data is transferred from the phone (client) to the Linux (server) database over the available network connection. This cycle is repeated continuously for a fixed time interval (or until the application is uninstalled).

In addition, we have a regular HTTP communication stream between the Facebook-Miner module of the Android application and the identically named Facebook application **SocialMine**. Access to the user's account is requested by the phone application and granted by the exchange of a so-called *"access_token"*. There is still room for improvement in this area (e.g. the responsibility of token management can be delegated to the Server[1] or the network connection can be set to match user preference).

[1] Server and Facebook interaction can be realized by using the JavaScript or the PHP SDK (`http://developers.facebook.com/docs/`)

Figure 3.1: Social Data Mining System

3.3 Architecture

In this section, I would like to outline the four main components of my application as shown in Figure 3.2. I will lay out a more elaborated description on each of them in the next chapter (*Chapter 4: Implementation*).

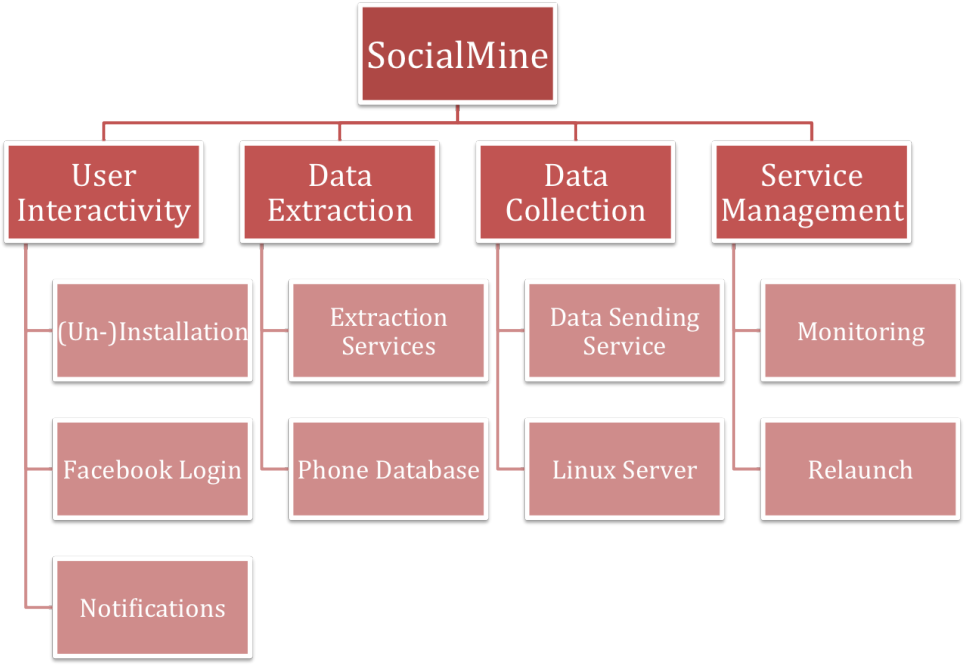

Figure 3.2: *SocialMine* Structure

The fundamental building blocks of the application are the following:

User Interactivity This block comprises of time and effort from the user's perspective, e.g. for granting permission to access personal data. Notification messages to greet, inform and alert the user as well as the (un-)installation procedure are integral adjuncts.

To give the user more control one might add a section devoted to configuration settings (e.g. data extraction modules, sending periods, sending mode - automatic/manual and network connection to name a few).

Data Extraction After gaining access to user data, we extract information from various disjoint services and filter them according to a specified desideratum. Data is stored on the user's local phone database.

Data Collection A prerequisite to data evaluation is data collection. Data retrieved in the previous step is now sent to our Linux Server at ETH in predefined intervals, thereby freeing storage space on the phone.

Service Management This unit is responsible for providing stability to the application by restarting inactive modules. This is rendered by mutual, as well as bootup monitoring of services.

The flawless functioning of the application necessitates each block to fulfil its task correctly and communicate this to the others.

Chapter 4

Implementation

This chapter outlines the schematic view of our Social Data Mining System in detail. The first section gives an overview on the general environment setting, whereas the other four articulate the functional specifications of each component of the Android application (see *Chapter 3.3: Architecture*).

4.1 Overview

SocialMine was programmed in Java using Eclipse Java EE IDE (Version: Helios Service Release 2) for Android Platform 2.2 (API Level: 8)[1]. It consists of a main Activity (*HelloUser*), eight data manipulation and one data sending Service. The Linux Server runs on Ubuntu (Version: 2.6.35-28-generic).

SocialMine communicates to the Server with PHP via HTTP posts and responses. The Server in turn, interacts with the database using MySQL (Server Version: 5.1.41-3ubuntu12.9, Protocol Version: 10).

4.2 User Interaction

SocialMine is a backend application, where the user's sole contribution is his willingness to share personal data. Hence, the demanded human effort and time is minimal.

After installing the application the user receives an address of welcome with a brief introduction to **SocialMine**. He is expected to log in to Facebook and grant us permission to read selected parts of his data - the general Facebook profile, inbox, wall and hobbies, among several others. The application runs silently without disrupting the user's foreground activity.

We provide the option of running the application for a fixed time period, after which an alert message along with an *Uninstall*-button pops up. The user is requested to consummate uninstallation at his earliest convenience.

[1] `http://developer.android.com/sdk/android-2.2.html`

SocialMine also contains the *About*-option in the menu, which holds a summary of the project and our contact information, in case the user wants to get in touch.

4.3 Data Extraction

Before going into details, let's have a look at the different services of **SocialMine**. **SocialMine** is anchored by five major data extraction units, each of which possesses its own batch of Android services. The units are given in the illustration below (Fig. 4.1), along with their corresponding list of services (each represents the name of the respective Java file).

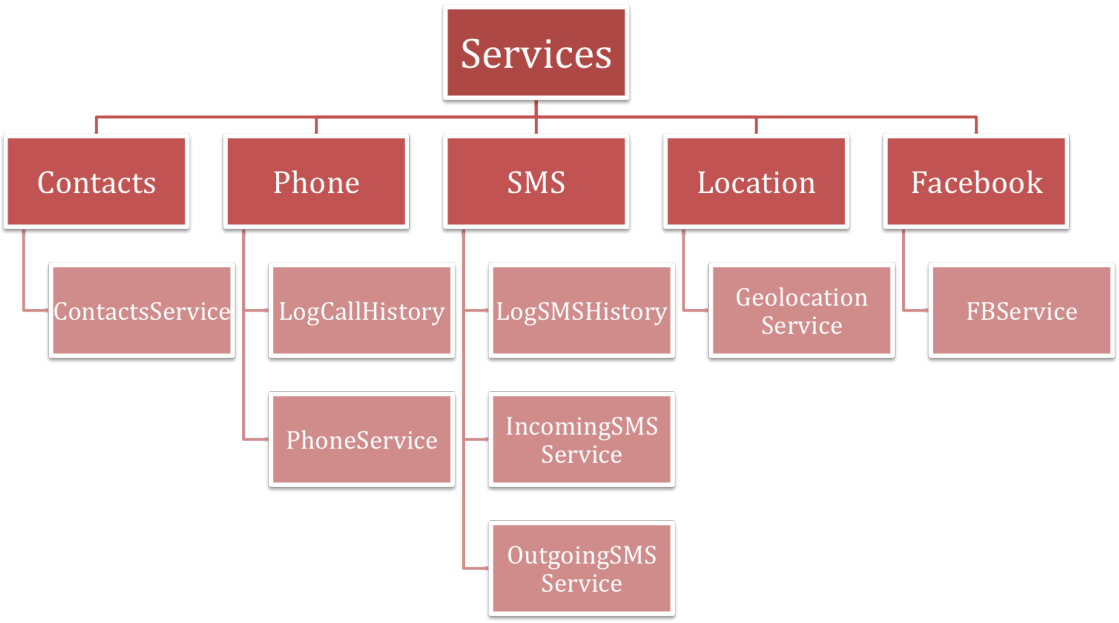

Figure 4.1: Overview of Services

Services are subdivided into three categories as shown in Table 4.1: **One-Time Services**, **Triggered Services** and **Periodic Services**.

Logging of History (Calls and SMS) is a vital, but non-recurring process. Thus, the reason for having a One-Time Service is easily comprehensible.
A Triggered-Service is invoked by a *Broadcast Receiver* once a certain action has taken place. This is a callback, which is registered to the respective event.
A Periodic Service is formed by two cycle-elements: *"On"* (Do Work) and *"Off"* (Sleep).

One might ask why certain services are triggered, whereas others run constantly. This is not arbitrary, but results from an Android-specific functionality, where certain actions (e.g. change of phone state, incoming SMS) are registered in-

stantly by the operating system and signalled to various applications, which call dependent processes. All other actions need to settle for periodic polling to detect updates.
Example: Action *"android.provider.Telephony.SMS_RECEIVED"* is required to activate phone vibration/ringtone processes.
Counterexample: The *"android.provider.Telephony.SMS_SENT"* - Action cannot be registered, since no (urgent) process demands to be triggered by this event.

One-Time Services	Triggered Services	Periodic Services
LogSMSHistory LogCallHistory	IncomingSMSService PhoneService	GeolocationService ContactsService OutgoingSMSService FBService

Table 4.1: Service Categorization

A collection service can either take the **incremental approach**, where only the newest relevant entry is stored, or the **iterative approach**, where all information is collected, irrespective of it having been recorded before.

The first strategy is applied when there is a timestamp attached to the entry of interest. We are compelled to recourse to the second option for all other cases, where there is no reference value available.

4.3.1 Contacts

The Contacts Class (*ContactsService.java*) periodically (once a day) obtains all information (iterative approach) from the built-in *Contacts* application on Android. We extract the following contact fields:

- Contact Names
- All Contact Numbers (Mobile, Work, Home, etc.)
- All Email Addresses (Mobile, Work, Home, etc.)
- All Postal Addresses (Mobile, Work, Home, etc.)
- Organization (Work Institution, Position)
- Notes (Remarks, Linkages between Social Network Friends & Phone Contacts)
- Websites
- All Instant Messaging ID's (WLM, Yahoo, Skype, etc.)

Note: All contacts are collected without differentiation of their type (Facebook, Twitter, SIM, Gmail). Therefore, a distinction mechanism must be implemented to obtain a more accurate connection graph.

4.3.2 Phone

The Phone Service is a bundle of three dependent Java classes, which interact with the internal *CallLog-Provider* of each system.

- *LogCallHistory.java*: This class records the call log pile, which has stacked up until now. This service is terminated thereafter.
- *PhoneServiceReceiver.java*: This so-called *Broadcast Receiver* is invoked as soon as the state of the phone changes, i.e. when the user receives or initiates a call. This receiver then triggers the second service.
- *PhoneService.java*: This service absorbs the newest entry (incremental approach) in its database and destroys itself.

The information we are looking for are Caller, Callee, Timestamps as well as Duration of the Call.

4.3.3 SMS

The SMS Service is another combination of individual classes.

- *LogSMSHistory.java*: All accumulated SMS' are collected. This service starts OutgoingSMSService and terminates thereafter.
- *OutgoingSMSService.java*: This periodic service collects all new messages (incremental approach), which have been sent in the past four hours.
- *SMSReceiver.java*: This Receiver is invoked as soon as the user receives an SMS. The Receiver then creates an instance of IncomingSMSService.
- *IncomingSMSService.java*: The recent SMS is added into the database (incremental approach). The service is stopped thenceforth.

The information we obtain includes Sender, Recipient, Timestamps as well as Length of the SMS.

4.3.4 Location

The Location Service (*GeolocationService.java*) periodically (every minute) queries the Network Connection (Wi-Fi or Mobile).

If a wireless connection is available, the *LocationManager* is invoked. It provides access to the system location services. These services allow applications to obtain periodic updates of the device's geographical location (latitude, longitude).

If there is only a mobile connection (no Wi-Fi), an instance of the *TelephonyManager* is called. The *TelephonyManager* then acquires two values representing the GSM Cell location: Cell-ID (a unique number of a GSM cell for a given operator) and the Location Area Code (LAC). There are some accuracy issues in this latter case, since a cell can cover several hundreds of meters up to several kilometers.
In addition, if Wi-Fi has been enabled, the *WifiManager* gives us a list of **Basic Service Set Identifiers (BSSID's)** and the signal strength of all the access points (AP) nearby. This can constrain the location region of the phone considerably.

4.3.5 Facebook

Facebook is today's most prominent representative of an online social networking platform for cultivating communication and sharing of content among friends. Hence, the Facebook-Miner is the richest of all modules in terms of extracted social content.

Post login and permission grant, an HTTP request (containing user credentials) is sent to Facebook with the *"access_token"* (wrapped in an HTTP response) in expectancy. **SocialMine** is now a trusted application to the user's Facebook account and can read selected user data.

At Facebook's core is the social graph. The new **Graph API**[1] presents a simple, consistent view of the Facebook social graph, uniformly representing objects (each assigned to a unique ID) in the graph and the connections between them (e.g. friend relationships, shared content and photo tags). We have primarily used this API to acquire data. However, since Facebook has not completely transferred the whole functionality to the new API, we had to fall back on the deprecated **Legacy REST API**[2] for obtaining the set of mutual friends between two Facebook users.

One of the most compelling features of Facebook SDK is **Single-Sign-On (SSO)**, an authentication method, which primarily works by redirecting the user to the native Facebook application on his device. He is prompted with an authentication dialog with the respective permissions and then redirected to the application with the appropriate *access_token*. If the user is already signed in, he is not required to type in his identity anew. SSO - capability is provided to an application by the Facebook framework (referring to the SDK) for Android.

Special attention should be paid to the following:

- The framework provided is insufficient for **SocialMine** to work, since the logging-in process is attached to the authentication screen. Hence, it is GUI-based. To avoid this repetitive and bothersome activity, the framework has been extended with a feature to save the *access_token* after login completion and restore it before the next collection round. This functionality is implemented in a static class called *FBSessionHandler*.
- Moreover, we need to extend the lifetime of our access, since the *access_token* (or rather *"session_token"*) is, by default, limited to a few hours and expires thereafter. This issue is solved by requesting lifelong access with the help of an extended permission called *"offline_access"*. This allows our app to perform authorized requests on behalf of the user and obtain fresh tokens at any time.

[1] `http://developers.facebook.com/docs/api/`
[2] `https://developers.facebook.com/docs/reference/rest/`

A Facebook data collection session commences every four hours and takes place in a special Thread inside *FBService.java*. A massive amount of details is downloaded from Facebook over the network connection (Wi-Fi or mobile) via HTTP requests/responses in every session (iterative approach) - all of which are enumerated below:

- User Profile (General Information, Work- & Education History)
- (Common-) Friends (Name, ID)
- Messages[1] (ID's, Participants, Replies, Timestamps)
- Wallposts[1] (ID's, Likes, Comments, Timestamps)
- Hobbies (Activities, Interests, Books, Music, Movies - with respective Timestamps, ID's and Categories)
- Pokes (Poker, Pokee, Timestamp)
- Likes (Name, Category, Timestamp)
- Groups (Name, Group-Position)
- Events (Name, Timestamps, Location)

4.4 Data Collection

Now that we're done extracting data from various services, we move on to the collection segment. The interaction of two elements is crucial for this to function.

4.4.1 Data Sending Service

The Sending Service (*SenderService.java*) periodically (once an hour) sends all the data available on the phone.

Each table of the phone database is mapped to a PHP file on the Linux Server, which executes the necessary operations to store data into the Server-side database. After receiving a success message from the Server, the local database on the phone is erased in order to make room for new data. If it fails, the current state is preserved until the next sending cycle.

For security reasons the phone first authenticates itself to the server. To achieve confidentiality data is encrypted with a symmetric 128-bit block encryption scheme using the *Advanced Encryption Standard (AES)*[2] prior to being sent to the Server IP.

[1] Note: Only the most recent threads are collected. To obtain the complete list one can make use of the *paging* tag included in the JSON response. It contains the URL for the next set of data. This requires the existing framework to be extending in order to issue HTTP commands directly from the application.

[2] `http://en.wikipedia.org/wiki/Advanced_Encryption_Standard`

4.4.2 Linux Server

The Linux Server resides in the server room at the *CS Group*. It contains our prodigious database called **SocialMine**, which stores data for each user[1]. The database is divided into 38 tables assigned to the different extraction modules.

Module	Table (Columns)
Location	GeoLocation (IMEI, Latitude, Longitude, Date, Time) CellInfo (IMEI, Cell-ID, LAC, Date, Time) WifiInfo (IMEI, BSSID, Strength, Date Time)
Phone	PhoneLogs (IMEI, Date, Time, Name, Phone Number, Duration, Call-Type)
SMS	IncomingSMS (IMEI, Date, Time, Thread-ID, Sender, Recipient, SMS-Length) OutgoingSMS (same as above)
Contacts	ContactName (IMEI, Contact-ID, Name) ContactPhoneDetails (IMEI, ID, Phone Number, Number-Type) ContactEmailDetails (IMEI, ID, Email, Email-Type) ContactAddressDetails (IMEI, ID, Address, Address-Type) ContactIM (IMEI, ID, IM, IM-Type) ContactNote (IMEI, ID, Note) ContactOrganizationDetails (IMEI, ID, Company-Name, Occupation-Title) ContactWebsite (IMEI, ID, Website)
Facebook	FBGeneralProfile (IMEI, Name, ID, Gender, Last Update, Birthday, Websites) FBEducationHistory (IMEI, School Name, Education Type, Graduation Year) FBSchoolClasses (IMEI, Class, School Name) FBSchoolConcentration (IMEI, School Name, Concentration) FBClassMates (IMEI, Name, School Name) FBWorkHistory (IMEI, Employer, Description, Start Date, End Date) FBProjects (IMEI, Name, Description, Start Date, End Date, Employer) FBColleagues (IMEI, Name, Employer) FBEvents (IMEI, Name, ID, Starting Time, End Time, Venue) FBGroups (IMEI, Name, ID, Position) FBPokes (IMEI, Recipient, Sender, Time) FBFriends (IMEI, Name, ID) FBLikes (IMEI, Name, ID, Category, Created Time) FBInterests (IMEI, Name, Category, Created Time) FBActivities (IMEI, Name, Category, Created Time) FBMovies (IMEI, Name, Category, Created Time) FBBooks (IMEI, Name, Category, Created Time) FBMusic (IMEI, Name, Category, Created Time) FBMessages (IMEI, Message-ID, Name, Last Update, Total Participants) FBParticipants (IMEI, Name, Message-ID) FBReplies (IMEI, Thread-ID, Replier, Time) FBWallposts (IMEI, Post-ID, Name, Created Time, Updated Time, Type) FBWallComments (IMEI, Post-ID, Name, Created Time, Number of Likes) FBWallLikes (IMEI, Name, Post-ID)

Table 4.2: *SocialMine* Database Structure

[1] Note: User data is distinguished with the help of the **International Mobile Station Equipment Identity (IMEI)**, which is a unique 15-digit serial number assigned to GSM mobile devices.

4.4.3 Communication Session

Let's have a step-by-step look at the communication session between the three key entities: **(Client-Side) Phone Database**, **Linux PHP Server** and **(Server-Side) MySQL Database**.

The graphical overview is given in Figure 4.2.

Step 1) - Prepare data to send: c = enc (m) = enc (IMEI + *authentication-password* + value1 + ### + value2 + ### ...)
Note: The sequence "###" is used to distinguish individual column values.

- Send HTTP Post: http://SERVERIP/tablename.php?encrypted=c

Step 2) - Decrypt data: m = dec (c)

- Verify *authenticationpassword*
- If correct, server authenticates itself to the database using credentials for database *SocialMine* and inserts data
- If incorrect, send HTTP Response "0" to phone database

Step 3) - If data is successfully inserted into the database → "true"

- If an error occurs → "false" is sent to Server

Step 4) - If the Server receives a true, it sends an HTTP Response "1" to activate database pruning

- Otherwise, it sends an HTTP Response "0" to keep database state

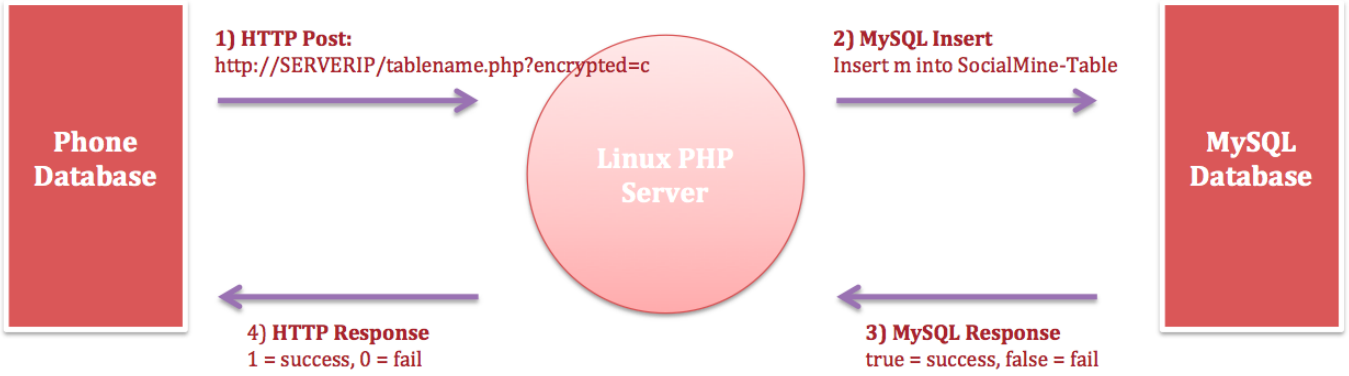

Figure 4.2: Communication Session

4.5 Service Management

Like for any other Android application, **SocialMine** stoppage and crashes are inevitable and cannot be entirely prevented. Thus, we must implement a functionality to resume services and make sure they are constantly running and serving their purpose.

To ensure services resume after phone bootup, the *BootupReceiver*-class restarts all periodic services immediately after receiving the *"BOOT_COMPLETED"*-signal from the operation system. After finishing its own task, each periodic service requests the Android *ActivityManager*, which interacts with all applications running on the system, for the current state of the other services and relaunches them if necessary. This *"All-for-Each and Each-for-All"* strategy guarantees redundancy and robustness to the application.

The implementation of a backup for Triggered and One-Time Services was beyond the scope of this project, but is highly recommended for further extensions.

Chapter 5

Application Deployment

In this chapter I will provide an introduction to the first official test run of our application, the **SocialMine Alpha-Test**.

5.1 Deployment Specifications

The **Alpha-Test** was held with the intention to gain first experiences with **SocialMine** as an application and the user (usage) behavior related to it. It led to a first interesting dataset, which was dissected and analyzed rigorously.

At the beginning, we recruited test candidates with a personal face-to-face request and word-of-mouth recommendation, which was followed by sending them an Invitation Email[1]. This led to a total of 9 users (including us) giving us their consent to mine and evaluate private social data. The participants were either members (research assistants) or project associates (Master students) of the *Communication Systems Group* at *ETH Zurich*.

The test ran for a total of seven days between Friday, June 10th 2011 and Friday, June 17th 2011. After a week's time, all users were appealed for providing feedback and personal observations. The survey was carried out via an online form on *Google Docs*[2].

5.2 Deployment Evaluation

Recorded data did not only give us information about users, it also revealed internal details about **SocialMine**.

Judging by the recorded timestamps of individual Services, we compute that the uptime of **SocialMine** Extraction Modules was 60% - 75% in average (Contacts: 75%, Facebook: 50%, SMS: 60 %, Phone: 100 %, Location: 70%). Services were turned off either due to an application crash or lack of memory space. However, they were all resumed at phone bootup.

[1] More details are provided in Appendix B.1

[2] The Feedback Mail and Survey Form Template are attached in Appendix B.2

We conclude that more emphasis must be put on the *Service Management* component (by periodic monitoring) to decrease downtime and maximize the amount of collected data.

Chapter 6

Data Evaluation

This chapter is entirely devoted to the evaluation results of the Alpha Test.

Stored Data was filtered with general MySQL commands[1]. Mathematical calculations to obtain mean, variance, standard deviation values and distance vectors[2], as well as algebraic operations for data structuring were performed by MathWorks MATLAB (Version: R2010b). The Charts Tool of Microsoft Word (Version: 14.0.0) was used to depict all evaluation results graphically.

We have inspected three different social dimensions and sorted the results into the following graphs: **Contact Graph**, **Interaction Graph** and **Communication Patterns Graph**.

We also take a look at the results obtained from the User Survey performed at the end of the test period.

6.1 Contact Graph

The **Contact Graph** represents the number of social ties of **SocialMine** users. These ties were formed either by adding a friend to the internal phone *Contacts* or to the online *Facebook friends list*.

The average number of user contacts is 308 (Standard Deviation: 349.5). Users have 208 Facebook friends in average[3] (Standard Deviation: 131.4), which is much higher than the count provided by Facebook Statistics[4] (130).

On average, there are 61 overlappings between the two sets. This tells us that is still a high amount of separation between *online* and *real* social ties. This can be explained by the fact that adding people to one's Facebook community is easy and does not necessarily infer a personal relationship, whereas a phone

[1] For more information, check *Appendix A.4 Operating the Server*

[2] Credit to MATLAB Script written by Langqiu Sun:
`http://www.mathworks.com/matlabcentral/fileexchange/5256-pos2dist`

[3] The corresponding plot is given in Figure B.1

[4] `http://www.facebook.com/press/info.php?statistics`

contact suggests that a minimum of one face-to-face meeting has taken place (with very few exceptions).

6.2 Interaction Graph

Our Social Data Mining System records four different communication channels: Calls, SMS, Location (face-to-face encounters) and Facebook-Interaction (Messages, Wallposts, Pokes, Events) as well as two different contact sets (Contacts list, Facebook friends).

With the help of the **Interaction Graph** (Fig. 6.1), we intend to show the correlations between the contact sets of individual communication mediums.

To illustrate these correlations optimally we have elected to depict them in form of a **Graph**, where each red node stands for a communication channel and a blue node represents a set of contacts. Each correlation pair is represented by an arrow. The numbers (connected to the edges and nodes) indicate the average count of people in the (intersection) region of the corresponding node(s).

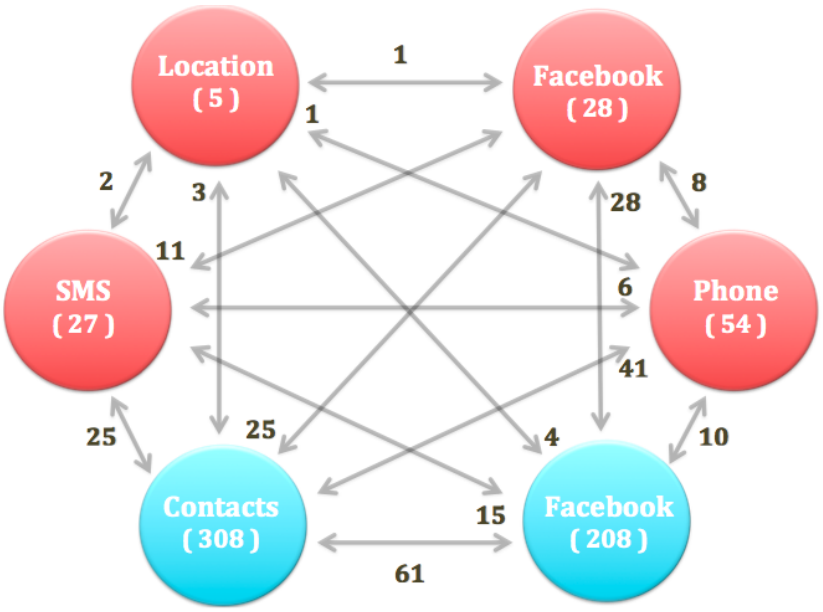

Figure 6.1: Interaction Graph

Example 1: An average user communicates to 54 different people over phone calls.
Example 2: There are 61 overlapping connections between a user's Contacts list and his Facebook friends.
Example 3: A user stays in touch (through Pokes, Messages) with 11 out of 27 SMS contacts over Facebook.

From the illustration above (Fig. 6.1), we conclude that personal and regular interaction over multiple mediums (e.g. SMS and Phone, SMS and Facebook, Phone and Location) is (as expected) limited to a few people (close relations).

These correlations can help to classify contacts (e.g. a person you do not meet, but constantly communicate to over Facebook is most probably a stranger or a remote friend) and determine the tie strength of a relationship (a person you meet, message to and talk to over the phone can be considered as someone you are close to).

6.3 Communication Patterns Graph

Now we turn to our last and final graph. The **Communication Patterns Graph** has a deeper look at the user's social behavior and derives patterns from the recorded data.

We try to provide an answer to frequency-related questions (*"How many calls does a user attend every day?"*, *"How does the daily SMS traffic look like?"*, *"How often does the user interact on Facebook?"*, *"How many people does he meet daily?"*), as well as duration-specific questions (*"What is the average duration of a call?"*, *"What is the average length of an SMS?"*, *"How long do two people interact personally?"*, *"How does the time expenditure of a user look like in the course of a day?"*).

Each medium of communication is considered separately.

6.3.1 Phone Data

We conclude that a user is exposed to 6.7 calls on average every day out of which 1.8 are incoming, 3.5 are outgoing and 1.4 are missed calls. The average duration of a call sums up to 2 minutes 24 seconds for incoming and 1 minute 24 seconds for outgoing calls[1].

6.3.2 SMS Data

An average user receives 1.6 SMS and sends 1.4 SMS each day. Thus, the daily traffic amounts to 3 SMS interactions in average. The mean value for SMS length is 73 characters (84 for outgoing and 69 for incoming ones)[2].

6.3.3 Location Data

The Time Schedule[3] tells us that an average user usually spend 6 hours 46 minutes at his workplace (ETH in our case), 13 hours 31 minutes at home, 1 hour 24 minutes for (outdoor) meals and travels 2 hours 17 minutes each day.

[1] Detailed statistics are shown in Figures B.2 and B.3
[2] Plots representing SMS Frequency and SMS Duration are given in Figures B.4 and B.5
[3] The Schedule is depicted in Figure B.6

The Collocation Graph[1] reveals that an average user usually meets 5.78 work colleagues a day, with whom he interacts[2] for 2 hours 21 minutes.

6.3.4 Facebook Data

Judging from the most recent entries collected on Facebook, users have an average of 13 Facebook Message Threads, 22 Wallposts, 8 Likes, 0.8 Pokes (from and to), 26 Groups and 0.4 Events they take part in[3].

6.3.5 Summarizing Remark

We have recorded user behavior based on five social aspects and derived statistical averages and patterns from them.

These habitual rituals can be used to characterize users and gather information about their environment (e.g. high work activity can point towards an upcoming important event or indicate workaholism) and tie strength (the more often and longer we interact with someone, the closer the bond) of relationships. We can also derive user preferences from these activities (Is the user a *Call*-, an *SMS* or rather a *Facebook*-person?).

6.4 Data Amount Statistics

Over the seven days we collected 14.12 MB in total for our nine test candidates. This gives us 1.57 MB per person for seven days and 230 KB per day.
This can be considered as an acceptable amount of data.

Our calculations show that the maximum amount of data comes from the Contacts-Fetcher (1.34 MB per day per person). The Facebook-Miner ranks second with a high margin (143.4 KB per day per person), followed by the Location-Tracker and the SMS-Collector (125.14 KB per day per person), whereas the Call-Logger occupies the least amount of database storage (52.34 KB per day per person)[4].

Table 6.1 shows a few calculations for reasonable test periods and candidate groups, which are expected for future test runs.

[1] The graph is portrayed by the collocation duration (Figure B.7) and collocation frequency (Figure B.8)
[2] The terms *meet* and *interact* in this context are assumed if two parties are within reach of 5m from each other
[3] Facebook statistics are depicted in Figure B.9
[4] The corresponding percentage for each Module is given in Figure B.10
[5] These would be reasonable deployment parameters for the **Beta-Test**

Test Candidates	Test period	Data Amount
1	1 day	230 KB
1	1 week	1.57 MB
1	1 Month	7 MB
1	6 Months	42 MB
1	1 Year	84 MB
50	1 Month	350 MB[5]
100	6 Months	420 MB
1000	1 Year	82 GB
5000	3 Years	1.2 TB

Table 6.1: Data Statistics

6.5 Feedback Results

In this section we look at the statistics obtained from the Feedback Survey in more detail.

6.5.1 General Information

A total of eight (out of nine) test candidates took part in the Feedback Survey.

The most commonly used phone model was a Nexus One (4 people), followed by HTC Desire HD (2 candidates), HTC Desire and Samsung Galaxy S (1 user each).

A total of 5 users never turned off their phones, whereas there is a single entry for all other options (Once in a few days, Daily at night, Daily at another time).

Only two of the candidates had enabled mobile network connection, whereas all others used Wi-Fi to collect data and forward it to the Linux Server.

As requested, each user logged in to Facebook.

6.5.2 Application Statistics

The majority of the participants (5 users, 63%) did not notice any application malfunctionality and only one single person was confronted with an unusual high amount of crashes[1].

Resource consumption was moderate[2]. On the whole, battery was discharged noticeably often (for 5 candidates). This has two major reasons: For once, the Contacts-Fetcher, the Phone-Logger and the SMS-Collector triggered Android Acore Processes (in order to access the Contacts Database and the CallLog-Provider), which drained the battery. A bug in *OutgoingSMSService* reinforced this consumption by causing the SMS collection start to from scratch in every

[1]This is given in Figure B.11
[2]The results are depicted in Figures B.12 and B.13

session (iterative, instead of the planned incremental approach). The operation speed remained constant for 87.5% of the people.

6.5.3 Suggestions for Improvement

The most striking and recurring comments concerned application transparency (i.e. *"What/How much data is being collected?"*, *"How do I know if everything is working fine?"*).

Live statistics as well as general averages were requested. A personal concern regarding privacy was expressed along with the suggestion to use hash values (for IMEI's) for data anonymization.

We perceive the encouraging feedback and the constructive suggestions as a motivation and hope to meet user expectations even better in future.

Chapter 7

Discussion

This chapter deals with two issues, which have been either deliberately neglected or suppressed up to this point. Let's first have a closer look at the boundaries, which confine **SocialMine** before we dive into one of the most prevalent and controversial topics of today: *Privacy.*

7.1 Limitations

First-off, it's the infrastructure, which includes the type of the internet connection (Wi-Fi, 2G or 3G), the transfer bandwidth as well as the storage capacity of our (client and server) databases that determine transmission speed and the amount of data we obtain.
Another crucial aspect is the resource reservoir. Battery consumption, computing power and financial restrictions[1] negatively influence data collection activity.

SocialMine is a background application. Hence, it lacks the **X-factor**, which appeals to the user. Therefore, the integration of a foreground user-focused activity (e.g. a game, personal statistics, a diary) is essential to gain attention and popularity amongst the masses.

According to a statistical analysis performed by **Gartner**, the world's leading Information Technology research and advisory company [2], last year, the share of Android devices in the industry is merely 22.7%, where Symbian continues to dominate the smartphone industry.
This is another drawback, since the location data is predominantly restricted by the environment, which the application is deployed in. It implies scattered and geographically dispersed information, which yields a narrow and one-sided set of collocation data[2]. The higher the density and the number of contact pairs, the more precise the subsequent analysis[3].

[1] Smartphones are owned by just one third of all mobile phone users. The market is concentrated on classes, where consumers have more disposable incomes and where networks are fast enough to explore the entire smartphone feature package.

[2] The Collocation Graph we designed as part of data evaluation was restricted to analysis of work colleagues.

[3] More accurate classification of relations will be feasible (e.g. people encountered at home are family members, the ones at leisure activities are friends).

However, things likely to change in favor of **SocialMine**, given the rising prominence and acceptability of smartphones - Android, in particular.

7.2 Privacy

The mobile phone of today has the capability of replacing a GPS, a bookstore, a shopping tour or even a social community. The exponential technological progress over the past decade has brought the world to users' fingertips, but at what cost?

The very same convenient and entertaining technology is used to gather personal (confidential?) user information and share it with advertisement agencies for profit. Data mining from smartphone applications has become pandemic. An investigation conducted by the Wall Street Journal [3] at the end of 2010 discovered that sensitive data was being sent to third parties by 101 popular applications - 56 of which transmitted the unique identifier, while 47 transmitted the phone's location. A classic example is the mainstream iPhone game called **Angry Birds**, which regularly sends user coordinates along with his IMEI to marketing companies - completely without user awareness, hereby making it impossible for him to *opt out*.

Data collection can be perceived as a *double-edged sword*, since it is a substantial necessity for numerous famous applications. Pinpointing location data is what allows users to check-in on *FourSquare*, acquire objects on *Gowalla* or locate the nearest cinema hall with the help of *Google Maps*.

The objective of **SocialMine** is by no means lucrative. Instead, we intend to collect personal data in order to identify user behavior and social patterns.

Within the scope of a study called *"Reality Mining"* [4] performed a few years ago, MIT professor Sandy Pentland claimed that by taking advantage of sensors in cell phones, such as the microphone or the accelerometers, one could extend the benefits of data mining into personal health care. Indications to depression could be found by observing the way a person speaks: Depressed people tend to talk more slowly - a change that speech analysis software on a phone is more likely to recognize than family or friends. Monitoring a phone's motion sensors might reveal slight changes in gait, an early indicator of ailments such as the Parkinson's disease.

Behavior-logging technology, if not imposed (by providing with an initial consent statement), can be truly beneficial. We must now ask ourselves the following: *"Is what I am receiving worth the exchange?"*.
Given the stardom and supremacy of Facebook as a social network, we can conclude that people are open to share their privacy if there is a personal gain or entertainment factor in for them.

This shall be taken into account in the succeeding optimizations of our work.

Chapter 8

Future Work

This chapter provides an outlook on the future of mobile data mining, making advanced development and optimization of **SocialMine** a focal point of discussion.

This project can be regarded as the pilot stage of an iterative design process of a Social Data Mining System. It offers the possibility to incorporate (technical) findings into further iterations, the next of which will be the classification and profiling of user behavior, the master thesis of my work colleague, Onur Mat.

In hindsight, we identify five major areas for further improvement.

1. **Data Extraction Modules** The Facebook-Miner can be extended by collecting the entire message and wallpost database (instead of just the most recent threads). Token management can be outsourced to the Server, thereby reducing the amount of formal activities on the application.

   The Contacts-Fetcher needs to be modified in a way to distinguish between *actual*, *automatic* and other types (Facebook, Twitter, Gmail) of contacts and thus, provide us with a more relevant range of friends.

2. **User Control** In order to offer more transparency to the user, we must consider a *Settings* section, where the user is given the option to in/exclude extraction modules, choose his network connection (Wi-Fi, 2G, 3G) as well as define sending periods and mode (automatic, manual) of preference.

   The option of viewing raw data is also worth including, since it comforts the user (regarding his privacy).

3. **Service Management** To maximize the amount of collected data and provide with accurate analysis results, we require Service stability. This can be implemented by periodic or triggered monitoring (e.g. with a *"Screen-On"* event) and Service relaunch.

4. **Data Anonymization** The protection of user identity remains an unsolved issue. A common concealment technique is the use of hash functions or

look-up tables. This has been recommended by a test candidate of the Alpha Test and encouraged by the **Device Analyzer**.

5. **User Attraction** At this point, we must put more emphasis on the design of an incentive, which draws user attention and distracts him from the collection activity running *behind-the-scenes*. This can be either based on usefulness (e.g. a diary), interest (e.g. live statistics of user behavior) or even a competition-oriented game (e.g. object collection similar to *Foursquare* and *Gowalla*).

The addition of the assets named above will empower us to reach out to a larger audience in the forthcoming second step. This will be conducted in the form of a large scale deployment, a **Beta-Test**.

Chapter 9

Conclusion

We have presented a Social Data Mining System built upon a backend oriented Android application called **SocialMine**.
The System collects a wide range of social metadata with the help of five different data extraction modules (Phone-Logger, SMS-Collector, Location-Tracker, Contacts-Fetcher and Facebook-Miner).
The analysis of the dataset obtained from a user study allowed us to inspect three different social dimensions and create corresponding graphs.

1. **Contact Graph**
   *"We conclude that there is a non-negligible gap between the user's **online** and **real** social ties. This observation can be justified by taking into account that the virtual friends circle does always imply a personal relationship, whereas the contacts list preponderantly covers the set of people, whom the user communicates with directly (e.g. over phone calls or SMS)."*

2. **Interaction Graph**
   "Personal interaction on a day-to-day basis over multiple channels (e.g. SMS and Facebook, Phone and Location) is restricted to a small set of users. These overlapping statistics can be helpful to categorize contacts (family, friend, acquaintance, stranger) and determine the tie strength of a relationship (a person you meet, message to and talk to over the phone can be considered as someone you are close to)."

3. **Communication Patterns Graph**
   "Behavior patterns can be used to characterize users and gather information about their environment, the tie strength (the more often and longer we interact with someone, the closer the bond) of relationships as well as deduce user preferences (Favorite means of communication: Calls, SMS', Facebook or Face-to-Face Meetings?)."

Many handheld devices now have the processing power and storage capacity of low-end desktop computers and can provide data, where users, resource and applications are mobile. Mobile data mining is not in a mature phase yet. However, it represents a promising area for researchers and professionals such as Pentland and Eagle [4], who are making consistent progress at connecting and rationally utilizing all the information available.

Appendix A

SocialMine - Instruction Manual

This part of the appendix provides a basic introduction to the technicalities of **SocialMine**. Anyone who wishes to modify or extend it should use this tutorial as a starting point.

A.1 Getting Started

Before the actual coding, we need to set up the right work environment. Carry out the following steps to do so:

1. I highly recommend *Eclipse Classic* as Java editor. Download the latest version from here[1].
2. You must install the *Android SDK*, as well as the *ADT plugin*, the combination[2] of which yields a powerful and integrated environment to build Android applications.
3. It's time to create your very first Android application: *Hello World*[3].
4. Since Facebook is an integral part of **SocialMine**, you need to arm your workspace for it. Follow the SDK documentation[4], but do not generate a new key (use the original one provided as a supplement) or edit application settings in the *Mobile and Devices* section of the corresponding Facebook application.
5. Now that the client side is ready to send, you must prepare the server side for receiving and storing data on a database. The server can either exist locally[5] or you can devote an entire machine to this task. Keep in mind that you must adjust the Server-Path accordingly in *SenderService.java*.

[1] `http://www.eclipse.org/downloads/`
[2] `http://developer.android.com/sdk/installing.html`
[3] `http://developer.android.com/resources/tutorials/hello-world.html`
[4] `http://developers.facebook.com/docs/guides/mobile/#android`
[5] XAMPP is an Apache distribution containing a Webserver, MySQL, PHP and Perl. Download Link: `http://www.apachefriends.org/de/xampp.html`

Note: In case you own an Android phone, you should run your application directly on it (not on the Emulator). Make sure you have turned on *USB Debugging* under *Development Settings*.

A.2 Coding

A.2.1 Modeling

Modifying, adding or removing services is easily feasible, as each of them is a pluggable block of classes. After deciding on the unit you intend to edit, you should select the corresponding Java files (see *Chapter 4: Implementation*) and begin.

One-Time Service This type of service is called right after installation from the main activity *HelloUser*. The principal functionality resides in the *onCreate()*-Method. To destroy the service after work, we call the *stopSelf()*-Method.

Triggered Service This type of service is invoked from a *BroadcastReceiver*. It's execution is identical with a **One-Time Service**. If you wish to add one of these, you must register an appropriate trigger-variable inside the *AndroidManifest.xml* file to the *BroadcastReceiver*, which is associated with it, e.g. "PHONE_STATE".

Periodic Service Data extraction takes place inside the *run()*-Method of an *Updater-Thread*. We have defined two time-variables *Delay* and *Session-interval*, which determine the waiting period between selected operations and individual collection rounds, respectively.

Facebook A Facebook-instance is generated in the following manner:

```
public static final String APP_ID = "number";
private Facebook mFacebook;

mFacebook = new Facebook(APP_ID);
FBSessionHandler.restore(mFacebook, getBaseContext());

if (mFacebook.isSessionValid())
{
    Intent f = new Intent();
    f.setAction("com.project.socialmine.FBService");
    getBaseContext().startService(f);
}
```

Table A.1: Facebook Instantiation

If you plan to add new features, you can either use the new **Graph API**[1] (recommended) or the old **Legacy REST API**[2] .

```
private AsyncFacebookRunner mAsyncRunner;
mAsyncRunner = new AsyncFacebookRunner(mFacebook);

\\Call the Graph API \\
mAsyncRunner.request("graphpath", new SampleRequestListener());

\\Call the Legacy API \\
Bundle parameters = new Bundle();
parameters.putString("method", "methodname");
parameters.putString("parametername1", parametervalue1);
parameters.putString("parametername2", parametervalue2);
..
..
mAsyncRunner.request(parameters, new SampleRequestListener());
```

Table A.2: Facebook API Call

Note: If you plan to add a new or remove an existing service from **SocialMine**, do not forget to in/ex-clude corresponding declarations and permission entries inside the *AndroidManifest.xml* - File.

A.2.2 Personal Advice

One of the very first tutorials[3] I worked through in the course of my project demonstrated an efficient style of writing code. I would like to pass this on.

1. Define a *TAG* for each class you create:
   private static final String TAG = NameOfMyClass.class.getSimpleName()

2. Log your code. If you are ignorant to what the application is doing, you fill never be able to fix errors. Example:
   Log.d(TAG, "Show me the output: " + foo.result())

3. Logs are printed on the internal Eclipse view pane *LogCat*. However, since it does not distinguish between individual applications, you must filter your output for the *TAG* you specified previously. The SDK comes with the so-called *ADB-Tool*, which lets you communicate with an emulator instance or a connected Android-device. It can be invoked from the *platform-tools*-directory inside the SDK by issuing the following command:
   hostname:platform-tools hmustermann# adb logcat *:S NameOfMyClass

[1] `http://developers.facebook.com/docs/api/`
[2] `https://developers.facebook.com/docs/reference/rest/`
[3] `http://marakana.com/techtv/android_bootcamp_screencast_series.html`

A.3 Running *SocialMine*

SocialMine can be executed in two ways (apart from emulation). Both are described below.

Eclipse You can run the application directly from the Eclipse workbench. However, you will not be able to log in to Facebook, since your application is not signed[1].

Application Package Another option is to run the application as a standard apk-file. In order to do so, you have to first sign your file with the key provided[2] and the credentials given in Table A.3.

Existing keystore	debug.keystore
Password	android
Alias	androiddebugkey
Password	android

Table A.3: Application Key Signing Credentials

The application signing key is the one which authenticates the Android application to your Facebook application. In case the key is misplaced or lost, you will have to create another one by following the Facebook SDK documentation[3]. The *alias* password is the same as the *keytool* one. Choose it wisely and place the key at an apt location on your system.

Note: Before running your application on the phone, you must enable *Unknown Sources* under *Application Settings*.

A.4 Operating the Server

XAMPP comes with a user-friendly interface, which allows you to create, modify, delete and view databases and tables efficiently. Using the terminal for database interaction is rather cumbersome.

Below is a list of the most common MySQL-commands, which will come in handy during manual communication[4].

[1] SSO authentication demands a valid key
[2] Eclipse: Right-click on Project → Android Tools → Export Signed Application Package
[3] `http://developers.facebook.com/docs/guides/mobile/#android`
[4] MySQL 5.1 Reference Manual: `http://dev.mysql.com/doc/refman/5.1/en/index.html`

\\Create Database \\
create database MYDATABASE;

\\Select Database \\
use MYDATABASE;

\\Delete Database \\
drop database MYDATABASE;

\\Create Table With Two Columns (First with String-, Second with Number-Value) \\
create table MYTABLE(id INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id),
MYCOLUMN1 VARCHAR(30), MYCOLUMN2 INT)")

\\Show All Tables In Selected Database \\
show tables;

\\Insert String-Value Into Table Column \\
insert into 'MYTABLE' ('MYCOLUMN') values('"MYSTRINGVALUE"');

\\View Table \\
select * from MYTABLE;

\\Filter Table To Show Specific Column \\
select MYCOLUMN from MYTABLE;

\\Filter Table To Show Distinct Column-Values \\
select distinct MYCOLUMN from MYTABLE;

\\Delete Entries Inside Table \\
delete from MYTABLE;

\\Delete Table \\
drop table MYTABLE;

Table A.4: MySQL Commands

Appendix B

Alpha-Test

B.1 Invitation E-Mail

Dear all,

As salient part of our project on Social Data Mining and User Behaviour Profiling we are conducting an Alpha Test for our application SocialMine.

The goal of this project is to analyze smartphone data related to social behavior. We expect to find a connection between the mobility pattern of users and their social relations. To do so we will be collecting different kind of social data - SMS, Call and Phone logs, as well as your Contacts and general data listed on your Facebook profile.
The Alpha-Test will run for 7 days and then deactivate itself. No user interaction is necessary during the test and power consumption is negligible.

Data is encrypted before being sent to our collection server and will also be entirely pruned after data evaluation and completion of the project.
The setup of SocialMine requires minimal effort from your side.
Please carry out the following steps:
1) Enable Unknown Resources (Settings → Application)
2) Install SocialMine.apk from the attachment
3) Open SocialMine and login to Facebook

We appreciate your participation and are looking forward to suggestions and feedback for further improvement of our project.

Best Regards,
Sascha Trifunovic
Ajita Gupta
Onur Mat

Attachment: SocialMine.apk

Table B.1: Invitation E-Mail

B.2 Feedback Form

The feedback form was created using Google Docs[1] and sent out to the participants at the end of the test period.

Dear all, As you might have noticed, the Alpha Test for SocialMine has come to an end. Your input and suggestions are vital for further improvement of our application. We therefore kindly request you to take a few minutes of your time to take part in the Feedback Survey. You can access the form under the following link: https://spreadsheets.google.com/spreadsheet/viewform? formkey=dHdwc2J1d3pNblhIQ2tJX3hQTzlwSEE6MQ#gid=0 Please let us know if you have faced and crucial difficulties or problems during the test period. We appreciate your participation and are looking forward to your feedback. Best Regards, Sascha Trifunovic Ajita Gupta Onur Mat

Table B.2: Feedback E-Mail

[1] `https://spreadsheets.google.com/spreadsheet/ccc?key=twpsbuwzMnXHCkI_xPO9pHA`

Feedback Survey: SocialMine Alpha-Test

Please take a few minutes to fill out the form below.
Your feedback is essential for further improvement of our project.

* Required

General

Full Name *

Device IMEI

Call the number *#06# to view 15-digit serial

Phone Model *

Examples: HTC Desire, Samsung Galaxy S, Motorola Milestone, Nexus One & Co.

How often did you shutoff your phone in the past 7 days? *

Shutting off refers to a longer period (> 6 hours)

- (•) Never
- ( ) Once in a few days
- ( ) Once a day (at night)
- ( ) Once a day (another time)

Have you enabled data roaming for mobile network? *

Network connection: 3G, 2G

- (•) Yes
- ( ) No

Project Information

Was the information provided in the invitation mail sufficient? *

- ○ Yes
- ○ No

If not, what additional information should be included?

Was the information provided in the 'About'-Section sufficient? *

- ○ Yes
- ○ No

If not, what additional information should be included?

Application Usage

Did you login to Facebook? *

- ○ Yes
- ○ No

If not, what was the reason?

- [ ] I don't have a Facebook account.
- [ ] I tried, but the it didn't work (e.g. due to crash).
- [ ] I didn't want to.
- [ ] Other:

How often did you notice a SocialMine application crash? *

- ○ Never
- ○ A few times
- ○ Unusually often

Resource Consumption

Did you notice any substantial change of battery consumption during the testperiod? *

Yes, a lot (Battery was discharged very quickly)

Did you notice any remarkable change in the operation speed of the phone? *

Yes, a lot (Phone functionality slowed down significantly)

Remarks

Feel free to add additional comments, personal observations or suggestions.

Submit

B.3 Data Evaluation Plots

This section includes all the different plots used to depict statistical average values, which are listed in *Chapter 6: Data Evaluation*.

B.3.1 Contacts Graph Plot

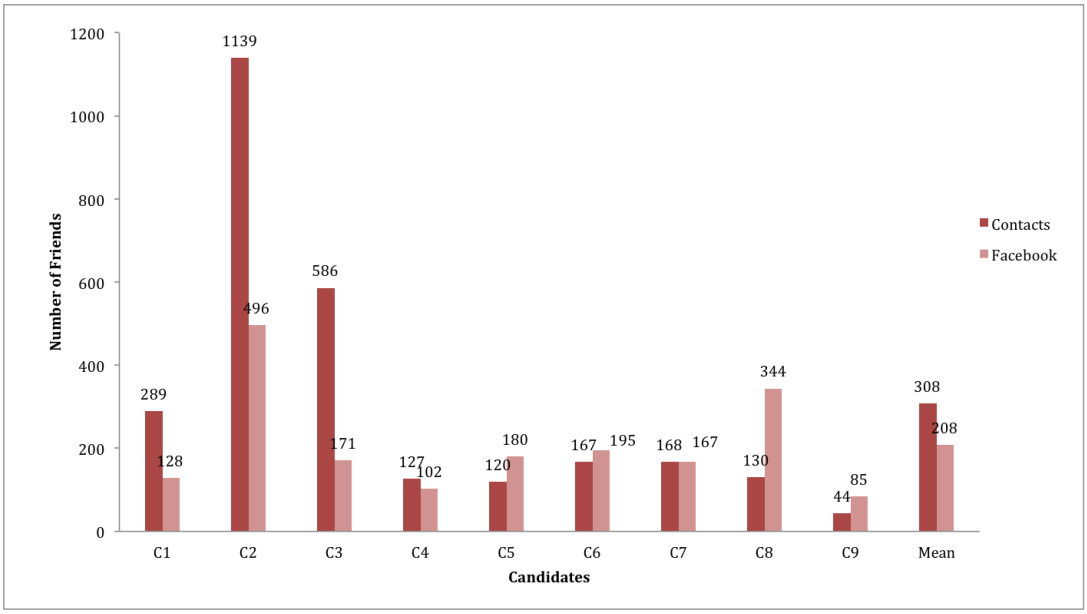

Figure B.1: Contacts Graph

B.3.2 Communication Patterns Graph Plots

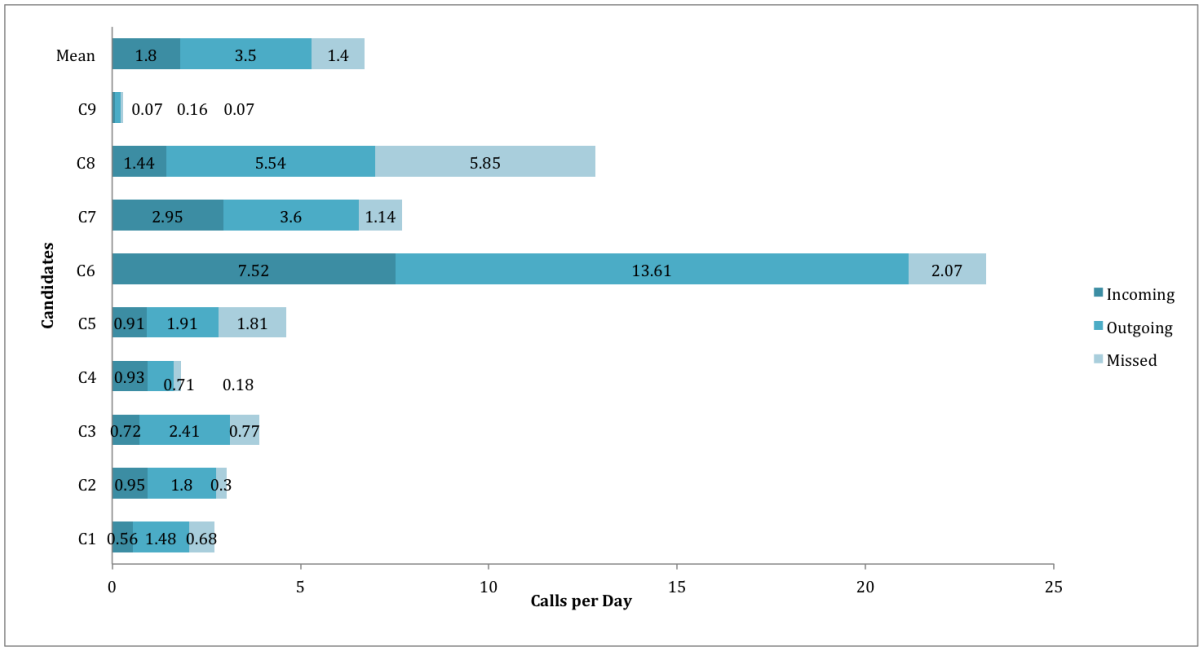

Figure B.2: Calls per Day

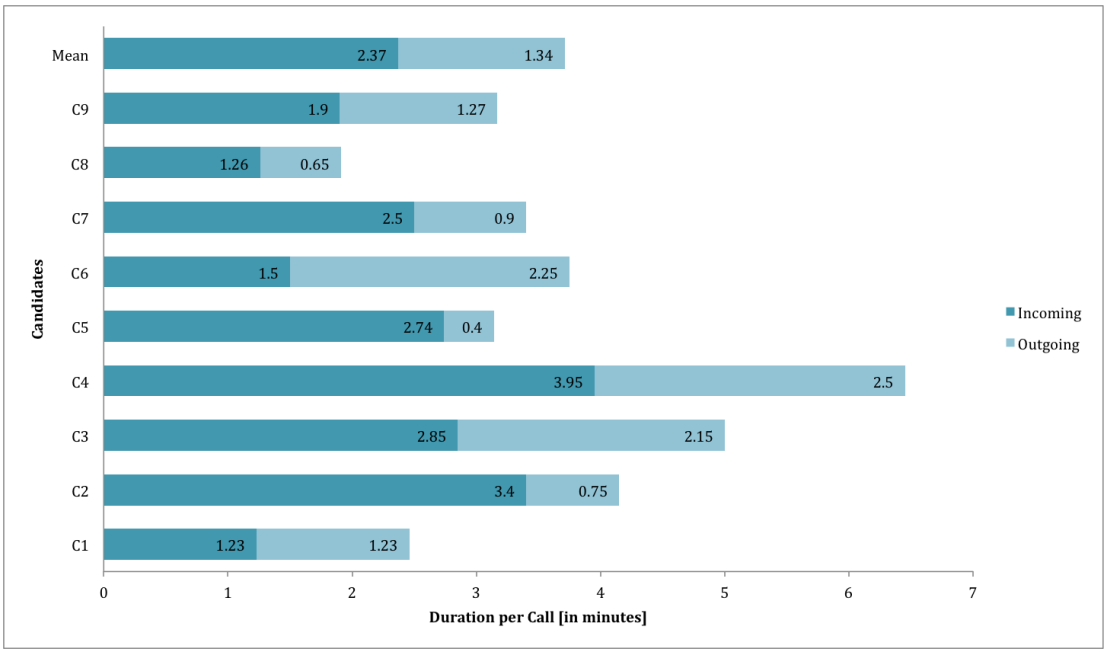

Figure B.3: Duration per Call

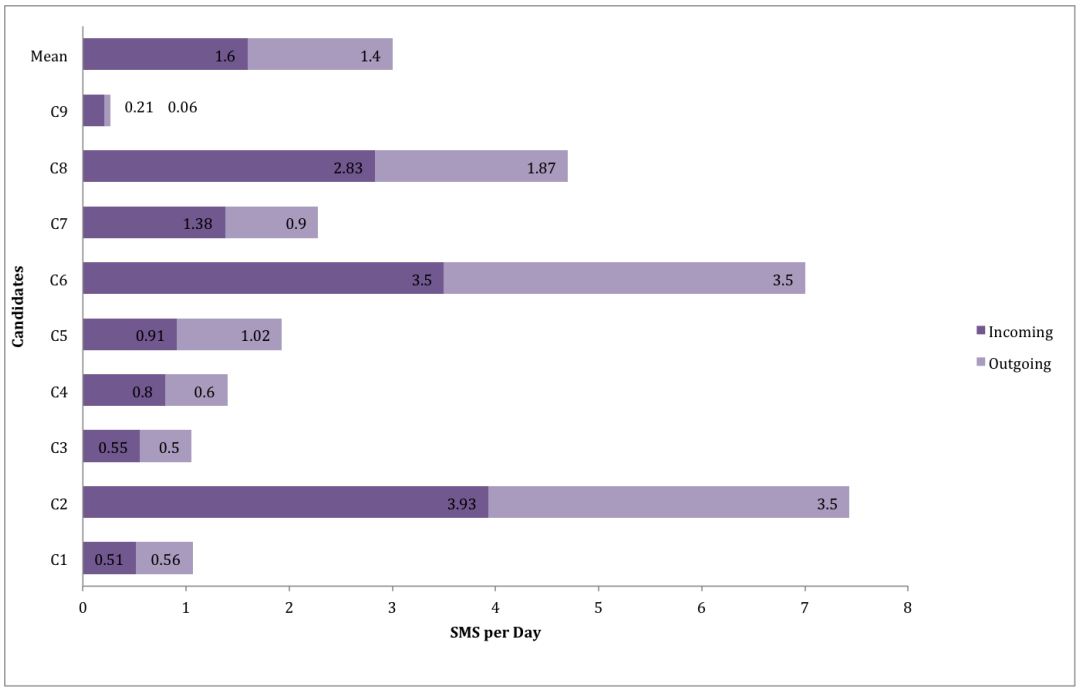

Figure B.4: SMS per Day

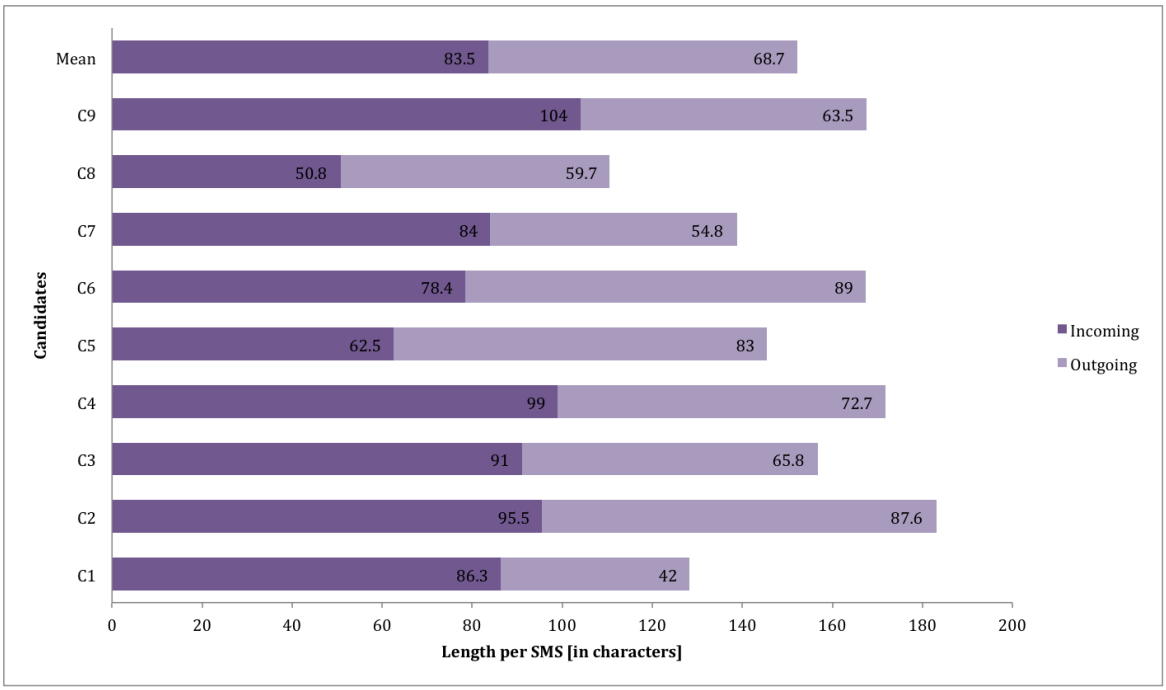

Figure B.5: Length per SMS

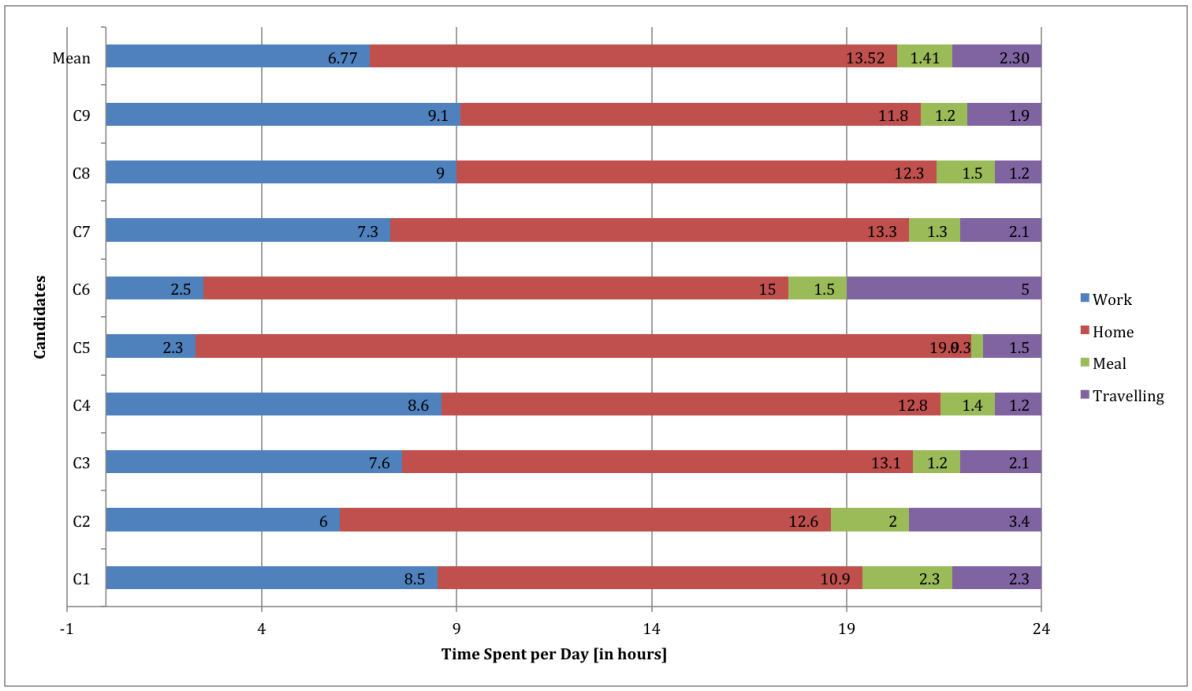

Figure B.6: Daily Time Schedule

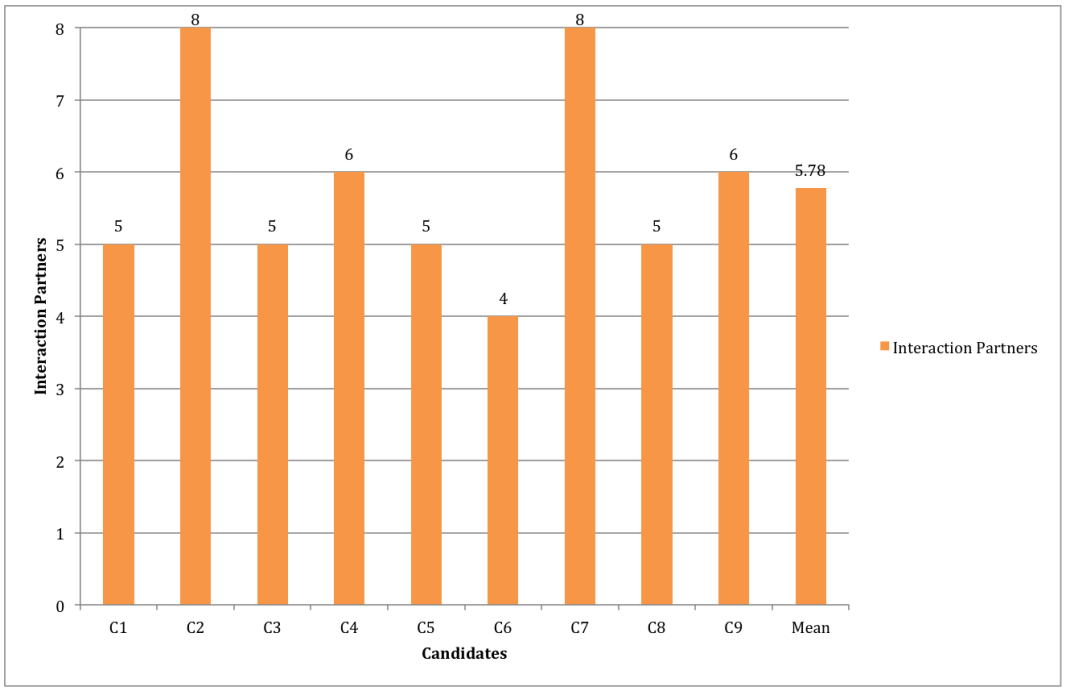

Figure B.7: Collocation Frequency

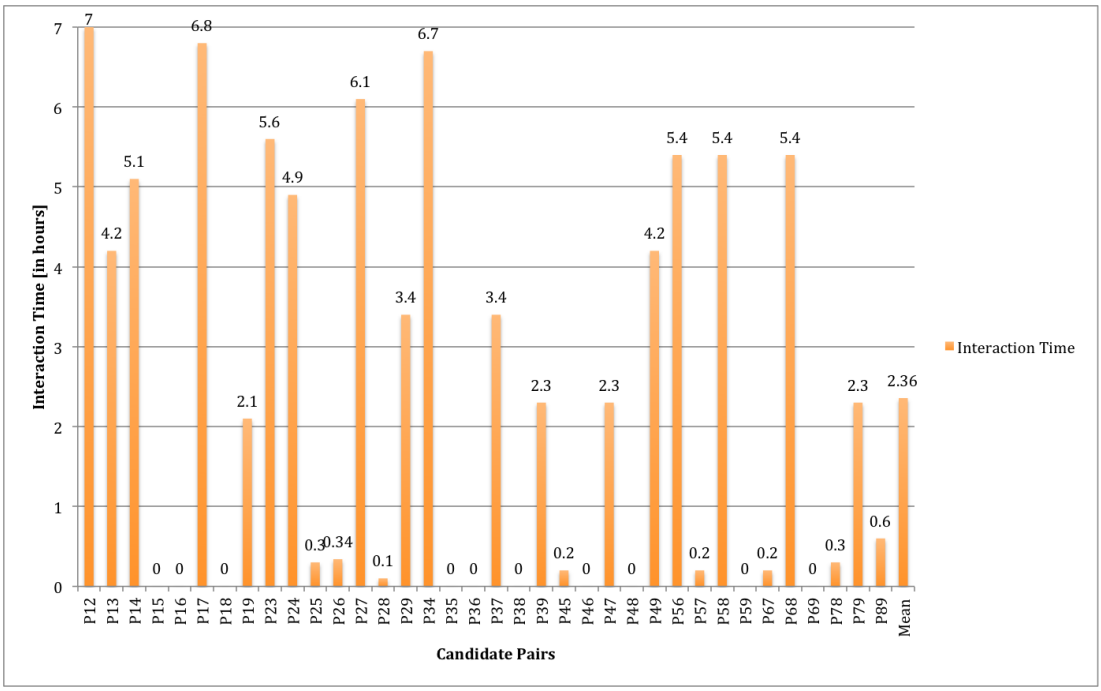

Figure B.8: Collocation Duration of Candidate Pairs

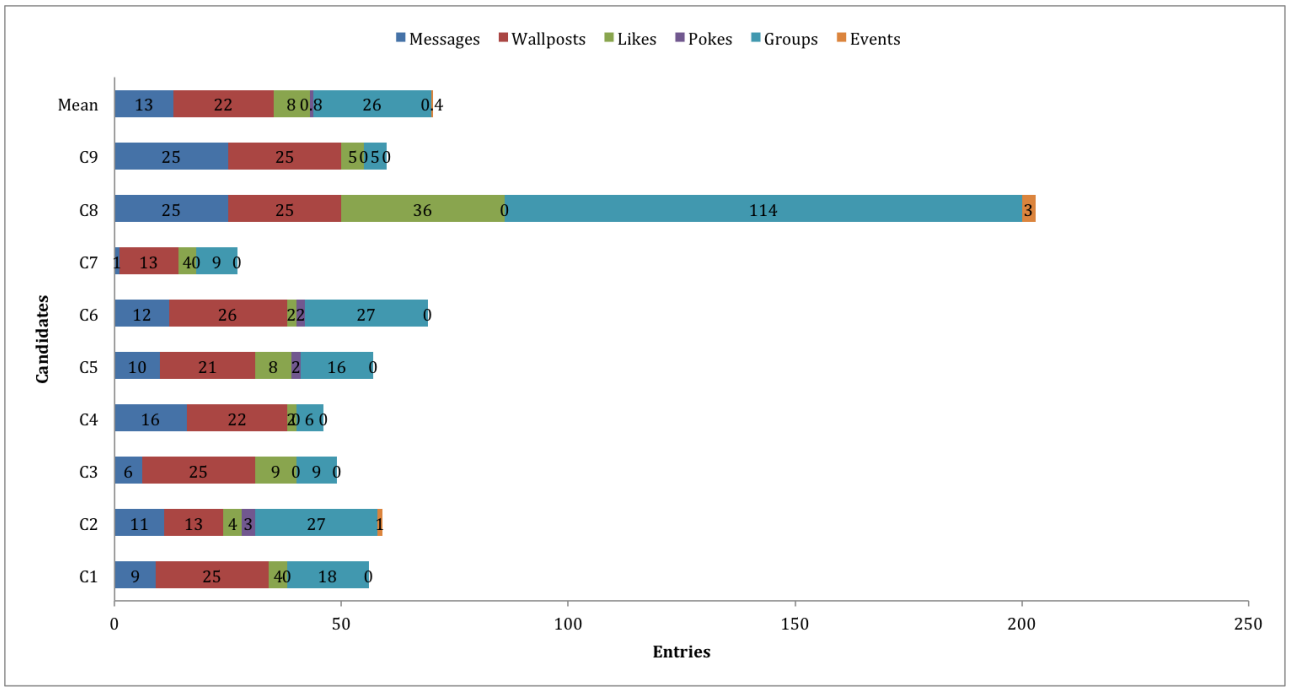

Figure B.9: Facebook Statistics

B.3.3 Data Amount Plot

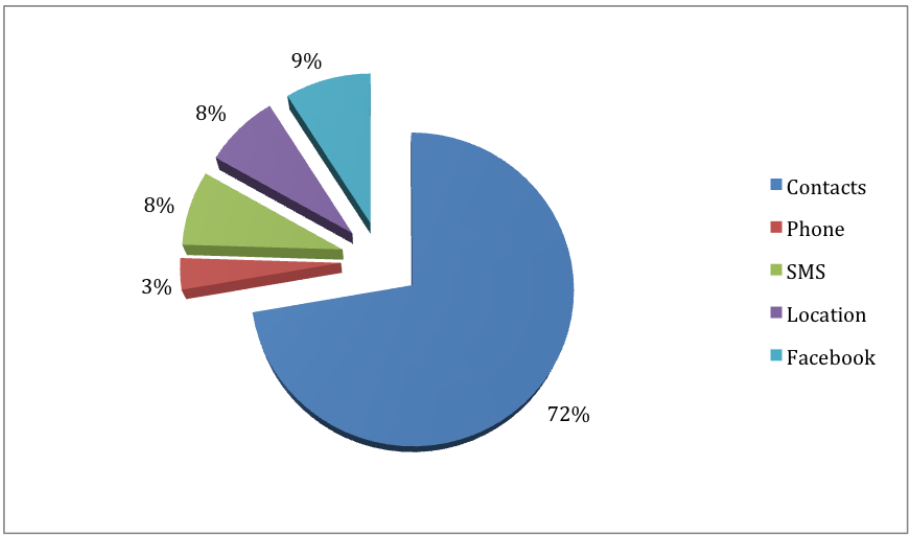

Figure B.10: Data Portions for Extraction Modules

B.3.4 Feedback Plots

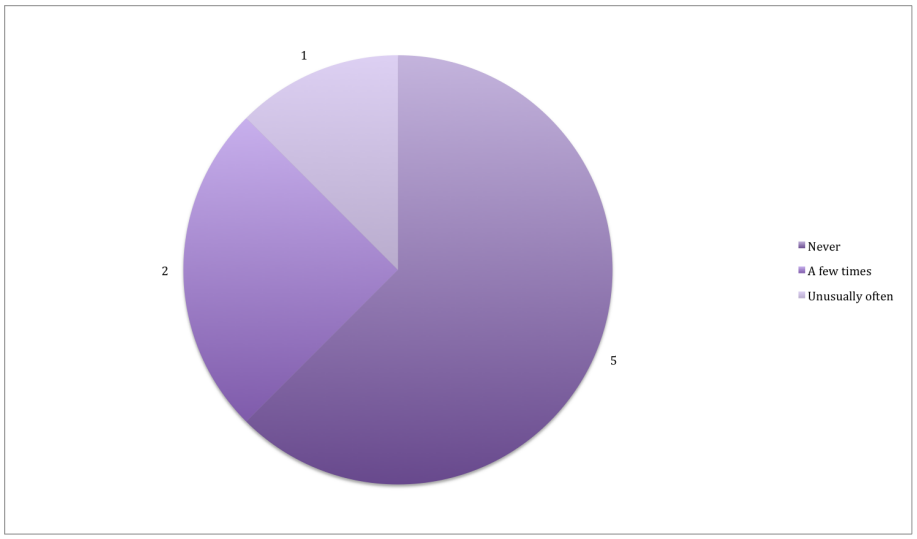

Figure B.11: Application Crash Frequency

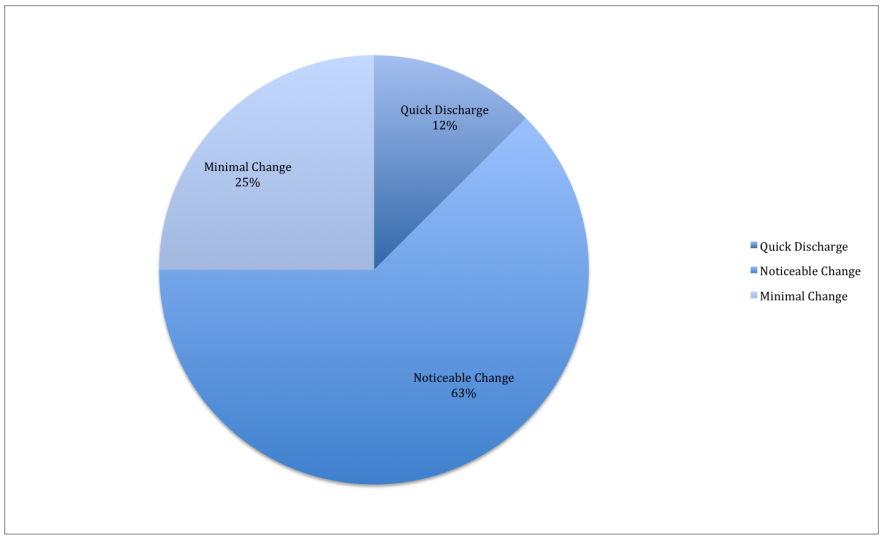

Figure B.12: Battery Consumption

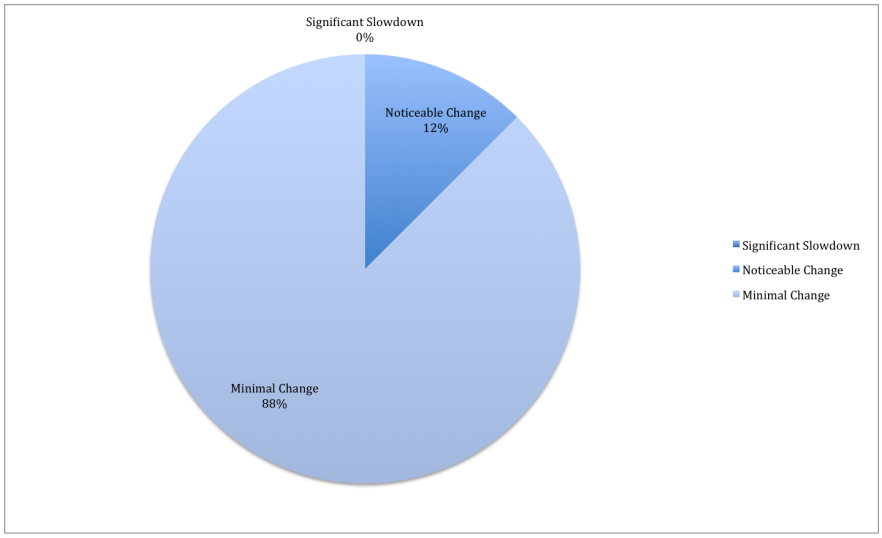

Figure B.13: Operation Speed

Bibliography

[1] Theus Hossmann, Franck Legendre, George Nomikos, and Thrasyvoulos Spyropoulos. Stumbl: Using facebook to collect rich datasets for opportunistic networking research. In *The Fifth IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC 11)*, Lucca, Italy, June 2011. IEEE.

[2] Tuong Huy Nguyen CK Lu Annette Zimmermann Atsuro Sato Hugues J. De La Vergne Roberta Cozza, Carolina Milanesi and Anshul Gupta. Market share analysis: Mobile devices, worldwide, 4q10 and 2010. *Mobile Communications Worldwide, Telecom Equipment Europe, Mobile Devices Worldwide, Telecom and Internet Markets Asia/Pacific*, pages 1–12, February 2011.

[3] Scott Thurm and Yukari Iwatani Kane. Your apps are watching you. *Wall Street Journal*, December 2010.

[4] Nathan Eagle and Alex (Sandy) Pentland. Reality mining: Sensing complex social systems. *Journal Personal and Ubiquitous Computing*, 10, March 2006.