INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Extracting and Relating Relevant Personal Information from Heterogeneous Sources

João Pedro Vieira Guerreiro

Dissertation for the degree of Master of Science in
Information Systems and Computer Engineering

Thesis Committee

Chairman: Doctor Joaquim Armando Pires Jorge
Adviser: Doctor Daniel Jorge Viegas Gonçalves
Members: Doctor Maria Teresa Caeiro Chambel

October 2009

Resumo

A proliferação dos dispositivos pessoais e a sua constante sensibilidade para as nossas interacções, juntamente com o aumento das capacidades de armazenamento, geram uma enorme quantidade de informação. Esta pode ser útil na obtenção de informação relevante para o utilizador quando necessário, em vez de complicar a sua vida com grupos de dados inúteis e não relacionados.

Uma análise ao estado-da-arte mostrou que a maioria das aplicações que tentam usar informação pessoal lidam apenas com um tipo de informação, e os que lidam com mais, não estabelecem quaisquer relações entre estas.

Apresentamos uma plataforma que, num processo iterativo de procura, usa a informação pessoal dos dispositivos do utilizador, juntamente com fontes de informação públicas, para fornecer informação útil e relevante do ponto de vista do utilizador. A informação extraída dos dispositivos, devido ao seu carácter pessoal e confiável, funciona como um filtro de informação de fontes menos confiáveis e estruturadas. Devido às diferentes representações das fontes de informação, definimos uma estrutura comum que permite o inter-relacionamento da informação como um todo coerente, em vez de segmentos separados. Para lidar com a informação não-estruturada, desenvolvemos um módulo de Língua Natural que devolve a informação relevante na representação comum.

Para avaliar a nossa abordagem, apresentamos uma aplicação de exemplo, cuja intenção é obter informação relevante sobre pessoas. Os resultados, analisados em conjunto com os utilizadores, sugerem que, na maioria dos casos, é possível obter informação relevante de várias fontes, quer esta seja esperada ou não, recorrendo a informação pessoal e pública.

Abstract

The proliferation of personal devices and their constant awareness of our interactions, together with an increasing storage capacity, have generated an enormous amount of information that can be useful to help the user obtaining relevant information when needed, instead of complicating his life with unrelated and useless data chunks.

A state-of-the-art analysis showed that most applications trying to make use of personal information only deal with a single information type, and those who deal with more do not establish any relations between them.

We present a framework that uses the personal information on user's devices, together with public online sources, to provide useful and relevant information from the user perspective, in an iterative seeking information process. The information retrieved from the users' personal devices, due to its personal and trustable character, works as a filter to the information retrieved from other less trustable and structured sources. Due to different representations used by the several possible sources, we defined a single structure to allow inter-relating the information as a coherent whole, instead of separate chunks. Also, to deal with non-structured information, we developed a Natural Language module that returns the relevant information on the single representation.

To evaluate our approach, we present an example application which purpose is to obtain relevant information about some person. The results, analyzed together with the users', suggested that, in most cases, it is possible to obtain relevant information from multiple sources, either they were expecting it or not, resorting to personal and public information.

Palavras Chave

Informação pessoal, Extracção de Informação, Informação Pública, Informação Inter-Relacionada, Base de Conhecimento, Processo Iterativo de Pesquisa

Keywords

Personal Information, Information Extraction, Public information, Inter-Related Information, Knowledge Base, Iterative Search Process

Acknowledgements

First of all, my special thanks are addressed to my family. My mother, father and brother always supported me and encouraged me to follow my preferences, doing my best in everything I propose to do. You are the best family I could have. My girlfriend (and the best), Joana, always supported me during the whole process motivating me and giving me the strength needed to proceed and improving my dissertation.

To all my friends, that gave me great days and nights of relax, party and big laughs and conversations. Also, to my friends' football team, 20Comer, thanks for all the pleasure, fun and competition involved on our games, this gave me more strength and joy to my daily life.

To my supervisor, Professor Daniel Gonçalves, special thanks for giving me the opportunity to do this dissertation. Thanks for being always available to answer my doubts, provide new ideas and advise me. Your constant support kept me on track and is a big reason for my motivation and interest on this dissertation and area.

To IST-TagusPark and INESC-ID, in particular to the Visualization and Intelligent MultiModal Interfaces Group, thanks for the support and conditions to the success of the dissertation.

I also thank the members of the juri for the constructive comments that allowed me to improve my dissertation and the approach to the future work.

Again, to my brother, thanks for the motivation and help you gave me to surpass all the obstacles and challenges on this dissertation and on my life. You are my role model.

Contents

List of Figures

List of Tables

1 Introduction

Not so long ago, our personal information, whether paper documents, archives, agendas, notes or interaction registers, like letters (*snail mai*l), was restricted to limited, private and confined environments. Times have changed and the Personal Information Management (PIM) research field has emerged and gain wider relevance in the last few years to face the challenges of an imense universe of digital information. With the advent of the internet and its underlying services, like sharing ones, document repositories, webmail, along with a proliferation of personal electronic devices with increasingly large storing capacities, our personal information is now scattered among several devices in an unrelated, but still connected, information network.

While the great amount of information can be seen as a problem and a challenge at several levels, it can also be faced as an opportunity. Recent technological advances in personal and mobile computing have presented mobile devices as tools that go beyond their initial purposes. It also happened, and still happens, with personal computers that, in a constant evolution, feature more and more applications and utilities resorting to technologies such as Bluetooth and GPS that are likely to be useful for novel applications. Particularly, mobile devices have become all-in-one devices gathering a set of functionalities to accomplish several goals (communication, leisure, productivity,..). The growing mobile device penetration rate in the society has increased the attention over mobile communications and several companies have been putting their efforts towards presenting innovations in mobile technology. Nokia, Apple, RIM and HTC are some examples which have contributed to the 13% share that Smartphones represent in the mobile phone market, according to statistics announced in 2008 by Canalys[1].

[1]Canalys Q3 2008 Statistics at http://www.mobilephonedevelopment.com/archives/708 (last visited in January 6, 2009)

Mobile devices are widely distributed and have become essential tools for the most of us. Moreover, and due to their constant use, they contain information about the users, their habits and daily interactions, as no other person or device. However, these devices do not store all the information the user may need. Rather, each person is likely to possess other devices, like desktop computers or laptops, having information divided (and sometimes replicated) among them. However, the ability to access all personal information from a mobile device, regardless of where it is stored ((Lamming et al., 2000; Flynn et al., 2000; Ahn and Pierce, 2005)), makes accessible all the documents or e-mails shared with their contacts, together with online information, at any time or location due to the almost total coverage of wireless network. Indeed, mobile communications are now able to guarantee constant connection to the network and thus other devices.

There have been some research projects focused on improving the user mobile experience providing them with useful information. However, most of them are related to an intimate group of contacts and the mutual sharing approval of personal or contextual information. Although that it is a possible scenario and may present value to an intimate social network it is hardly scalable to a wider scope as privacy issues arise. What if information about someone not-so-close is required? Or even about an event, or document? It is not acceptable that we wait for somebody to provide us that kind of information, when the information is likely to be at our reach and ours to take.

Our devices are aware of most of our interactions, so they have access to an enormous amount of personal information. Documents, e-mails, calendar, SMSs, phone calls and even presencial interactions are great examples. Also, with the world-wide-web, a lot of new public information sources have arisen. Search engines provide us with an enormous amount of information, people in blogs talk about everything (even their personal life) and recent studies ((Lugano, 2008; Huberman et al., 2009)) reveal that only a small group of a person's group of friends in social networks maintains constant activity with him. Many people add others as friends, only because they are friends of a friend or have similar interests, which means that the accessible information goes even beyond the barrier of our close relations.

The personal information in the users' devices can be used to filter and guide the retrieval of public information. Together, they will help the users to get information about people, events, subjects and places that are personally relevant to them instead of generic, one-size-fits-all, search results. Also, the credibility and personal character of information on user devices, can help filter the information from online sources, resolving the ambiguity inherent to them and presenting results with meaning and interest to the user. However, having multiple information sources with different structures, and some not structured at all, it is essential to find a single representation in order to easily inter-relate the information as a coherent whole, instead of separated chunks. In non-structured cases, it is also necessary to find the relevant parts and resort to natural language to process the information, structuring it as the common representation.

Our framework can back up the user with personal information from his devices and public information from online sources, resorting to an iterative process of seeking for information, using the most credible data to guide the searches on less trustable sources and resolve ambiguities. With a big variety of sources and their different credibility, it is imperative that the user has the idea of which is the most reliable information. Also, with different sources providing information, there is the need to reinforce the data confidence if it is replicated, even if not using the exact same words. For example, the BlogSpot refers "machines" as an interest of a person, and facebook

points "machinery". That mean exactly the same, so it needs to be treated like that, reinforcing the information, instead of dividing it in two different pieces of information.

The framework has to decide which information should be presented to the users, so it has to make two big decisions. First, when does it have all the data needed (or in some cases, enough data) stored and inter-related as a coherent whole, and should stop the iteration. And then, from that data, which is credible and relevant to the users to be presented to them.

This thesis is focused on the challenges related to knowledge extraction, understanding and inter-relation. It can be applied in multiple contexts, either mobile or stationary, since we may need information at anytime, anywhere. Our main contribution is the inner process and not the interface or device where it will be used. However, we believe that its greatest potentialities and values are present in mobile scenarios. Mobile devices are qualified to provide a ubiquitous access to this information, and will certainly occupy great part of the possible applications, working as an entry point.

1.1. Motivation

There is a lot of personal information about us and the people we interact with, on our personal computers and mobile devices. However, this information is spread among the user's information universe and its retrieval is highly dependent on the user's memory and searching abilities, which limits its use. Most of these records are effectively lost. Indeed, all this available data is not used to help the user to obtain relevant information from his own point of view. The available information can provide real knowledge and at the same time be used as a filter to other sources of information more ambiguous and with less credibility, mostly in the Internet "universe of information" (search engines, blogs or social networks). For example, searching Google for John Smith will output several results relating to persons with that name but with no guarantee that information about a specific one is provided. However, if the searcher is aware that John Smith has interest in mobile devices (among other things like rugby and cars) and is professionally linked to the mobile area, this information can help retrieving the correct result. Moreover, if user usually talks to John Smith about mobile technology, it is also reasonable that he desires information relating to that subject.

There are many sources of information that can be explored, however they have different representations and structures. Besides finding a single representation and structure to deal with that, a special attention is needed to process the non-structured sources, where natural language can be of big use. It is essential that the knowledge obtained is managed and shown as an interconnected whole and understandable to the user, instead of isolated chunks of information.

Several information sources are available and the potential useful information is huge, but nowadays the Personal Information is under used, when it could improve the quality and quantity of the retrieved information. Mobile devices, standing always with their owners are the perfect suspects to help them when they need to, working as personal assistants, responding to their searches and promptly answering their doubts. It is impossible for the users to recall all their interactions (e-mails, calls, sms or even personal interactions), calendar activity or information about events,

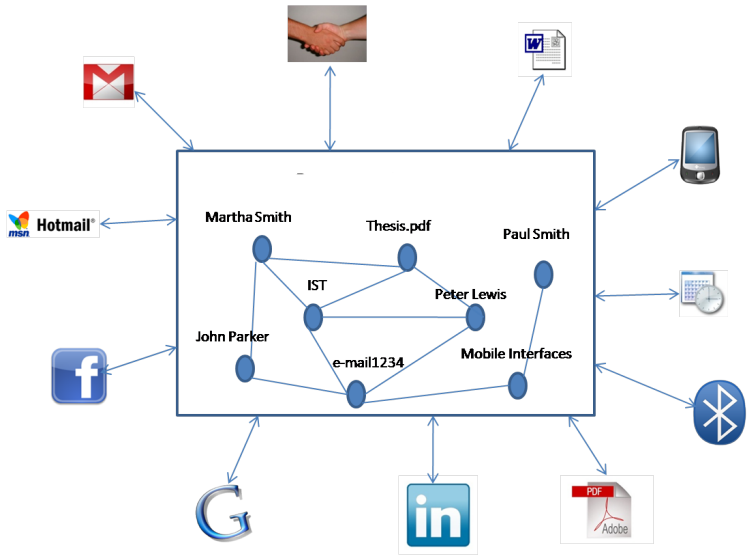

Figure 1.1: Possible information sources and relations in the Knowledge Base

documents or persons. Mobile devices have the capacity and characteristics to do that, working as a memory proxy, with the capability to enrich the provided knowledge by remotely resorting to other devices or even with filtered information from the world-wide-web.

In a social environment it is natural to wonder "I know that person, but where from?" or "I had some things to discuss with Manuel, but what were they?". It is not unusual to wonder about someone or something, and when that happens people often want their doubts promptly answered because that information may not be as useful when they later are with their personal computers. Besides, the time people waste trying to get it manually is priceless and a system that gathers all the information available and really useful to them, automatically, is certainly of big value. With the personal information existent on user devices and public information on online sources, mobile devices are the perfect candidate to be the point of entry of a system that can use all this knowledge to provide the user with synthetic and relevant information from the user point of view.

1.2. Proposed Approach

Our approach focuses on a social environment where people try to get more information about some person, document, event or subject. It is important that the information is collected automatically otherwise people would waste much time with it. To accomplish that, we want resort to the enormous amount of personal information existent in the user devices (mail, SMS, phone calls, documents, calendar, presential interactions,...) and to public information available on the world-wide-web, like social networks, search engines or blogs(figure 1.1).

We use an architecture composed by different modules, which turns it much more organized and extensible, with the advantage of changing only one module if it needs some improvements or modifications. Also, it is a plugin-based architecture, being each plugin related to a different

source of information. This option allows us to add a new information source without changing any part of the existent code. It is only necessary to add the plugin-related code. At this point, we have developed four plugins: Facebook, which retrieves the person's profile information; BlogSpot, extracts the "about me" section in blogs; Wikipedia, gets the most important information on its pages and Scribe, that uses an application made to monitor personal information and indexes it (Gonçalves, 2007). We use Scribe to retrieve information about the users' interactions, more concretely, e-mails (subjects, participants, direction, date and attachments).

Besides featuring plugins, the framework has a Plugin Manager that decides which plugins are useful for each situation. This is based on the plugins announcement of their capabilities, a defined number of main characteristics. At this point we are retrieving only information about people, so we defined interests, work, birth, contacts and interactions as main features, for being the most important ones in our opinion.

Personal information, besides providing real knowledge to the user, is used to filter the information from less credible and more ambiguous sources of information. Using an iterative process, continuously seeking for information, we find new data using the most credible as context. This can also resolve some ambiguities, and gives more relevance to the most frequent, credible and relevant information from the user point of view. The module responsible for this iterative process in our framework is the Coordinator. It has to control the searches and store the information on the Knowledge Base, a semantic network that contains the information retrieved from the plugins, inter-related as a coherent whole. Additionally, and always consulting the Knowledge Base, it has to check which main characteristics are suitable to continue iterating and when should it stop. When it stops, it has to decide which information should be presented to the user.

There are many sources of information that can be explored; however each one of them uses a different representation and structure. It is essential to harmonize the information and the best way to do that is finding a single representation to all the information, extracted from the various sources. With that, we turned possible the insertion of this data on a Knowledge Base, interconnected and consistent, instead of isolated chunks. We can establish relations among different types of information and even reinforce the confidence of replicated information from different sources. Besides finding a single representation and structure to deal with that, a special attention is needed to process the non-structured sources, where natural language can be of big use. We have a Natural Language Module to deal with the non-structured information forwarded by the Coordinator when it detects some. This module processes the information and returns it to the Coordinator using the single representation specified.

With these features, the framework can be useful, for instance, trying to get more information about some person without invade his privacy, resorting only to personal or public information; getting the information needed to a meeting or conference; or find persons, documents and emails related to some subject.

1.3. Contributions

This dissertation shows the possibilities of extracting and managing personal and public information, to help answering the user doubts and searches. The **main contributions** of the work that resulted in this document are:

State of the art analysis A study on current and past researches in obtaining and using personal information, that besides analyzing each approach, points out their advantages and disadvantages. Additionally, the different approaches are compared based on a set of main criteria found relevant.

Heterogeneous information gathering Use the personal information existent in the user's devices to help him in his daily tasks, working as a personal assistant providing information when he needs to. Either it is about a person, document, event or subject; our approach can gather information, automatically, from various sources and present it to the user, with relevant information from his point of view.

Content-filtering of public sources Use personal information to filter information from ambiguous sources, mostly in the Internet. Using search engines, blogs or social networks as sources of information, we gain a lot of information, but not as trustable as that in our devices. Thus, we filter that information to guarantee its relevance to our searches.

Hetereogeneous personal information as an inter-related coherent whole Manage the information from different sources as a coherent and inter-related whole, instead of separate chunks of information. There are several sources of information, each one with a different representation, and it is crucial to find a single structure for all kinds of information from all kind of sources.

Personal Information Framework A framework that uses the personal information existent on user devices, and public online sources to provide relevant information from the user point of view. This framework inter-relate the information and continuously seek for more information on a iterative process of searching.

Example Application An example application that can provide relevant information about some person, on user demand, making use of his personal information and public sources, like blogs and social networks. With the personal information as its basis, the results are useful from the user point of view.

1.4. Publications

The work presented in this dissertation resulted in two original publications accepted in national peer-reviewed scientific conferences. Below we present them:

Guerreiro, J., Guerreiro, T. and Gonçalves, D. (2009), *GeniusPhone - A Personal Companion for Everyday Situations*. In Proceedings of the 17th Portuguese Conference on Computer Graphics. Covilhã, Portugal, October, 2009.

Guerreiro, J., Guerreiro, T. and Gonçalves, D. (2009), *GeniusPhone - Extracting and Relating Relevant Personal Information*. In Proceedings of the 15th Portuguese Conference on Pattern Recognition. Aveiro, Portugal, October, 2009.

1.5. Document Outline

To fully understand the state-of-the-art, in Chapter 2 we make an analysis of the current systems that try to use personal information to improve the users' experience and discuss its benefits and limitations accordingly to some important features. In Chapter 3 we present our approach to overcome the aforementioned limitations and provide a coherent and inter-related personal information space, describing its most important concerns and detailing the architecture. Chapter 4 describes an example application, GeniusPhone, which purpose is finding relevant information about persons. In Chapter 5, we describe the procedure of the system user evaluation, and present its results and discussion. Finally, in Chapter 6, we conclude by pointing the major results of this dissertation, its contributions and limitations, and indicating possible directions for future work in the area.

2 Related Work

Requiring a document, a mail message, or trying to remind where we know someone from is a common task for almost everyone. Current communication technologies enable us to interconnect our personal information spaces and access information everywhere at anytime. Particularly mobile devices, due to their constant use, are becoming users' companions, and the information they have about the user, its habits and daily interactions, can be exploited in multiple ways. In this section, I survey the different applications trying to help the users in their daily tasks and discuss the advantages and disadvantages of each application according to relevant features.

2.1. Systems Description

In this section the applications are divided in three areas based on: getting others information; use of personal information and documents ubiquity, and are described individually. We also show an additional area related to some smart applications using the recent and promising software *Android*.

2.1.1. Gathering other user's information

This sub-section refers to the applications that focus mainly on others' information and context. Their purpose is to know information about the others' profiles or interests, or what they are doing. The first one is used to find people with the same interests or someone they know but do not

remember the name or who introduced them. The second is to get information about their context to know what is the right time to communicate or what is the best communication channel to do it.

ParcTab

The Context-Aware research area has several different contributions across several different contexts. In the mobile context, one of the first and very important projects was developed by Bill N Schilit et al, with the ParcTab (Figure 2.1) (Schilit et al., 1993; Schilit et al., 1994). ParcTab is a PDA that communicates via an InfraRed cellular network. With a series of spread transceivers, the devices are easily on other ParcTab's range and stationary equipment.

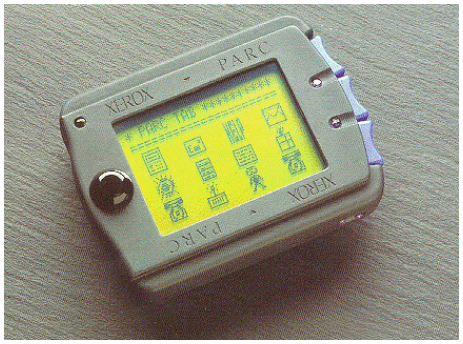

Figure 2.1: The ParcTab Hardware Device

The base of its creation is to examine and react to a change of context. It uses locations and other information related to the context, like the people who are in a room. Each ParcTab can track the other's location, provide them its location, and interact with them or other available equipments. In this way, it is possible for the employees of a firm: to know where to find others if they need; access a large dictionary, what would be difficult due the size of the tab; be used like a remote control (to lights, temperature..); control window migration, users can select one display to migrate windows of his applications to it. This is useful, for example to migrate the information to other employee or to a workstation.

With ParcTab applications, it is possible to make distance comparisons between devices. With this, if an employee needs help, he can find the nearest employee to help him; he can print, by default, to the nearest printer and find where it is; etc. The insights launched with ParcTab made the project very well known and used by several other researchers as a base in their mobile context-aware applications.

The ParcTab, by itself, does not handle with the other users context information besides the location. It can find where something is and interact with it, but not if it is available. Other disadvantage is that the evolving environment has to be equipped with transceivers, so the Tabs can be captured. Simultaneously the system presented scalability issues as the InfraRed system does not behave properly when the network is crowded with several ParcTabs.

Live AddressBook

The Live AddressBook (Milewski and Smith, 2000) application helps users getting context information about the other subscribers. It is an application based on the web, which can be accessed from any device with a browser that connects to the live address book server. Basically this application provides information about the users availability with the states Available, Urgent Only, Away and Do Not Disturb; Location, like office, home, mobile or other chosen by the user; and

optional message written by the user, for example "working". This helps the users to find out how and if they should communicate with others.

All of this context information is selected/written by the user manually, which is a big disadvantage due to the time he loses with it and the big probability to forget to change it.

Context-Phonebook

The Context Phonebook (Schmidt et al., 2001) helps users getting context information about their contacts, with the intention to know if it is the right moment to make a call.

The Application uses WAP and is provided in the network. When the phonebook is used, the WAP-Application is invoked. The context information used is the connection state, availability and location of the contacts. Instead of calling and knowing if the phone rang, sent to answering machine, or get unreachable or busy line responses, the application provides the user connection state information. To show the user availability, the user chooses one state that will appear to others. Those states are shown like a traffic light and correspond to: Red, Busy only accepts emergency calls; Yellow, busy but accepts important calls; Green, free to accept calls. The location shows if the user is at work, home or mobile.

Awarenex

The ConNexus is a prototype that integrates awareness information and communication tools, to provide the user more knowledge about the "buddy" context, and facilitate their interaction. The project ConNexus runs on a desktop computer, and the Awarenex (Tang et al., 2001) is the extension to wireless mobile devices like cell phones or Palms. In the Contacts list of an Awarenex user appears his contacts location (office, mobile, home...), activity indicators (idle time, logged of time, the most important appointment...). With that information, the user can guess if the contact is available for contact. The contacts locator shows the information of every device that is active, with preponderance to the most recently active one (figure 2.2), for example in the office, computer and phone. When you're trying to reach the contact, it automatically suggests the better way to do that.

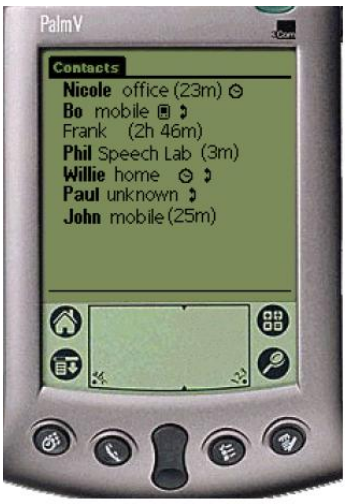

Figure 2.2: Contact List shows a list of users for each most recent activity

Awarenex has a particular characteristic that is the speech interface. When you call someone and identify him as an Awarenex user you can begin a dialog. For example:

Awarenex: *Manel is not answering the phone. Would you like to leave voice-mail, page him, or check status?*

Caller: *Check his status.*

Awarenex: *Manel has an appointment at 1:30pm to 6:00pm Eastern Time, "working at home". His computer at home is active. What next?*

Caller: *Call his home.*

Awarenex: *Calling Manel at home.*

Like the Live Contacts, the location information is very limited in this application, showing only home, mobile or work. It is also possible to indicate the best way to communicate. It has the same disadvantage that is only monitoring the presence in the devices and not the availability.

iCAMS

The iCAMS (Nakanishi et al., 2002) is a web-based application that helps users getting context information about their contacts, to know what the best channel to communicate with them is. It uses the contacts location and schedule information, but the last one is not visible to the user for matters of privacy. The location has the precision of approximately 100 meters and is provided by the location-detection service of the PHS (Personal Handyphone System). The application sorts the contacts by their proximity according to the last known position, and has icons to show if they are moving and in which direction. With this information the user has a suggestion of how to communicate, but can choose another channel of communication if he knows more about the contact context. The user can also make rules for some status, like "meeting", sorting the channel of communication that he prefers.

One limitation of this application is that it only uses the location and schedule information to provide context to the users. The location precision is enough when we talk about a large universe, but in a small one like a university it is not. The schedule information being invisible to the users and used only to sort the best channel of communication is a great advantage in the big question of privacy.

Live Contacts

The Live Contacts (ter Hofte et al., 2004) helps getting context information about the user contacts. Using this application it is possible to know who the user should call, at the right time via the right communication channel. The context information in Live Contacts is a combination of: The Messenger status (online/offline/...); the calendar information (type of appointment) and the last known location (Home, Mobile, Work).

It represents the availability preferences using colors (red, orange, green) to indicate the best way to communicate. The user can call immediately to a contact or ask to be reminded to do it when, for example, an appointment is over.

The location information is very limited in this application. Only home, mobile or work sometimes may not be enough. However, monitoring these different locations it is possible to indicate the best way to communicate with the contacts. One disadvantage in Live Contacts is that it only monitors the presence in the devices and not the availability.

Lilsys

After the Awarenex project, James Bo Begole et al (Begole et al., 2004) made an advance to the Lilsys prototype, which focus on the difference between presence and availability. The availability is closer to the person presence on the device, than to the device status. The IM(instant messaging) status can be online, but idle for 1 hour, and it doesn't mean that the person is not there. He can be reading, but the idle time suggests he is not there. Also, sometimes when a user is using a device it may occur that he is very busy and not receptive to be interrupted. Lilsys inherits the device and calendar information from Awarenex and adds sound, door (open/closed), motion and phone sensors (figure 2.3). It's possible to turn of the device when the user does not want to share his context with others.

Figure 2.3: Lilsys sensor and data acquisition module

Like the Live Contacts and Awarenex the location information is limited, but it is possible to indicate the best way to communicate. One big advantage is that the user status can change not only when he uses the computer, or other device, but when he enters the room, or close the door. It is the availability instead of presence. However, the off-the-shelf sensors, with an awkward appearance and audible clicks, are a disadvantage to take in count.

ContextPhone

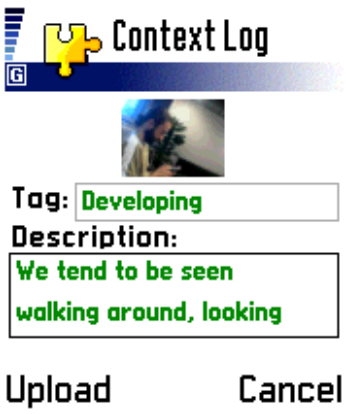

Figure 2.4: Context Media

Mika Raento et al. (Raento et al., 2005) stood out in the context-aware area with the appearance of the ContextPhone. It appears trying to fill the gap between the operating systems functionality and the features that application developers need. The authors followed a human-centered research strategy, and have done several iterations until the last version. They implemented three innovative applications: ContextMedia, ContextLogger and ContextContacts.

With ContextMedia, after capturing the media, it is possible to immediately store and share it (photos, sound, text, video). Besides, it is possible to annotate automatically a description of the situation it was taken in, providing additional information (Figure 2.4). It includes features such as location and nearby Bluetooth devices.

The ContextLogger is an application for recording mobility data. Its goal is to give researchers a way to get interesting data unobtrusively (and can be used by the users too). It has sensors and

applications to alert for context changes, and write that in a local file. It has been used by the MIT Media Lab's Reality Mining with their students to measure the social relationships evolution and strength to model social-network dynamics. The logs are not shown in a friendly way to the user, so this is an application almost exclusive to researchers and people used to work with this kind of language.

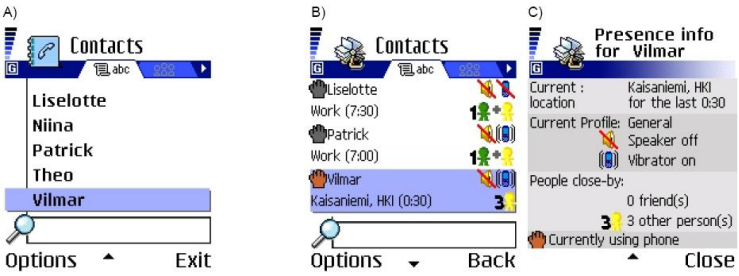

Figure 2.5: ContextContacts sequence example

The ContextContacts (Raento et al., 2005; Oulasvirta et al., 2005) lets users automatically communicate, represent and exchange context information among each other. With this information sharing, the person who is trying to make a call can have, in his contact list, information about the other person context. The context is represented by features like the current location (and how long is there), the current profile, Speaker and Vibrator state, people near him, and the time for the last phone use. Attending to the times that mobile calls fail, if the user knows about the others context it can save time and money (Figure 2.5).

Giving the exact location instead of only work, mobile or home is an advantage in the competition with similar applications. The disadvantage is that the ContextContacts does not give any information about the other devices, like the computer or other mobile devices owned by the contacts. He may not be with the ContextPhone, but on his personal computer, so the best way to communicate should be an email or Instant Messaging.

"WhozThat?"

The system "WhozThat?" (Beach et al., 2008) integrates the social networks with the mobile devices, which is called mobile social networking (MoSoNet). It tries to answer the usual question "Who's that?". The system shares social network IDs, so people can get information about someone. With these IDs people can request the interests and information of others in online social network profiles like FaceBook or MySpace. With this information, people can more easily initiate a conversation to someone they found interesting or someone they know but do not remember the name or who introduced them.

It is possible to build context-aware applications with this capability. An example the authors give is a context aware music player. It captures the profiles of the users in the bar and collects their music interests trying to find a playlist that pleases everyone.

Having access to online social network profiles, turns possible to analyze the social networks and find mutual friends or groups. Has the disadvantage that people sometimes can feel spied by the

other, but have the option to turn their profiles only visible to friends. The problem is that the application would lose its purpose if only visible to friends.

Contacts Application

This project (Jung et al., 2008) creates a new mobile contacts application based on a few design drivers. Among other features, they found important to efficiently access contacts, especially the important ones, differentiating them. They think in a contact as a repository of personal information, which should be used to add more personal or contextual information (photos, birthday, communication history, etc) (Figure 2.6). The basic structure of their prototype is divided in 4 areas: Names and Groups hosts the list of all contacts and groups with the additional feature of Top Contacts, which can be manually or automatically selected by communication frequency; Smart Groups shows automatically created groups with common attributes like Popular(top communicators), Birthday(recent birthdays) and Shiny New(contacts recently added); About Me allows the user to define his identity and store personally sensitive information; Directory Search allows the user to seek for contact information in external databases, such as Yellow Pages.

Figure 2.6: Contact viewer

It uses the user's personal information to automatically select the Top contacts, with the advantage of adding new smart groups with the recently added and recent birthday contacts. With the possibility of adding context (communication history, contact adding date or last communication) it is easier to recall, for example, who a particular contact is. The possibility of searching in sources like yellow pages for additional information is also a big advantage. It does not have information or relation to other devices owned by the user. Also, the context/information related to the contact could be bigger.

Friendlee

Friendlee(Ankolekar et al., 2009) is a mobile social networking application for social relationships (Figure 2.7). It uses many of the contacts context awareness indicators, such as the current location; time spent there; status (available, busy, etc) and also highlights those who recently changed. However, this is the biggest similarity to the other applications described in this section. Friendlee diverge by being a real network application, turning possible the browsing of, not only our friends but also their friends. Comparing to other social networks like facebook or linkedIn, Friendlee innovates by analyzing the user's call and messaging activity to verify which his closest social contacts are. It is based on phone conversation frequency, recency and duration to assign a weight that determines how close that person is to the user. Separating the closest persons from the others, it is created an intimate social network. The user can easily access to the persons on this network and share his context. This sharing is only possible with mutual consent. Having the closest friends properly identified, it is possible to browse their connections, since probably many will be known to the user. It is also possible to get implicit recommendations for

businesses by browsing the closest friends (e.g. the dentist their friend likes to go to).

Friendlee has the great advantage of using the user's personal information to assume which are his closest contacts in the social network. With that it can easily configure different accesses to context information. A good example is the different granularities of the location information (country, city or GPS-based street address). However, not having any information or reference to other devices owned by the user is a disadvantage.

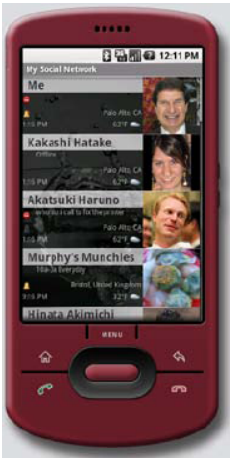

Figure 2.7: Friendlee screen with the list of contacts

2.1.2. Use of Personal Information on User Devices

The Use of Personal Information on User Devices sub-section is about the applications that use the user own personal or contextual information. This information can be used to change the phone settings, to set reminders according to a specific situation or helping the user to remember about something.

"Forget-Me-Not"

One of the most interesting applications that uses the ParcTab is "Forget-Me-Not", by Mik Lamming and Mike Flynn (Lamming and Flynn, 1994). The "Forget-me-Not" is a support for the human memory. It can be used to find a document, remember somebody's name or how to use a program or a piece of machinery. It is difficult to support all the information (all the documents shared, etc), because there is a lot of data, and the space for it is not enough. So, the alternative to easily remember it, is recording the user's context of the event. Things like the location of one meeting, who was there, what happened there or what happened before or after, are added into a personal biography like a log. This is called the memory prosthesis (Figure 2.8) (Eldridge et al., 1993; Eldridge et al., 1994; Lamming et al., 1994).

Figure 2.8: Using a prosthetic episodic memory device

Some examples of what "Forget-Me-Not" can do are: The user encountered Pablo Aimar in the

Luz Stadium, when he was printing one document that sent to Reyes, before going to Colombo. If the user does not remember the document's name, but can only recall that he has sent it before going to Colombo, maybe the rest of context makes him remember; Record which programs are running on a workstation. The main focus goes to e-mail transactions (Incoming and outgoing e-mails); Record File Exchanging and printing. When, where and from who documents where received/sent; Record outgoing Telephone calls.

Forget-Me-Not can determine information about the user's context to help him in a posterior situation, but the sharing of documents between users is not explored.

CybreMinder

CybreMinder (Dey and Abowd, 2000) is a system for mobile phones that provides an automated reminder based on context information. It uses the Context Toolkit (Salber et al., 1999) which aids in the building of context-aware applications. Instead of using post-its or to-do lists, the CybreMinder reminds the user to do something only when he wants to. These reminders are based on situations like time, location, co-presence or number of people in the same location, the user schedule, and contacts status.

To create the reminder, the user can choose the reminder receiver, subject, priority, expiration date and the message associated (figure 2.9), together with the context conditions for the reminder delivery.

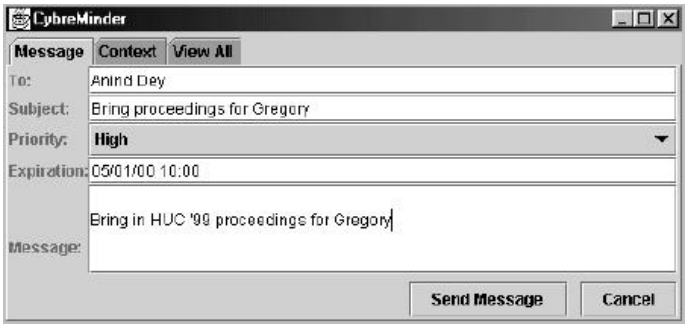

Figure 2.9: Reminder creation tool

According to other reminder applications, the CybreMinder has the big advantage of considering a big variety of context information. Other systems only consider the time and/or location.

Social Net

Social Net (Terry et al., 2002) is a very interesting application that uses RF-communication (Bluetooth) with an original strategy of interest-matching. Like other applications, it broadcasts a user's presence and detect others nearby, but also records the time and duration of encounters with the other users. With this, it tries to find patterns of physical proximity between people over time, to assume that they may have interests in common.

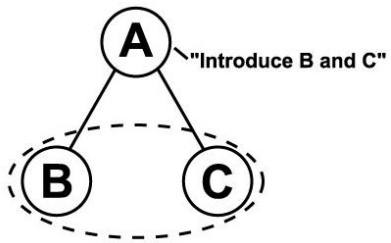

Figure 2.10: Introducing in Social Net

Each user has a friends list, and those who are not friends but are interest-matched are marked as unknown. When two friends encounter each other, the application shares the unknown lists. If one of them has a friend that is on the other's unknown list, he is suggested to introduce them (figure 2.10).

The biggest advantage is about to the user privacy. There is no need to share profiles, and the intermediation is in charge of a mutual friend. The decentralized design of the application also contributed to that. However it has the disadvantage that only considers the spaces that user's co-habit, which can be tricky about their interests. Another problem is that people after being introduced, usually do not add the other as friends, so other friend may be suggested to introduce them again.

SenSay

In the context-aware applications, sensors are an important and very useful tool to help determine and react to changes of context. Daniel Siewiorek et al explore it in the SenSay Project (Siewiorek et al., 2003). SenSay is a Context-Aware mobile phone, that adapts its behavior according to the user state and what surrounds him. It uses accelerometers, light and microphone sensors and, at the same time, the current context of the phone user. The calendars, the address books and task lists are important to find out if the user, for example, is having an important meeting and does not want to be interrupted.

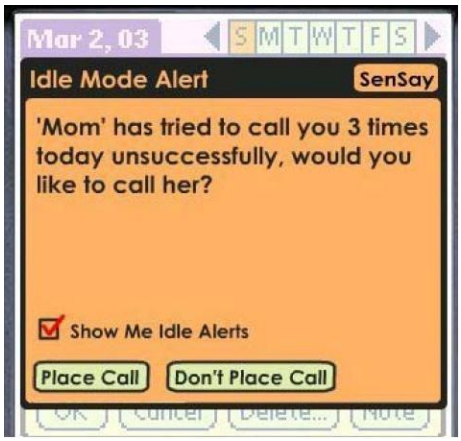

Figure 2.11: SenSay user Interface displaying a call suggestion

In SenSay there are four possible states: Uninterruptable, Idle, Active and Normal. These states depend on a combination (with a weight) of the sensory data, the user information and previous states and data. These states influence things like the ringer or vibration. It can provide the ability to communicate the urgency of the calls, make call suggestions (Figure 2.11), or give the feedback of the SenSay user state.

The big advantage is that it is possible to automatically manage the profile settings considering the user context, together with the suggestions made. The problem is that probably the user loses some important calls, because the callers do not know the user's context when they make the decision to call. The considerably big sensor box in the hip of the user is also negative.

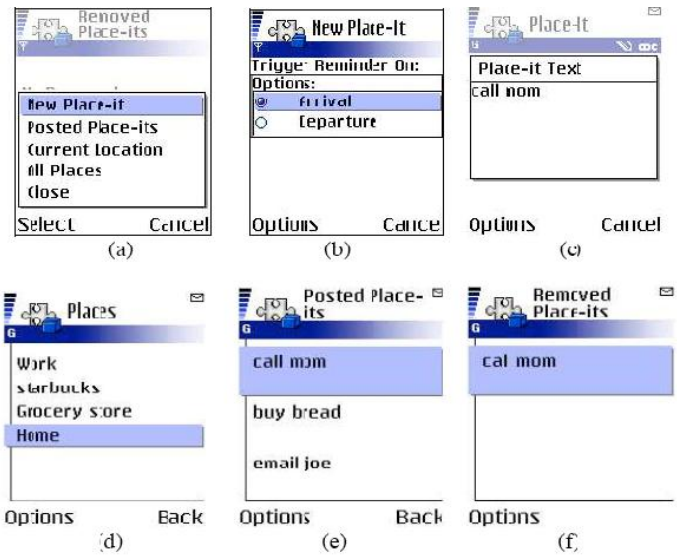

Figure 2.12: **(a)** Creating a new Place-It; **(b)** Setting the trigger to the arrival; **(c)** Typing the text; **(d)** Posting the note to 'Home'; **(e)** Showing all posted Place-Its; **(f)** Reminder triggered when she arrives

Place-Its

Place-Its (Sohn et al., 2005) is an application for mobile phones that provide an automated reminder based on location. Instead of using post-its or to-do lists, the Place-Its reminds the user to do something only when he wants to. These reminders are based on a place, like home, office or others. For example, when I arrive home, I need to call my mother (figure 2.12). When the user is at one place, it is possible to mark that location as a known place to posterior association with a reminder.

This application is based on three steps: the trigger, text and place. The trigger identifies if the reminder should be activated on the arrival or departure of one place. The text is the message associated with the reminder. It is possible, at any time, to view the list of current Place-Its.

Considering only the location, is for the authors an advantage when compared with conventional reminders that only consider one issue, the time. Other very similar applications like ComMotion (Marmasse and Schmandt, 2000), took the option of considering the time too.

HiCon

With the idea that in the future the world and persons will be covered with lots of sensors and little devices, HiCon (Cho et al., 2008) is a framework for "hierarchical context monitoring and composition that supports advanced context-aware services requiring scalable monitoring and composition of dynamic contexts". The hierarchy is divided in three levels: PocketMon, a personal context monitoring platform running on mobile devices, such as location, temperature, actions or emotional status; HiperMon, a high-performance system for massive context data processing, such as information about populations, traffic situation or weather; and EGI, for wide-area context data

delivery, such as global weather or worldwide hot news.

Various compositions can be achieved with this framework. It can involve a single level (horizontal composition) or multiple contexts (vertical composition). An example of a Horizontal composition is the emotional status that can be influenced from heart rate, voice tone, body temperature, and so on. An example for Vertical compositions is a personal healthcare service that helps a patient to find the most appropriate hospital. It involves the physical status and the traffic conditions or medical staff availability.

Some applications were used in a PocketMon. SympaThings controls nearby smart objects to adapt to a person's emotional contexts, and Running Bomber is a game to give additional fun to runners that use accelerometers in their wrist and pass the bomb to others by shaking the hands. An application developed in HiperMon was the UbiCab. It monitors patterns of people and taxies and dispatches available taxis to potential passengers.

The bigger advantage is that HiCon can give context information at different levels and a combination of those levels. However, it seems that the different types of information are not related, being treated as different chunks of information.

TapGlance

TapGlance (Robbins et al., 2008) is a design proposal for supporting common Personal Information Management on a mobile device. The author tries to provide equal accessibility to different kinds of actions and information, using a search TopBar and nine access buttons (emails, date/time, tasks, calendar and some person driven). Given the degree of attention given by the user, TapGlance presents the information at various levels of abstraction. When the user pulls the phone out of their pocket the time and date tiles have the leading role, but if he wants to get more information about some particular tile he just has to press-and-hold in the respective number. The different tile sizes can also be adjusted, giving more dominance to the most important in that particular occasion(figure 2.13).

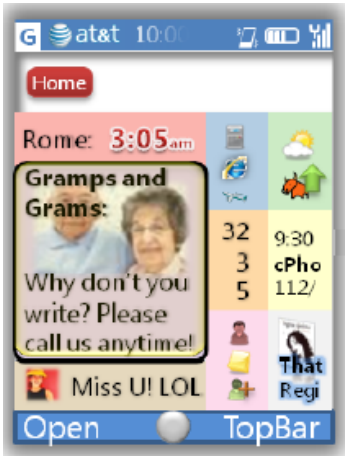

Figure 2.13: The "Person 1" tile is shown enlarged because it has urgent information in it

TapGlance has the great advantage of proposing a unified interface for accessing many different types of personal information, removing the idea of different applications for different sources of personal information. The biggest disadvantage is that the different kinds of personal information seem to be isolated from each other, having no relations established among them.

Associative PDA 2.0

The Associative PDA 2.0 (Falke, 2008) goal is to automatically relate different personal information features. It is based on the Associative PDA [Diehl06], which results were above the expected due to the artificial personal information the user's had to simulate. To avoid this, Associative PDA 2.0 handles only with note-taking (figure 2.14). When a note is created it automatically gathers its time/date, place and nearby people. It uses WiFi for location sensing and Bluetooth to identify the nearby persons.

Figure 2.14: Paper prototype of the note taking screen

Like "Forget-Me-Not", it has the advantage of relating something that happened with its context, which can help to recall, in this case, in which circumstances the note was taken and what it really means. Unfortunately, it does not relate this information with other features of personal information.

2.1.3. Documents ubiquity

The Documents Ubiquity area refers to the idea of accessing information, mainly documents, anytime and anywhere.

Satchel

The seed of this project was the "Forget-Me-Not", which could record the events of user's context, like printing or transferring documents. So, the Satchel Project (Lamming et al., 2000; Flynn et al., 2000) was built to take advantage of the user's context to improve the user interface for printing and transferring documents. It provides easy finding and accessing document services that has access to the user area. When the document location is found, if it is in front of a printer (Satchel enabled) the Satchel presents a printing service and the same with a scanner, or another person with Satchel to transfer documents. It uses document tokens, which are small and can be transmitted quickly and cheaply over a wireless channel. There is no need to remember to send a document after something because the transaction can be started immediately and closure is achieved.

A good advantage is that Satchel architecture is based on the web, so it was possible to provide almost world-wide coverage for services like faxing or e-mailing. An example: The authors printed several times from their UK laboratory to the Xerox research center in New York. One disadvantage is that the Satchel architecture presumes that the devices are turned on and possess fixed addresses, what usually is not that simple.

SEREFE

SEREndipitous File Exchange (Ahn and Pierce, 2005) was built on top of an Instant Messaging system, and has the goal to provide access to documents anywhere at any time, without having to anticipate what documents and at what time/situation they will need to share it. SEREFE allows users to use any of their devices to share information with others or to transfer between their own devices, using different channels to do that.

Figure 2.15: The inbox, incoming file, choose method, and choose device screens for receiving a file

With SEREFE it is possible to send files, after a search to find out which devices had the document to be sent; receive files even if SEREFE is turned off, the user receives an instant message alerting he received a message. The user chooses the device that he wants to receive the document and via the right channel (figure 2.15) requesting files; deferring message handling, if the user wants to finish the document before sending; creating reminders, when the device that has the document is turned off; send files to non-SEREFE users and sending files between personal devices.

This system has the advantage of facilitating the real-time transfer of information when the documents are currently accessible and at the same time help users when the files are inaccessible. The fact that it can easily transfer documents from one device to another is also very interesting.

2.1.4. Android and Some Smart Applications

Android[1] is "a software stack for mobile devices that includes an operating system, middleware and key applications" developed by Google. It stands out for the simplicity and easy way for building new applications providing the tools and APIs necessary for it. It has some core applications such as maps, calendar or contacts, and developers have full access to the same framework APIs used in this applications. Any application publishing its capabilities for being used by other applications is a great advantage.

Some applications emerged for the Android Developer Challenge, and more will certainly follow. In the finalists there are some interesting:

LifeAware[2] application takes special attention to the surroundings and the location of the user's

[1] http://code.google.com/android/what-is-android.html (last visited in January 6, 2009)
[2] http://www.lifeaware.net/features.php (last visited in January 6, 2009)

friends and family. It provides the last known location of his contacts (figure 2.16), and it is possible to make some setup notifications based on someone arriving or leaving certain location. For example, when the sons leave school the user receives a notification. It is also possible to tag some locations and share them with friends.

PhoneBook 2.0[3] is an address book that puts together instant messaging, social networks and life-streaming features. It displays context information about the users' contacts, such as location, presence and status updates. It can also form groups of contacts, such as the most/least used contacts, nearby contacts or emergency contacts.

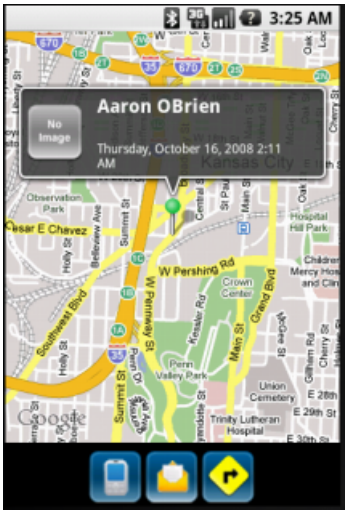

Figure 2.16: LifeAware example

Locale[4] is an application that changes the user settings according to some user context conditions, like location or battery level. It allows the user to create situations that within certain conditions change some of the phone settings. For example, the "At Work" will be on, when the location condition is "Avenida Cavaco Silva" and trigger the user ringer setting to vibrate.

Teradesk e-storage[5] "is a virtual file storage and remote file access tool", which stores the files in a server, allowing transfers without data loss. The user can preview and visualize the item before downloading it. He can share files between Desktop computers and cell phones, and has integration with GoogleDocs, so it is possible to read and edit documents.

2.2. Overall Discussion

Previously in this document some relevant applications were described, accordingly to some features that are useful to define their strengths and weaknesses.

In this section, we compare these applications according to those features to get a global view of the best practices and limitations in the current state of this survey's area. Figure 2.1 presents the overall comparison.

Online Information

The feature Online Information refers to the immense information repository that is the Web. Very few applications make use of the biggest source of information. Live Contacts and LilSys collect the contacts' Instant Messaging status, which is relevant information to make assumptions of their current context and "WhozThat?" extracts information of social network profiles, either to find out if some persons have interests in common, or to recall about the identity of someone that

[3]http://www.voxmobili.com/phonebook20/ (last visited in January 6, 2009)
[4]http://www.androidlocale.com/ (last visited in January 6, 2009)
[5]http://www.teradesk.com (last visited in January 6, 2009)

looks familiar to the user but he cannot remember who he is or who introduced them. The contacts application has a directory search, which allows the user to seek for contacts information in external databases, such as Yellow Pages. The online information provided by these applications can be very useful, but there is no doubt that there is a big universe of sources of information to explore in this matter.

Multi-Devices Information

Mobile devices' ubiquity makes them a personal device that accompanies the user all the time and although their increasing storage capabilities, it is unfeasible to store all the information required. That is why it is important to withdraw information like documents from other devices, e.g., our desktop computer, so we can have access to them anytime and anywhere. In Satchel and SEREFE there is the possibility to find in which device the documents are, and with that, have access to them and transfer to the user's cell phone or to other person's specified device. In Satchel there are services provided according to the situation. For example, when the user is in front of a printer, a document printing service is offered. Although the possibility of getting the documents from other devices, it is not possible to get, for example, the documents shared with a certain person. Awarenex and Lilsys do not get information that is in other devices, but offers information about the other devices. Other devices being on/off, or the idle time, can be a good indicator of the context of the user to help in the decision of calling or not.

Contacts Context Information

When deciding either to call or not, with the preoccupation of not interrupting others, their context is extremely important. There are lots of ways to get context information. Those that only base their decision on one feature (most projects only consider the user's location) are rated low in this parameter. Those who consider some more context than the location are classified as medium. Context-Phonebook reveals the contacts' connection state and Live Adressbook their status. Both of them show the contacts availability and iCAMS and Live Contacts present schedule information. Live Contacts also shows the Instant Messaging status. The applications classified with high have a bigger set of context information types. ContextPhone and Friendlee besides providing location have information about the user current profile, his activity on the phone and the persons that surround him. Awarenex and LilSys have some features provided by the ones characterized by Medium and also the information about the other devices as discussed in the last parameter. Lilsys also features sensors that help to understand the user's context, such as sound and motion sensors. All of these features can be useful to determine our contacts' context and, the more useful features we get, the better.

User Personal Information

The personal user information is useful when he wants to change some settings in his profile, set some action based on his own information or just turn easier the user's life. This is the case of reminders, systems that alter the phone profile settings based on the user information or create top groups of contacts. The low rated cases refer to the ones that only consider the user's location. Place-Its is for reminders, ParcTab for finding other devices to interact in different manners and

Social Net to match the locations time sharing with others. The ones classified as high use more personal information. Forget-me-Not uses the personal interactions with others, such as the documents sent or who we meet. In other scope, SenSay and HiCon are mostly based on sensors and schedule information to provide the user information, and ContextPhone uses the information of what the user is doing in ContextLogger and ContextMedia. CybreMinder uses the user location and schedule, together with his status and other people nearby. Friendlee and Contacts use communication monitoring to verify the user calls and messaging activity, in order to highlight an intimate group of friends.

Sensor Based Information

In this feature, applications that use only location sensors are excluded. The use of sensors to capture some outside information is adopted by SenSay, LilSys and HiCon. It is a good way to capture the context, because it helps showing the availability instead the presence in a device. It is possible to know if the user is in the room when his status is Online in an Instant Messaging service (LilSys for example). In SenSay, it is used to change the profile settings of the phone. There is no doubt that the sensors are useful but their sizes and appearances are difficultly accepted to be carried by the users.

Inter-Related Information

Most applications only deal with a single type of information. Either it is contextual information, or information from social networks or documents. However, those who deal with more than one type, do not inter-related that information, and treat it as a coherent whole. For these applications, for example the TapGlance, there are a lot of information (e-mails, agenda, SMSs, etc), but they are all disconnected/separated from the others. Associative PDA 2.0, comes with the purpose of resolving this problem, however their prototype only focus on note-taking (again a single type of information).

Social Interaction Aware

Applications are aware of social interaction in two specific matters. Social Awareness a feature in applications like "WhozThat?" or Social Net in matters of meeting people with the same interests or trying to find some more information about someone. CybreMinder, ContextPhone, Associative PDA 2.0 and "Forget-me-Not" are social aware with the purpose of doing something according to the presence of other people. ContextPhone tries to provide context information to help their contacts knowing if it is a good time to call and CybreMinder can trigger a reminder by a presence of somebody in the surroundings. However, keeping a history about social interactions is a great resource that is only focused in "Forget-me-Not" and Associative PDA 2.0. "Forget-me-Not" records context information such as, who I sent a document, and with whom I was when I did that. That may help me remember about other things related to the situation. Associative PDA 2.0 associates the persons nearby when a note was taken.

Invisible Contacts Context

In the applications that deal with the contacts' context information, revealing it may be a little intrusive. To fight this, applications like iCAMS try to hide the information, only using it to calculate what the best communication channel is.

Information Extraction Mode

When we are talking about using our own context for changing settings or reminders, we want that information to be extracted automatically, but when providing our context to others we may still want it to be delivered automatically, manual or even have both to choose. Most applications context extraction is made in an automatic mode, but some like Live Addressbook feature it manually. They claim that with this option they only show what they want but automatically, making some information invisible, turn that possible too. Doing that manually results in a big waste of time and certainly a big amount of times that the user forgets to change his state. In "WhozThat?", the social network profile is filled manually to later consulting by others, so does not need to be constantly refreshed. The Context-Phonebook and iCAMS are classified with automatic/manual because some of the context information is extracted automatically and other is manual.

Privacy Intrusion Level

Nowadays, privacy is a big issue almost everywhere. With the advances of technology and new inventions we tend to take some of peoples' privacy. This problem is more relevant when sharing our context with others. What is the limit for our privacy? What should we show? Those are questions that certainly vary from person to person, but it is possible to make a classification based on privacy intrusiveness. Most of the applications classified with low do not share context information with the contacts and those who have, like Live Addressbook or Context-Phonebook, or have the information hidden or only show the information that the user introduces manually. The applications classified with medium, without being very intrusive have some points where may cause some discomfort about the information revealed. With ParcTab and iCAMS it is possible to know where the user is though not with a perfect accuracy. With Live Contacts and Awarenex though the location is not exact it reveals schedule information, and LilSys is considered high because it is similar to those ones but has sensors that can be intrusive, such as the sensor that indicates if the door is opened or closed. ContextPhone is also highly intrusive due to his location accuracy that shows exactly where the user is and the people surrounding him. "WhozThat?" is in this category because the user can have a lot of information in the social network profiles that will be checked by others. There are some privacy issues in these applications but almost all of them have the option to hide the information they do not want to show to whom they want, and are used in a friend context. On one hand, the possibility of hiding the information is a relief but on the other hand an application like "WhozThat?" hiding the profiles to unknown people is taking away the its essence.

Information Storage/Processing

The information storage and processing is divided in two categories: local and centralized. The local refers to when the application occurs only in the users' device or uses direct communication with others to exchange information. Applications like SenSay or Place-Its that run on the user's device and do not need other users' information, are local. ContextPhone and Social Net are examples of applications that, though needing information from others, interact with them directly without a centralized server. The majority of these applications have a centralized server that is invoked when information is needed or constantly refreshed. Systems like SEREFE and Satchel that need to find out where documents are and start transfers are stipulated to use this type of storage and processing.

2.3. Summary

All the discussed applications improve the user's daily life and relation with his surrounding environment by using some kind of contextual or personal information. Although each project is presented as stronger in one or several topics, there are still some gaps that need to be filled and several information sources and scenarios that can be explored to improve the user's daily life.

One very interesting application is "Forget-Me-Not" due to the great capability of making use of social interactions. Other applications, like ContextPhone or CybreMinder, are social interaction-aware, but do not make full use of that awareness. "Forget-Me-Not" records social interactions, together with sending/receiving documents, for posterior help to recall about it. Also Associative PDA 2.0 records the user context when taking notes, for posterior recalling help. Friendlee and Contacts Application use communications information (call and messaging activity) to select the user's closest friends. However, there is not a system that records the social interactions with other persons in different matters. It would be very useful to have a system that could provide the information on the last documents I shared with a specific person and what and when were our last e-mails or SMS exchanges. It would also be very interesting to know when and where we were together or had a phone conversation.

There are a lot of applications that explore the contacts' context, where those who make a difference are ContextPhone and LilSys. Almost every systems use location information and some like these two use sensors to obtain context information. ContextPhone stands out for being aware of the social interactions of the contacts, knowing who surrounds them. LilSys strongest point is revealing information of other devices state. This area is filled with a big quantity of applications, that explore different ways to know information about the contacts, but privacy is always a big issue. These applications are interesting as well as the users allow and want their friends to be aware of their context information. The major problem is that no one wants that someone, not even their wife or best friend, to always know what he is doing, where he is or with whom. It is a huge level of intrusion that is difficult for anyone to accept.

Only a few sources of information use the huge "world" called Internet. "WhozThat?" searches for online social network profiles like facebook or MySpace and Contacts Application use a Directory Search to find more information about a contact in external databases like Yellow Pages.

Online social network profiles combined with other sources of information over the internet like search engines or blogs, can provide useful information, in personal or professional matters.

There are several sources of information which can be explored to help the user in different situations. However, many applications focus only on a single type of information, and those who focus on more deal with that as separate chunks of information instead of relate the information from the different sources. For instance, TapGlance proposes an interface to deal with different types of information, but do not establish any relation between them. It would be very useful to ask for information about a contact and receive the mail, SMS, phone calls and agenda activity, instead of searching for him in each one of that features.

The Satchel Project and SEREFE stood out for making the ubiquity of information possible, in their case documents, which is of use of the mobile devices also ubiquitous quality. With this it is possible to access documents everywhere at anytime.

		Types of information						Application Features				
	Features Applications	Online Information	Multi-Devices Information	Contacts Context Information	User Personal Information	Sensor-Based Information	Inter-related information	Social Interaction aware	Invisible Contacts Context	Information Extraction Mode	Privacy Intrusion Level	Information storage/ processing
Gathering contacts information	ParcTab	Inexistent	No	Low	Low	No	No	No	No	Automatic	Medium	Local
	Live AddressBook	Inexistent	No	Medium	Inexistent	No	n.a	No	No	Manual	Low	Centralized
	Context-Phonebook	Inexistent	No	Medium	Inexistent	No	n.a	No	No	Automatic / Manual	Low	Centralized
	Awarenex	Inexistent	Yes	High	Inexistent	No	n.a	No	No	Automatic	Medium	Centralized
	iCams	Inexistent	No	Medium	Inexistent	No	n.a	No	Some	Automatic / Manual	Medium	Centralized
	LiveContacts	Low	No	Medium	Inexistent	No	n.a	No	No	Automatic	Medium	Centralized
	LilSys	Low	Yes	High	Inexistent	Yes	n.a	No	No	Automatic	High	Centralized
	ContextPhone	Inexistent	No	High	High	Yes	No	Yes	No	Automatic	High	Local
	WhozThat?	Medium	No	Inexistent	Inexistent	No	n.a	Yes	No	Manual	High	Centralized
	Contacts	Medium	No	Low	Medium	No	No	Yes	No	Automatic	Low	Local
	Friendlee	Inexistent	No	High	Medium	No	No	Yes	No	Automatic	High	Centralized
Use of Personal Information in Mobile Devices	Forget-Me-Not	Inexistent	No	Inexistent	High	No	No	Yes	n.a	Automatic	Low	Local
	CybreMinder	Inexistent	No	Low	High	No	No	Yes	n.a	Automatic	Low	n.a
	SocialNet	Inexistent	No	Low	Low	No	No	Yes	n.a	Automatic	Low	Local
	Sensay	Inexistent	No	Inexistent	High	Yes	No	No	n.a	Automatic	Low	Local
	Place-Its	Inexistent	No	Inexistent	Low	No	No	No	n.a	Automatic	Low	Local
	HiCon	Inexistent	No	High	High	Yes	No	Yes	No	Automatic	High	n.a
	TapGlance	Inexistent	No	Low	Medium	No	No	No	No	Automatic	Low	Local
	Associative PDA 2.0	Inexistent	No	n.a	Medium	Yes	No	Yes	No	Automatic	Low	Local
Documents Ubiquity	Satchel	Inexistent	Yes	Low	n.a	n.a.	n.a	No	n.a	n.a	Inexistent	Centralized
	SEREFE	Inexistent	Yes	n.a	Inexistent	No	n.a	No	n.a	n.a	Inexistent	Centralized

Table 2.1: Overall Comparison of the different applications according to different features

3 Extracting and Relating Relevant Personal Information

In this section we describe our approach, focusing on its most important features and presenting some possible scenarios. We detail the framework architecture and explain how it is used to solve the main questions raised in our research. We discuss the sources and ways of extracting relevant information, either personal (from the users' devices) or public (mainly from online sources) to help them in their daily tasks. Also, we show the possibility of managing the information retrieved from those different sources as a coherent whole, inter-relating it, instead of separate chunks. Finally, we present an example application that tries to provide relevant information about some person, built on top of the aforementioned framework.

3.1. The Approach

There is an imense quantity of personal information about us, and the people we interact with, in our personal computers and mobile devices. However, to access that information, the users have to know its whereabouts or how to find it, wherever and whenever they need it. With the increasing storage capacities of personal computers and mobile devices, it is impossible for the

users to memorize what is and is not stored therein, and most importantly, how to gather all the information they require.

Nowadays, the almost complete wireless network coverage allows the access to other devices (personal computers) and to the world-wide-web, and the mobile devices' advantage of being always with their owners, makes these devices an excellent entry point to help clarifying the users' doubts and help them on their daily basic tasks. In a social environment it is natural to wonder *I know that person, but where from?* or *I had some things to discuss with Jack, but what were they?*. The approach presented herein is able to provide answers to those questions by gathering and interrelating information from the users' devices enriched with other public information sources, to offer them context- and personally-sensitive information when it is most needed.

In the next sub-sections we describe some possible scenarios and the most important features of our approach.

3.1.1. Possible Scenarios

The framework developed in this thesis' context aims to help the user in several different situations. We outline some meaningful ones:

I am at a party and I find someone that seems familiar talking to a friend of mine. Using his Bluetooth ID, I ask the system about our past interactions. I get the information that we were together two years ago at the CHI Conference, and had exchanged 2 email messages and a document (that I can access if I want to). The document's subject is also shown, as are the people I have forwarded it to. I ask for more information about him. I continue drinking my vodka and see that his wife is Maria Parker and was in a conference in Japan a week ago.

My phone is ringing. It is a work colleague with whom I haven't had contact for a long time, but worked on some projects with. I accept the call and put it on the hand-free mode so I can see the information shown. It appears that we haven't had personal contact for six months, and the last time we shared an e-mail was two years ago. It is also referenced that by that time we both worked on Human Resources and shared 3 projects. The related documents are presented. After the regular introductory talk, he says that he has to do a project related to one we worked on and needs some documentation to understand and modify it. Thanks to GeniusPhone, an application that instantiates the approach presented in this thesis, I know what that project is about and immediately send him the related information.

I have a scheduled meeting in a couple of hours, and GeniusPhone reminds me about it. At the same time, it gathers information about Persons, Documents, SMS or Mails that may be relevant to the meeting. I navigate the data and get a document in the same folder as a suggested one, and add a mail message from a suggested participant.

I am at a meeting and my boss says that after the last break, we would discuss some matters about the new technologies in mobile devices for blind persons. Apparently he does not know that I have no experience in that area, but I see one opportunity to look good anyway. I search my new mobile phone application for this subject and the results show me that my friend Harris sent me some mails about his PhD that involves blind person's usage of mobile devices, and a SMS inviting me to his final Doctorate discussion/presentation. At the same time, some information about the recent advances, extracted from Google, is presented. After

seeing the results I have an idea about this subject, but to a better preparation I call my friend Harris which resumes me his knowledge about the area.

3.1.2. Information Sources

There are many information sources that we can resort to help the user seeking relevant information. Some existent applications rely on other users to provide their personal information. This can be a problem in two specific ways: First, it is very difficult for someone to accept that others have access to their information; Second, this is only reasonable in an intimate social context (If I share my information with someone, it has to be a very close friend). Also, it is only applicable in a context where we need information about a person. To avoid privacy issues and dependency on what others may provide, we consider only the personal information existent in the user's devices (personal computers and mobile devices) and public sources of information.

Our personal computers' storage capacities allow us to have an enormous quantity of information, some of which we may have no idea where to find. Several documents and different applications (calendars, mail messages, Instant Messaging logs, etc) are spread out by many locations/folders, and even for each one of these applications, many times, it is difficult to find/reach what the user is looking for.

Mobile devices, with the advantage of being always with their owners, are aware of most of their daily interactions and habits. However, and even more than in personal computers, this information is not integrated turning its exploration into an arduous task.

These devices, personal computers and mobile devices, provide an enormous quantity of personal information. Some good examples are: documents; mail; MSN logs; calls logs; SMS and presence detection.

The world-wide-web is a gigantic "universe of information", as we can get most of our doubts or curiosities enlightened, either from a search engine, social networks profiles or blogs. Nowadays, search engines can provide information about everything, either showing personal, event or institutional pages. Sources like Wikipedia or similar sites provide additional information about a variety of subjects. Also, in blogs people describe their entire life, their work, personal life, children, etc. In online social networks, many persons only keep a close relationship to a small group of "friends", adding the others only for being friends of a friend (FOAF) or having similar interests. All of these online sources reinforce the idea that the Internet contains many unknown or hard to recall information, that can be useful to help the users in several situations of their daily life. Some good examples of public/online sources are: Facebook, Hi5, among other social networks; Twitter; LinkedIn as well as other professional social networks; Google and other search engines; BlogSpot and Wikipedia.

3.1.3. Personal Information as a filter to Public Sources

We are dealing with two different types of information: personal information existent in our devices; and public information mainly from online sources. Although the users' devices provide reliable and relevant information from the users' point of view, with a proper meaning to them, public sources generally are much more ambiguous. For example, if a user searches for information about *Tony Parker* on his devices, it is most likely that he/she finds information about a single person (in a few cases there may be two, but hardly more). However, if he searches for the same person in a search engine there will be thousands of results relative to several different *Tony Parkers*. How can the user know who is the "right" one? Also, there can be a lot of information about the same person, how can the user get information about the subjects that really matter to him/her? The personal information retrieved from the user's devices, is the perfect candidate to help filtering the ambiguous information that public sources provide. Thus, it is possible to identify which data is related to our search and at the same time collect relevant results from the user point of view. On the previous example, it would be possible to filter the data about that specific *Tony Parker*, on subjects that really matter to the user. If he shared some mail messages with him about politics and mobile devices, he wants the information about that and not about rugby or football, which he might also like but is not that important for the user.

3.1.4. Information as a Coherent Whole

Like previously mentioned, there are several possible sources of information. Most past and recent applications deal only with one type of information, and those who deal with more do not establish any relation between them. Each information source has a single representation, and due to this heterogeneity it is essential to find a single structure capable of dealing with the different kinds of sources, inter-relating the information and representing it as a coherent whole, instead of isolated chunks. This would turn the process of finding information about something and its references much easier and general. For instance, if I need some information about the CHI conference I could search my mailbox trying to find some e-mail that talk about it; search on Google and Wikipedia for CHI and finally search in my SMS inbox to find who mentioned that conference to me a year ago. In possession of an unified integrated index, I could search only for CHI Conference, immediately obtaining information from all the above sources inter-related as a coherent whole.

The subsections above describe the main focuses and goals of our framework, and each one of these features has some challenges inherent to its nature. We considered each of the main features and its challenges, together with some smaller issues, in order to build a framework adjusted to our needs. Also, we reflected on the integration of the different questions we found important to structure our approach.

Like aforementioned, there are many information sources that we can use in the user's benefit and it is impossible to know when a new one will appear. Considering this, it was crucial that introducing a new information source could be easily done, without changing and compromising other modules.

A lot of information can be retrieved from these sources, so there was the need to keep and structure all that information equally (either from users' devices or online sources). A special attention had to be given to non-structured sources in order to represent it as the remaining information. Since there is data more relevant, precise or credible there was the need to rate it, so the user (and the system itself during his iterative searching proccess) could have that perception.

Due to these differences in data precision and credibility, and as previously mentioned, personal information in the users' devices, being more credible and relevant from the user point of view, can be used to filter information from more ambiguous sources (mainly online). To accomplish this we had to elaborate a process of continuous iterations, so the information could be reevaluated and reweighted. Thus, the information that really matters to the user can presented to him, including some extracted from less trustable sources.

To accomplish these challenges we built a framework with an architecture divided in three main modules detailed in the following sections.

3.2. Architecture

The framework architecture is based on three main components: Plugins; Plugin Manager and Coordinator (Figure 3.1). Plugins are responsible for extracting the data from the different information sources and structure it into the common representation. The non-structured information is marked by the plugins, so it can be identified and sent to the Natural Language Module by the Coordinator. These plugins register in the Plugin Manager, which is responsible for selecting which plugins are suitable for each search. The Coordinator is responsible for requesting information from the plugins, store the results in a knowledge base, and iteratively requesting more information from the different sources to clarify or reinforce some knowledge. This framework is developed in the Python language, version 2.6.

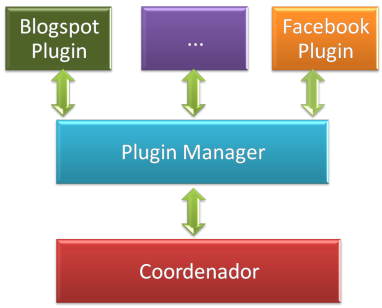

Figure 3.1: Main architecture components

In the following subsections we describe the main components, together with the coordinator modules and the architecture workflow.

3.2.1. Plugins

One of the most important assumptions of our approach relies in the ability to access personal and public information. To feed the system with this essential data, the system features a plug-in based architecture corresponding to the different information sources. Plugins are the direct contact with those sources and each one of these plugins inherits from a single class due to the

similarities and shared properties among all of them.

Different sources have different structures and representations, so the information retrieved from each plug-in needs to be transformed to a single one, to simplify further processing of this information. To accomplish this, each plugin has an adapter where it sends the information to be processed and transformed into the unique representation. With this, each plug-in is able to produce structured information (relative to a Person, Document, Event,...) or tag it as unstructured (subject to further contextual processing by the natural language processing module).

("Tony Parker", "Interests in", "soccer", 0.7, "facebook")
("Tony Parker", "Interests in", "accessibility", 0.8, "facebook")
("Tony Parker", "works in", "INESC-ID", 0.8, "facebook")
("non-structured", "Tony Parker", "Here it goes the non structured information", 0.7, "facebook")
("Tony Parker", "relationship", "married", 0.7, "facebook")
("Tony Parker", "graduated in", "Science of Sports"", 0.7, "facebook")
("Tony Parker", "hometown", "Dallas", 0.7, "facebook")
("Tony Parker", "activity", "Teacher", 0.7, "facebook")

Figure 3.2: Plugin retrieved information structured

The single representation used is a list containing tuples with some characteristics of the information found. These features are: subject (the search); predicate (the relation); subject (the information); a weight the plugin assigns to that tuple; and the source of the information (An example of a plugin's result is on Figure 3.2). This structure will be better discussed on the Coordinator section.

The weight value is the confidence the plugin has on that piece of information and its scale is between 0 and 1. These values depend on the credibility and relevance that piece of information could have to users. For example, if the users' find information about some person's interests and favourite TV shows, it is understandable that the interests are more relevant to them. Also, if some information needs further natural language processing, it has less credibility, so less confidence. We chose this scale to have a limited space of values that allow us to convert it as a percentage. Also, it facilitates its usage. The weight is not relevant when we desire to store concrete properties of a given object, because they are not taken as probable or improbable. For instance, the tuple referring that *Tony Parker* sent me "*DocumentX.pdf*" needs to be weighted; however, the size, extension and path of the document do not need, because they are properties of that document. To deal with these cases, we assign the maximum value of 1, but do not need to use it.

Each plugin returns a variable that marks the necessity of coming back, or not, to that plugin with additional context. This is important to avoid searching for information in a specific source without good perspectives of finding new relevant information. For instance, in a search for Tony Parker, if a single person is found in facebook, there will be no need to come back to that plugin for additional information, because the relevant one was retrieved. However, if two or more Tony Parkers were found, there will be the need to come back to the plugin in order to disambiguate with the additional context to find which the Tony Parker we are looking for is.

Upon registration the plug-ins inform the Plugin Manager about the information they can obtain, enabling it to present requests to the most adequate sources. Their registration is based on some base fields that we considered to be the most important ones, according to the predicted use scenarios. Since currently we are dealing only with people, the main characteristics are: interests, work, contacts, birth and interactions. However, this classification can be easily extended, as we will describe in the next section (Plugin Manager).

At this point, we have four different plugins involving social networks (Facebook), blogs(BlogSpot), search engines (Wikipedia) and personal information (Scribe).

The plugins that deal with the personal information existent in the user devices need to be previously supplied by the indexed files. It is impossible to search in that devices every time the users need some information. So, the personal information needs to be previously indexed, either on personal computers or mobile devices. Currently, we index e-mails on personal computers using Scribe, an existent monitor of personal information that collects information from various sources (from e-mails, to calendars or documents) (Gonçalves, 2007). However, some adaptations were made to support frequent keywords and e-mail platforms like thunderbird and Gmail. Currently, we find some keywords that represent the most important (frequent) words exchanged by the user and a searched person, using *tfidf*. Then, we select a few keywords for each mail message and count the number of mail messages that contain that keyword. The confidence of each keyword will be dependent of the number of times it appears. Using Scribe, we can access the mail messages involving some person and use some important fields (subject, date, direction, participants and attachments) to help the user. This Plugin's capabilities are "contacts" and "interactions".

On Facebook, since we cannot discover and access a friend of a friend profile (due to privacy issues), we base our searches only on our friends' pictures and profile. When a facebook user fills many fields with his information it can be a great source. It has information about the users' interests, activities, relationship and cultural preferences among others. Like mentioned above, and because not all the information has the same interest, the plugin does not give the same weight to all the information it extracts. Also, when it finds non-structured information it decreases the field natural value. To get the information we use the facebook API, searching first for the userID based on his name, and with that search for the most interesting fields. This plugin registers on Plugin Manager with the following capabilities: "interests"; "work" and "birth".

On Wikipedia, we search for the person's web page and knowing the webpage structure we extract the information that first appears on Wikipedia searches. It is like an abstract and shows the most important things to know about that person. We do not extract more information because it would be entering on a bigger level of detail, which we do not want in this application, since we want synthetic and objective information. That information is marked as unstructured to further analysis from the Natural Language Module. This plugin registers with the following capabilities: "work" and "birth".

On BlogSpot we search for the section "about me" in blogs. But first, we search in google blogsearch for blogs using the person's name and possibly some additional context, and select only the BlogSpot ones. Then, we search those which its owner has the name of the searched person and extract that information from the "about me" section. We mark it as non-structured for further analysis. It registers with "interests" and "work" capabilities.

It is important to mention that in some cases, at this point Facebook and Scribe, when we find more than one possible persons, the weight is slightly decreased. This decision is supported by the fact that, in these plugins, if there are two or more people found, at least one is not the person we are looking for. On the other side, in BlogSpot, it is perfectly possible that a single person has two or more different blogs, so it does not apply to all plugins.

The biggest advantage on a plug-in based system is that it eases the addition of new plug-ins, therefore easily extending the system with new information sources. If a new source is found, it is only required to add the self-contained code of that plug-in, without changing the other parts of the architecture. As the iterative search process and the Plugin Manager base their decisions on the capabilities declared by the plug-ins upon registration, integration with the remaining information sources is always assured. In that matter, it is only necessary to choose the plugin main capabilities upon registration.

3.2.2. Plugin Manager

The Plugin Manager's main task is to decide which plugins should be called, considering the current search. To accomplish this, it has to know which plugins can provide useful and relevant information, and avoid making repetitive and useless searches. Like previously mentioned, each plugin has to register in the Plugin Manager informing what kind of information they can obtain. Figure 3.3 shows, in 1, the Plugin Manager asking for plugins registration and, in 2, their response with their search capabilities. The plugin registration has three levels. The first one, "all", contains all the plugins which registered on the Plugin Manager to prevent some case that we need to search all the plugins to find everything we can about something. The second level is related to the type of searches the plugin can make. This includes persons, events, subjects or institutions. The third level depends on the second one and is related to the main characteristics chosen by us. These are the most important features we found for each one second level types. Related to persons we have interests, work, birth, interactions and contacts. On the other side, with events, we can have type, date, location, participants and description. As we are dealing with People, we present an example of our current list of BlogSpot plugin's capabilities, considering that scope:

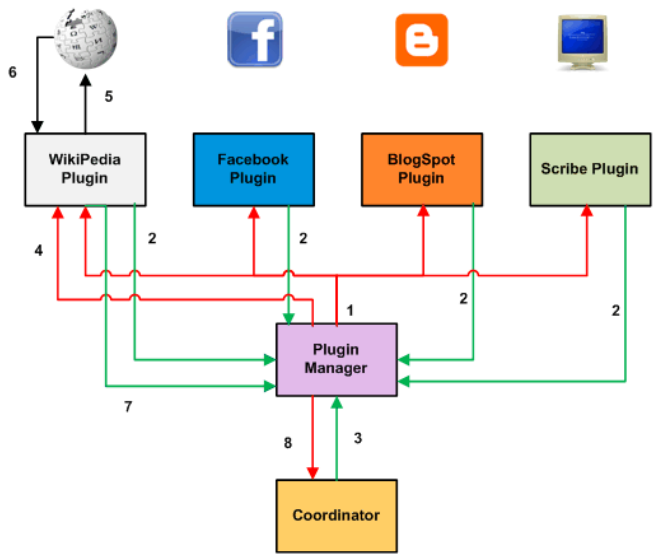

Figure 3.3: Plugins Registration in 1 and 2; Plugin Manager selection of plugins procedure in 3-8

```
capabilities = ["people", "interests", "work"]
```

If we add more types of information it is very simple. Since BlogSpot plugin can also provide information about an institution, mainly about its description, foundation or people involved, to register on the Plugin Manager, it has only to add some capabilities to the list:

```
capabilities = ["people", "interests", "work",
"institution", "description", foundation", "participants"]
```

Those characteristics used upon registration were chosen by us, following our own criteria for the most important issues for each domain. However, introducing a new main characteristic is a very simple procedure. Although these capabilities are documented, Plugin Manager does not establish any restriction to Plugin's capabilities. There is a relation of trust between them and if a new capability is found it needs only to be added on the plugins' side. Other kinds of information, not considered as main characteristics, are also retrieved, and can still be presented to the user, but will not be considered in the Plugin Manager decision-making.

Like mentioned on Plugins' section, to avoid repetitive and useless searches, they send a "comming back" flag and the Plugin Manager checks which plugins are marked as "not worth searching". Also, the Plugin Manager guarantees that in each one of the search iterations, though they can be in more than one searchable capabilities, each plugin is called at most one time. Since the context, if there is some, is the same for every capability, there is no need/utility to search the same plugin with the same additional information.

When the Plugin Manager receives the searching features from the coordinator (Figure 3.3, in 3) it decides which plugins will be called, because it knows which one returns what and which ones are worth to be called (In this example, only Wikipedia Plugin will be called, in 4). The plugins access their sources (5) and retrieve the information (6). After receiving the search results from all plugins (7), it sends the results to the coordinator (8). This results are a list from all sources that returned information, and all on the common representation that we already mentioned. The non-structured information is marked as that for further analysis of Natural Language Processing Module.

3.2.3. Coordinator

The main module of the system is the Coordinator which is responsible for requesting information from the plugins, store the results in the knowledge base, and iteratively requesting more information from the different sources to clarify or reinforce some knowledge. Ultimately, it will gather the data with higher confidence levels and send it to the user. Indeed, without this module the information in the knowledge base would grow uncontrollably in every iteration and would retrieve everything in every search.

One of this module's main tasks is to ensure the storage of the information received from the plug-ins in the knowledge base. While some information sources are able to produce structured knowledge, others are likely to need some extra consideration. To this end, our framework features a Natural Language module, one that is able to look into a text and try to retrieve structured information from it. The coordinator is responsible for deciding when this needs to be performed, based on the type of reply given by the plugins (structured or not structured). Structured information is based on tuples (Subject, Predicate, Object, weight, source) due to our necessity of small and concise/synthetic pieces of information, instead of big paragraphs that would take longer to understand with much useless information. Subject, predicate and object represent the informa-

tion and weight and source are features of that information. The structure is always the same, but in some cases the weight and source are not used. In our current work, the best example is related with mail messages. The information that users have shared some mail message with a friend is weighted based on its credibility, but the relation between that message and its attributes does not need to have specific weight associated. We chose this option because we think there is no added value in having different weights for the different fields.

The weight allows the plug-ins to provide information with different confidence levels on their trustworthiness or value. For instance, considering Facebook, some persons fill their interests like an enumeration (separated by commas or newlines), while others write a text describing it. Considering the latter, as it requires further analysis, its weight will be less than in the former scenarios, because information already structured by the user is most likely to be right than information structured automatically by the Natural Language module. In this case, the application scope defines the weight but this can also be useful for a plug-in to tag some piece of information as less or more trustable. Besides the weight given by the plug-in to the information, the Coordinator calculates a new value, considering also the weight it gives to each plug-in. It is natural that a search on Google has less credibility than one performed on Facebook, due to their non-structured and structured character; and personal information (from the user's document space) is more credible and important, in the users' point of view, than that from the aforementioned sources. The calculation of the new weight is basic and consists only in a multiplication of the two values, because we want both values to count the same. We can have plugins that find their information relevant, but the Coordinator does not assert much credibility to that plugins, so the weight needs to be diminished, or vice-versa, a plugin credible to the coordinator, but less certain of its information. Figure 3.4 shows an example. This allows us to consider two features, the confidence the Coordinator has in the plug-in, and the confidence and emphasis the plug-in gives to that information.

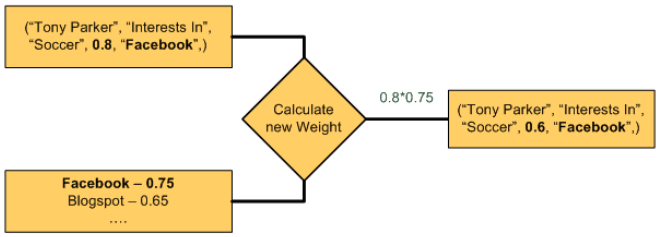

Figure 3.4: Calculation the weight of a piece of information example

The hardest task for the Coordinator is deciding what to present to the user. The knowledge base is fed with enormous amounts of information that can have several relations between them. However, for a particular search, the user desires a finite, concise and understandable result. What should it iterate? When should it stop? To answer these questions, the coordinator has to analyze the new information retrieved by the plugins and compare it with the existent in the knowledge base, to find out the necessity of iterating. We explain all this process on section 3.3.

Like mentioned above, before storing the information on the knowledge base or just comparing it, there is the necessity of process the non-structured information sending it to the Natural Language Module. We explain this module on the next sub-section.

Figure 3.5: Example of unstructured information on facebook activities (left) and structured (right)

Non-structured information and Natural Language Module

In a perfect scenario all sources provide structured information, and there are some that can, e.g., the personal information retrieved from the user's devices or most of the information extracted by social networks. That allows us to easily manipulate the information and store it as an inter-related whole.

However, that is not always possible due to the large amount of useful sources that can be explored. Mainly on online sources, like blogs, personal pages or Wikipedia, the information is not structured. We cannot add an entire text or sentence to the knowledge base, or it would be useless to the user, which need synthetic and concise results. In the wide range of information sources available online, we have explored so far the ones involving personal blogs and Wikipedia. We have created mechanisms to deal with known blog and wikipedia pages structure, resorting to an HTML parser, and can easily extract the information block where the users talk about themselves. This un-structured information is not useful, unless we can retrieve the information that really matters to us and structure it in the same way we do to the other sources.

To this end, for each piece of information, plugins mark it as structured or unstructured, so the Coordinator can redirect that part to the Natural Language Module. Plugins that need to access to web pages with unstructured information comprises simple algorithms to identify interesting blocks of information and ignore all page layout related information as well as other less identifiable sentences. At this point, it is quite simple. On the other hand, there may be plugins that deal with the both types (structured or not). In Figure 3.5 we show an example of two different manners to describe the user activities in facebook. On the first one it is unstructured as a text, and on the second, it is structured (separated by commas or newline). To detect when we are dealing with unstructured information, we establish a relation between the number of words and punctuation/newlines, because when they occur very often it means that probably it is a sequence of responses for that field.

To deal with non structured sets of information, the Coordinator sends it to the Natural Language Module, which tries to filter the data and extract information that can enrich our knowledge base. We will explain the algorithm behaviour making use of the following example:

"My research interests are in improving accessibility to mobile devices and to empower disabled

persons through their use. I am a PhD Student at the Technical Superior Institute, Technical University of Lisbon, in Portugal and a researcher at INESC-ID."

1. **Splitting in sentences** - Split the block of information in sentences, making use of the punctuation (".", "!", ":", ";", "..." and "?");

   [My research interests are in improving accessibility to mobile devices and to empower disabled persons through their use]

   [I am a PhD Student at the Technical Superior Institute, Technical University of Lisbon, in Portugal and a researcher at INESC-ID]

2. **Splitting by keywords** - Split the sentences in relevant parts, we use some keywords we found the most relevant (interest, like, work, am, is, etc). Each part ends when other keyword is found or the final of the sentence is reached;

   [**interests** are in improving accessibility to mobile devices and to empower disabled persons through their use]

   [**am** a PhD Student at the Technical Superior Institute, Technical University of Lisbon, in Portugal and a]

   [**researcher** at INESC-ID]

3. **Atomicity** - The commas, prepositions and other words usually used to provide continuity of a subject ("and", "at") are used to divide the information, so each feature can be atomic and better structured;

   interests - "are"; "improving accessibility to mobile devices"; " to empower disabled persons through their use"

   am - "PhD Student"; "Technical Superior Institute"; "Technical University of Lisbon"; "Portugal"; "a"

   researcher - "INESC-ID"

4. **Removing irrelevants** - Some words ("a", "the", etc) are removed for being irrelevant, and some parts are also removed because they enter in an irrelevant level of detail (parts following, "whose", "while", "where", "although", etc, and parts between parenthesis).

   interests - "improving accessibility to mobile devices"; " to empower disabled persons"

   am - "PhD Student"; "Technical Superior Institute"; "Technical University of Lisbon"; "Portugal"

   researcher - "INESC-ID"

5. **Replacing** - Replace some keywords to make more sense, and others to the common representation of the main features.

   Interests in - "improving accessibility to mobile devices"; " to empower disabled persons"

   is - "PhD Student"; "Technical Superior Institute"; "Technical University of Lisbon"; "Portugal"

   research - "INESC-ID"

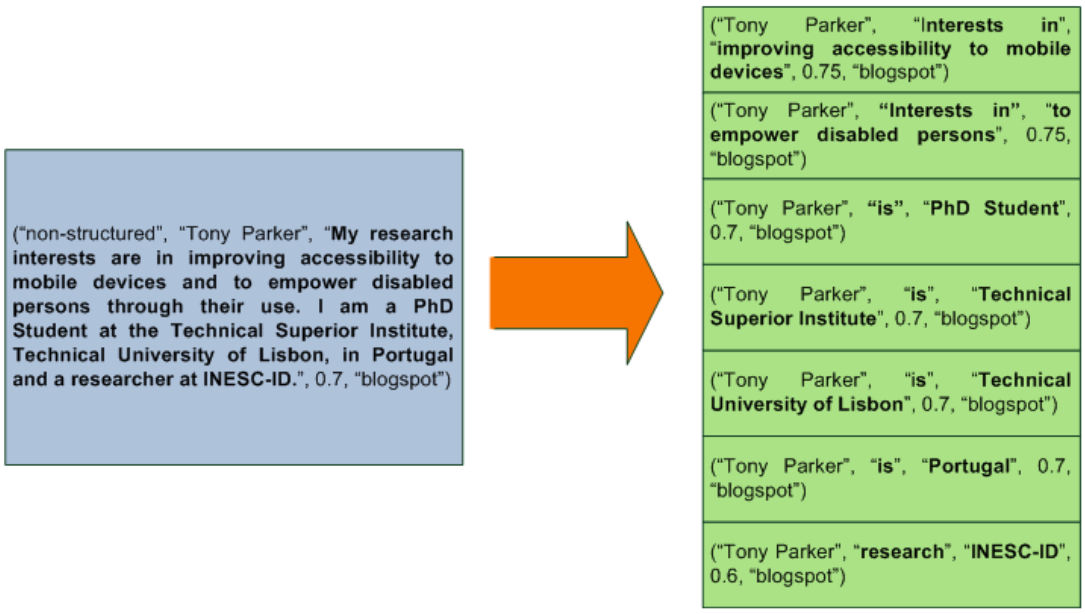

Figure 3.6: Natural Language Input and Output example

Although still limited, the module is already able to tag the components of a given text and filter data accordingly. Moreover, it encloses functions to filter unwanted data. Since our intention is to create atomic relations instead of big sentences, and punctuation (except ",") usually means changing of subject, Step 1 is very simple and only separates the text in different sentences. In step 2, the words are stemmed in order to get their root and compare it with our keywords. Using it, we can match, for instance "work", "working", "worked" and "works", because all these words' root is "work". Each sentence is divided by this keywords. Each phrase starts when one of them is found and ends when other one starts or the end of sentence. In step 3, since we already have the sentence divided in phrases, our relations are selected (the keywords). So, we use punctuation or words that generally are used to provide continuity of a subject to divide our oration atomically to better structure the information. Since we are trying to extract the most important features and do not want a big level of detail, some parts of sentences can be dismissed. In step 4, we use some words that usually mean a bigger level of detail and delete the text ahead. Also, we delete some words that add nothing to results. In step 5, we replace some relation words to make more sense. Also, the relations that are related to our main features (interests, work, contacts, etc), we replace it for the common representation of those features. In Figure 3.6, we can see an example of what entries in the Natural Language Module and the result structured in several tuples, as our common representation. This information is now structured and prepared to be stored or compared in the Knowledge Base.

3.2.4. Knowledge Base

With an enormous amount of information that can be retrieved from several sources, there is the need to store and organize it to get real knowledge. Otherwise, the data would be spread out, and it would be impossible to extract meaningful and inter-related information. To represent

the information extracted we use a Knowledge base, a semantic network which gathers all relevant information collected from the different plugins, inter-relating it into a coherent whole. Our internal representation is based on Subject-Predicate-Object relations, for example "Tony Parker-Interests in-Mobile Devices", since we need objective and synthetic information. We do not want big descriptions that take too long to read and understand, with higher detail levels. One of our framework's bases is the simplicity and atomic character of the stored and presented information. To accomplish that, the Knowledge Base uses the *Resource Description Framework* (RDF) which is a match to our needs, representing the information as triples (Subject, Predicate, Object). RDF is simple to use and allows some manipulation due to its permissions to store everything. There are no obligations or restrictions to the information represented on the Knowledge Base, so the triple subject-predicate-object can be anything we want. This is very useful for our two case frames.

The information considered is represented by resorting to two different case frames: simple or weighted. The simple case frame is used when we desire to store the properties of a given object. They are not taken as probable or improbable. They are just elements defining some entity. A simple case frame example is the characterization of a document, e.g., name, path, creation date, modification date, keywords,... An example is presented in Figure 3.7. In this case frame, is used the standard RDF representation, with the subject being the node representing the document, the predicate are the different relations (characteristics) and objects are their values.

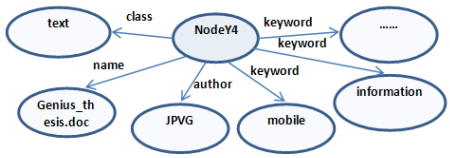

Figure 3.7: Simple RDF Case Frame Example (Doc)

However, while the information that defines some node can use the standard representation, when we need to represent other features, the simple case frame is not enough. In particular, to some information we desire to add two extra items: weight and origin. Figure 3.8 presents a scenario where the weighted case frame is used.

In this frame, the components of a traditional relation are placed as links (predicates) between nodes and objects. This enables us to include as many features as we desire for this particular information. So, the subject would be the node representing this piece of information, the different predicates are the set subject, relation, object, weight and source, and the objects are their values.

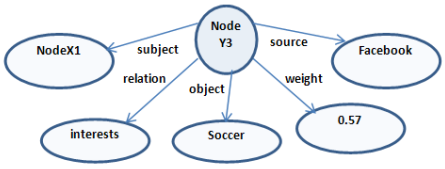

Figure 3.8: Weighted RDF Case Frame Example

Figure 3.9 presents a piece of the framework knowledge base enclosing both case frames and relations between the user's interests and incoming messages. It is relevant to mention that our subjects on relations are always nodes that have unique identifiers; either they are persons, events or documents. It was an option to have their names working as identifiers so we would not need these nodes, but with them we can have more flexibility. For example, we can have a person with two different names (only first and last, or the complete one), so we want to be able to represent them as the same, and not two different persons.

There is the need to access the information represented in the Knowledge Base, either to present

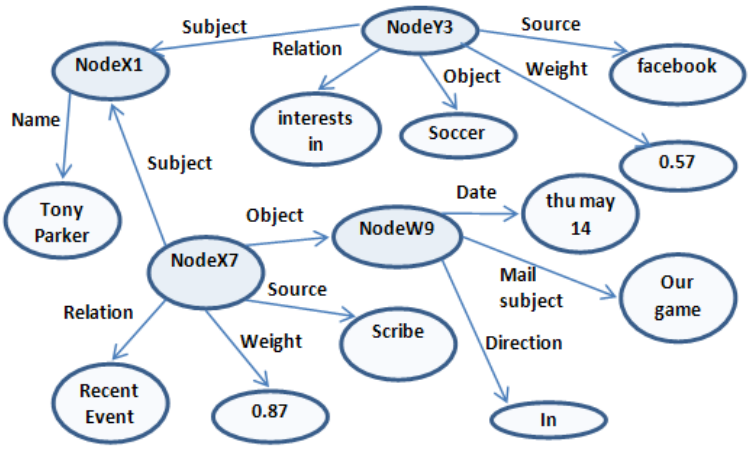

Figure 3.9: Stored information example

it to the user, or to verify if there is information that need to be reinforced when new information arrives. To accomplish that, we use RDF SPARQL query language. Like other query languages, it allows us to select some fields of the Knowledge Base, based on some conditions (”Where” clauses). In the following source code example we are searching the knowledge base, to know if the new information found is already on it or not.

```
spqgraph = SPARQLGraph(graph=self.knowledgeGraph)

select = ("?node", "?wei")

where = GraphPattern([
("?person", Literal('name'), Literal("Tony Parker")),
("?node", Literal('subject'), "?person"),
("?node", Literal('relation'), Literal("interests in")),
("?node", Literal('object'), Literal("soccer")),
("?node", Literal('weight'), "?wei")])

relationForWeight = query(spqgraph, selectWei, whereWei)
```

The ”Where” clauses filter the information that corresponds to the searched person, relation and object. With ”select”, we find the parent node and the information weight. Also, with this language we can easily add constrictions. A good example is when we decide which information we should show to the user. We can consider a minimum weight (for example, 0.5 of 1), in order to present only information with good possibilities of being truth and useful.

```
where2.addConstraint(sparqlOperators.gt("?wei",float("0.50")))
```

3.3. Iteratively seeking for information

The most important Module supporting this process is the Coordinator. It inserts the new information in the Knowledge Base, checks the necessity of more iteration and chooses what should be shown to the user. All of this is made resorting to the information stored in our semantic network.

In Figure 3.10, we present the workflow involving the coordination of this iteration process. First, the Coordinator sends to the Plugin Manager what it wants to retrieve (1), and the Plugin Manager, after getting that information from the plugins, returns the resulting list of tuples(2). The coordinator checks if there is non-structured information, by looking at the first element of each tuple, and sends it to the Natural Language Module (3), which returns it properly structured (4). Then, the Coordinator sends the data to the Knowledge Base, which adds or reinforces the information in the semantic network (5). At the same time, and based on the new and consulting the old information in the Knowledge Base (6), the Coordinator decides the necessity of iterating or not. If there is no need to keep searching for information, the Coordinator sends the information to the user (7). On the other hand, if it is useful to keep iterating, the Coordinator asks again the Plugin Manager for information (1), but this time with additional context that will possibly help the searches. This process, from 1 to 6 is repeated until the Coordinator decides that one more iteration is useless.

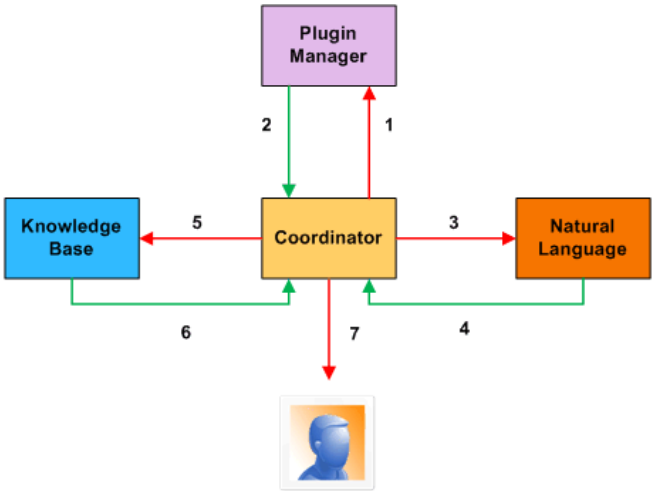

Figure 3.10: Workflow involving the Coordinator and the process of iteration

Like previously mentioned, plug-ins registration phase is based on the most important fields for the application, which can provide us a worthy and precious description. Although the plugins return a variable number of fields, the iteration is also based on that specification. Recalling, since we are dealing only with people, the main characteristics are interests, work, birth, contacts and interactions, but can easily be extended by adding it on a python dictionary. This dictionary contains the words that are related to each main characteristic. In contacts we have email, address and phone number; in interactions, we have recent event, because that's our specification for exchanged emails.

The coordinator checks, for each main field in the previous iteration, the number of new triples with a reasonable value of credibility (for example, 0.5 from 0 to 1). If there is a considerable set of new and valuable information, it is not needed to search again that field. On the other hand, if there are not new values (or only a few), it is worth to search again that field. All those fields are marked to being searched on the next iteration, so the Coordinator can call the field functions

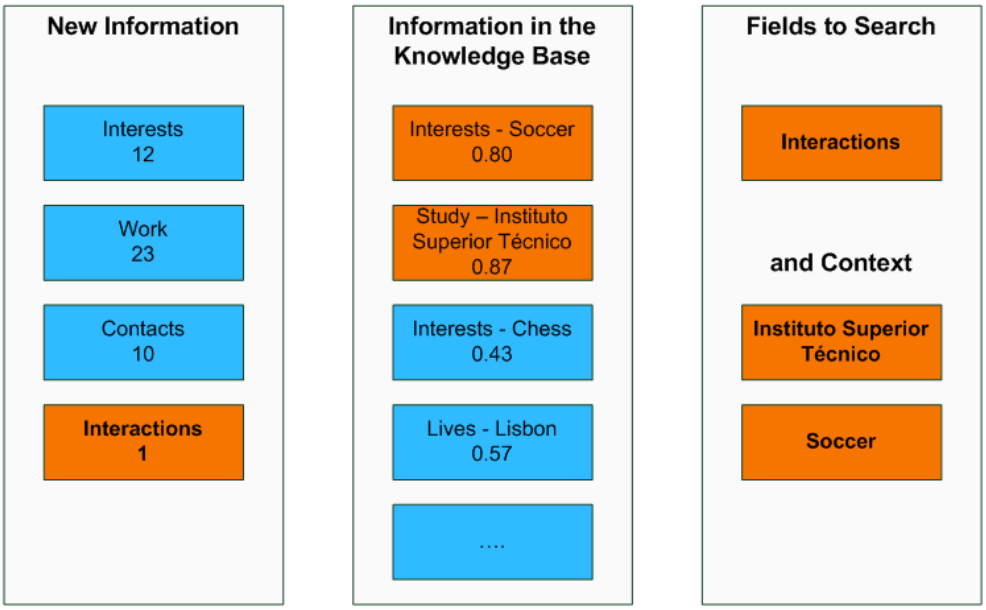

Figure 3.11: Persons main characteristics and its quantity of information extracted; some information in the KB and its weight; and the field and information used in the next iteration

which basically send to the Plugin Manager the kind of information to be searched. In these iterations, the best/higher values in the knowledge base are used as additional context. Thus, highly valued information is used to improve the quality of the results and help disambiguating. Figure 3.11 shows an example of the number of information for each person's main features, some knowledge base's information weights and the selected fields to search and context to the next iteration. Resorting to a flag the plugins send upon the previous iteration, the Coordinator can also avoid some plugins that most certainly will not provide new and relevant information.

Although the previous iteration checking may select the main fields to be searched, when a lot of information is found (in all fields), it does not mean that there is not more information useful in our sources of information (that can be improved using appropriate context). To definitely stop iterating, or to keep iterating even if our main fields are well supplied, we consider the number of new values, relating to the number of old values obtained in this iteration (Figure 3.12). The number of new values has to be bigger than $\frac{1}{4}$ the old ones to keep searching. If it is smaller, there are only a few new values, so it is most probable that the

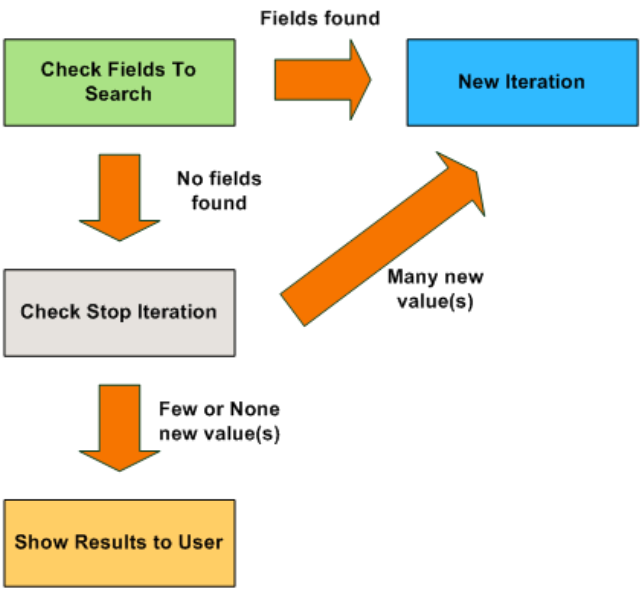

Figure 3.12: Iteration process decision-making

next iterations will converge to zero.
For instance, if in the current iteration the algorithm finds five new values and one hundred old ones, it is not worth to continue iterating, but if instead we have the same five new values and ten old ones, maybe we can still get new information on the next iteration. Hence, the ultimate decision is based on the ratio between old and new information, one that verifies if the search is converging. The value deciding that ratio can be easily changed.

When this process is terminated, we need to decide which information we present to the user . Since we have all the data weighted, we can establish a limit and the one with a bigger value can be presented to the user. In the next section, we present an example application which uses 0.5 for limit (from 0 to 1), but indicates the confidence to the user.

3.4. Updating the Knowledge

The coordinator feeds the Knowledge Base with information to be stored in the semantic network. Like previously mentioned in its section, the Knowledge Base module stores the information based on a triple "subject-predicate-object", with some adjusts in case frames that need additional data. Each plugin assigns a different weight to the information it extracts and the coordinator recognizes different degrees of credibility for each plugin. Also, different plugins might provide the same information about a concept, thus accumulating evidence of its truthfulness, while in other cases opposing information might result. To account for this, the semantic network allows the different relationships between concepts to be weighted, as an indicator of their credibility and at the same time associate their source(s) of information(s).

It is important to notice that the same information may be retrieved from different plug-ins. It is likely that the duplicated information is relevant. However, it is not trivial to reinforce the information mostly due to the diversity of forms it can be presented. We want similar values to be marked as equals, so when both appear instead of having two different values, the weight is recalculated. A good example is someone that has in his Facebook interests "Machines", and in his blog says to be interested in "Machinery". We want both to be the same, and to accomplish that we stem the information before inserting it in the Knowledge Base. Stemming gets the root of the word, so when we have similar words they count as the same. When we want to show the information to the user, we reconvert the Stemmed word in the smaller word originating that Stemmed one (ex: "machine"). We present a similar example on Figure 3.13. Also, related to the words associated to our main characteristics, we have equivalent words that are replaced so they count as the same. For example, "like", "adore", "love", "care" and "passion" are replaced by "interests", but can easily be extended.

When some information chunk, already indexed in the knowledge base, presents itself again, the confidence on that piece of data is reinforced. However, it is important to notice that the different sources and different types of information are already weighted differently (plugin weight * triple weight). We developed an algorithm that respects the value of the information and maintains a normalized weight scale.

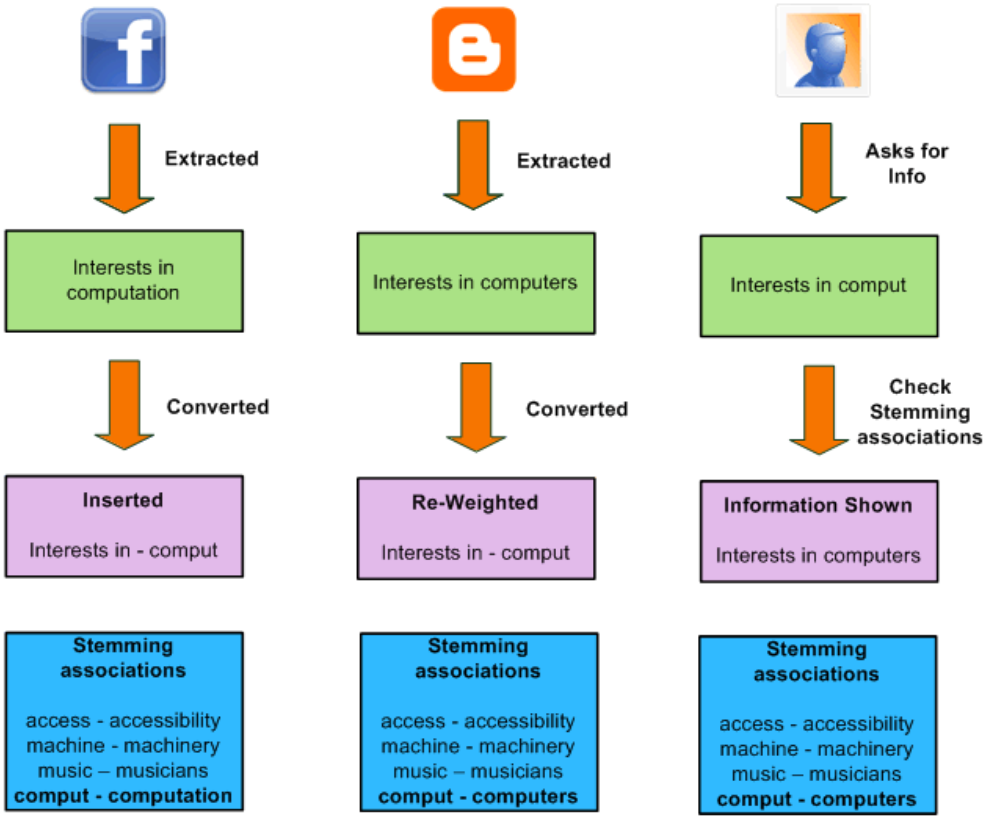

Figure 3.13: Usage of stemming for duplicate information and its presentation

Considering that a relation is previously weighted with 0.8 (in a scale from 0 to 1). A duplicated entry is detected with a weight of 0.5. The weights are recalculated as follows:

1. The relation is to be reinforced. To this end, the initial 0.8 are guaranteed and we are only working with the remaining weight percentage

$$\textbf{1 - OldWeight [0.8] = 0.2}$$

2. We use the new weight (0.5 in the example) as a reference value to scale the remaining weight (0.2)

$$\textbf{AddWeight = 0.2 * ArrivingWeight[0.5] = 0.1}$$

3. The calculated value is added to the old value.

$$\textbf{NewWeight = OldWeight + AddWeight = 0.9}$$

The new weight on the Knowledge Base would now be 0.9. This algorithm allows us to always reinforce the information when similar information arrives, but in a moderate and consistent percentage. Since the old weight was 0.8 and the information appears again, we always want the new weight to be bigger than the old one. So, we use the remaining 0.2 to help calculating the new value and add it to the 0.8.

When the new information is added or the new weight calculated, there is a counter keeping the register of the number of new and old (duplicated) pieces of information. This helps the Coordinator in its iteration decision-making, by being aware of the old information related to the new one. If there is much more old information than new, there is no need to continue interating, because it is converging to zero.

To keep recycling and refreshing the information we use timestamps, so that older data starts losing its weight. It is important because that information, someday correct, is now incorrect or out of date. The older the information is more weight it loses. If the information is still recent (1 month or less) the weight stills the same, and being the information found on the same plugins, it does not change the current weight. However, if the information is old enough (more than one month), a new weight is calculated to decrease its value. The expression used to find the value to decrease is based on months (between 1 and 2 months count as 2; between 2 and 3 count as 3, and so on). The maximum value is 12, so information with 12 months or more have the same value (12). The expression is:

$$\textit{Value to Decrease} = \ln((\textit{months}/10) + 1)$$

Since this expression possible month values are between 2 and 12, the values to decrease are between 0.18 and 0.79 approximately. The old weight minus the decreasing value, results on our new weight (cannot be less than 0). An example, a tuple with 5 months old and a weight of 0.8, using this expression, is now near 0.4. If that information appears on the next search, which means it is still relevant, so it will increase its weight again. If it does not appear, it means that the information might be incorrect or obsolescent, so it maintains the lower weight.

In this section we presented our approach, some possible scenarios and the main challenges involving it. We explained the different components of our architecture and described the specific tasks for each one of them and their role in the whole architecture. Also, we detailed our solutions and algorithms for the most important challenges. In the next section, we will present an example application that uses our framework, in order to prove and evaluate our assumptions.

4
Example Application

To prove that our framework can provide relevant information to the user and satisfactory results, we developed an example application with the goal of obtaining information about a person (Figure 4.1). The main concern of our interface is to turn the process easy to interact with, centering it on a simple search task, but at the same time make use of all the features involving our framework. That includes the personal and public information, including its filtering, the iteration process, the data from different sources and structures (also non-structured) being represented as a coherent and inter-related whole, the decision of what to present to the user and all it brings.

This application resorts to four different types of information sources: personal information existent on the user devices (e-mails); social networks (facebook); blogs (BlogSpot) and Wikipedia. We limited the sources in this proof-of-concept application to these sources as they represent different types of information, enough to prove our assumptions and at the same time keep the system evaluation simple and quicker. The users' devices (in this case, their personal computers), provide us with e-mails, which are rich interaction-wise and are likely to be useful for the user and at the same time are helpful in filtering information from public online sources (using the information therein). As a representative social network we selected facebook as it is widely spread worldwide and particularly within the portuguese society. Also, it is the most used and preferred by those who tested our application. BlogSpot/Blogger is also a very popular blog platform and Wikipedia has descriptions of many people, mainly famous or important in their areas. In this application context, we have to support searches for people with different relationships with the users. It can be a very close friend, some person they know but have not much information or even a celebrity or a very important person. We chose these information sources with the intent to cover all these scenarios. Scribe and Facebook are more intimate, so apply mostly to close friends, but there is also a big probability to contain information about a "known" person. On the

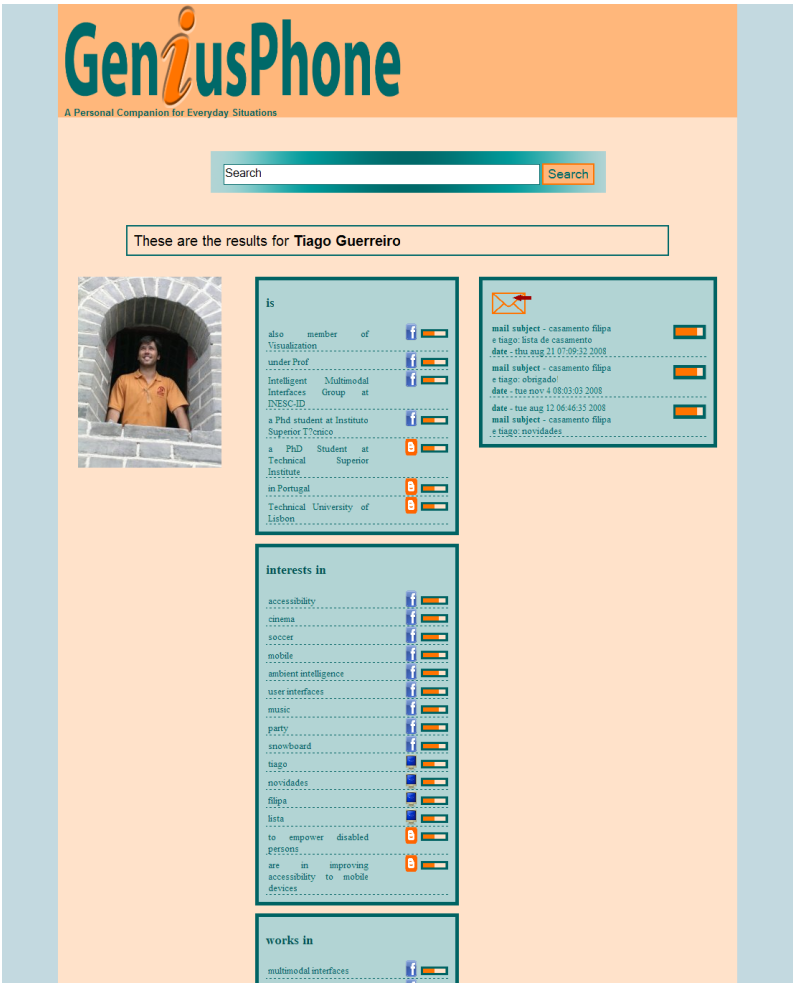

Figure 4.1: GeniusPhone example application - general presentation of results

other side, Wikipedia only contains information about celebrities or very important persons, and BlogSpot is more transversal and can provide information about either one of these scenarios.

4.1. Scenarios

To better understand the utility of this application and information sources, we outline three meaningful scenarios:

Tony's birthday is on Friday and he is a big friend. I am trying to find a present for him, but have no good ideas. I search in GeniusPhone for Tony Smith. It finds the last mails we shared and after seeing the subjects of two of them ("Top10 dunks" and "beginning in web designing") I remembered that he loves basketball and is learning web designing. I had the idea of a ticket to a Basketball game, but I'm not sure about the team he supports. Also, I thought about giving him some software that may help in his web designing challenges. I notice that some information was found from his facebook profile. I realize that his interest in Basketball is based on LeBron James, from Cleveland Cavaliers, so I bought 2 tickets for the next game.

I am at a party and I find someone that seems familiar talking to a friend of mine. I caught his name, Joseph

Ivey, and ask the system for information about him. It shows that we exchanged some e-mails two years ago about computer graphics and the CHI conference, and recall immediately who he is. At the same time, I see some information that was extracted from his blog, saying he is now interested on Personal Information Management and Information Retrieval, and his hobbies are Soccer and Golf, which I had no idea.

I am at a business meeting in Paris and someone starts talking about Frédéric Mitterrand. I have no idea who he is but keep listening to catch something. I get very confused because they talk about politics and movies, books and much more. So, to avoid looking bad if they ask me something, I ask the system for information about him. By searching on Wikipedia it informs me that he is the current French Minister of Culture and Communications, so I understand all that talking about politics. Also, it mentions that he was an actor, a television presenter and a writer. Now that I know who he is, I can participate on the conversation, even not knowing everything.

4.2. Application Platform

There are many contexts where GeniusPhone can be used. Using mobile devices as an entry point to GeniusPhone framework it is possible to use it almost everywhere. Besides that, it is perfectly possible that users want/need to use it in the comfort of their homes or offices. For example, if someone calls their home telephone and they are near their Laptop, it is quicker and very useful to access the application there. We found an online service accessible through a webpage as the best option to deploy GeniusPhone applications. This solution makes GeniusPhone accessible from every device. The only limitation for this solution is the obligation to always be connected to the Internet, but this application nature makes this an obligation to all possible solutions. Also, it does not need previous installation or configurations to perform a quick search.

4.3. Interface

Figure 4.2: GeniusPhone search Interface

The interface for this application is very simple and focused on the search task. To search for somebody it is only necessary to write the person's name and click enter/search (figure 4.2). Our main focus is evaluating the system performance and we do not want a complicated interface. We also tried to present the results in a simple and intuitive way and at the same time allow to

keep searching for someone else. So, the search part stays equal and at the same location, and the results are presented below. A general view of a search results is presented on figure 4.1.

The results page is divided in three columns, one for the photo, other for general information and the last for interactions (in this case, only e-mails). Like figure 4.1 shows, the left column shows the picture of the searched person, in this case retrieved from facebook. A picture can be very important, for example, when users hear some person talking about other one that they have no idea who he is. With a picture users can easily recall (if they have already seen him). Basically, by presenting a picture we are giving a face to that person. Also, we show only one picture because our main concern is to have the idea of who that person is. It is not necessary to have an enormous amount of pictures. However, sometimes it is not possible to get any picture, but to always maintain the presentation structure a person icon is shown at the same location (figure 4.3).

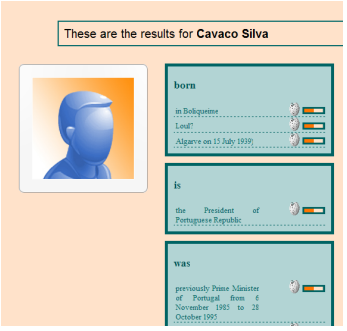

Figure 4.3: Icon replacing picture

Figure 4.4: GeniusPhone result's interface middle column, which presents the general information

On the middle column we present general information. This is all the information extracted that do not include photos or personal interactions. We thought it would be better separate interactions and put it on the right column for better organization. Figure 4.5 shows some examples of the middle column, where we have the information separated by relations ("is", "interests", etc), and a list of results for each one of them. Each item has a bar indicating the confidence GeniusPhone has on that piece of information (highlighted on the first item of the left image).

We only show results with a confidence superior to 0.5 (from 0 to 1), because less than that is

information not credible. Also, each item has a clickable icon indicating its information source, and opening the concrete webpage from where it came from, providing the information context and access to more detailed information, if necessary. Figure 4.5 shows two examples clicking on a BlogSpot and Wikipedia icons. Facebook icon is also clickable and opens the person's Facebook profile.

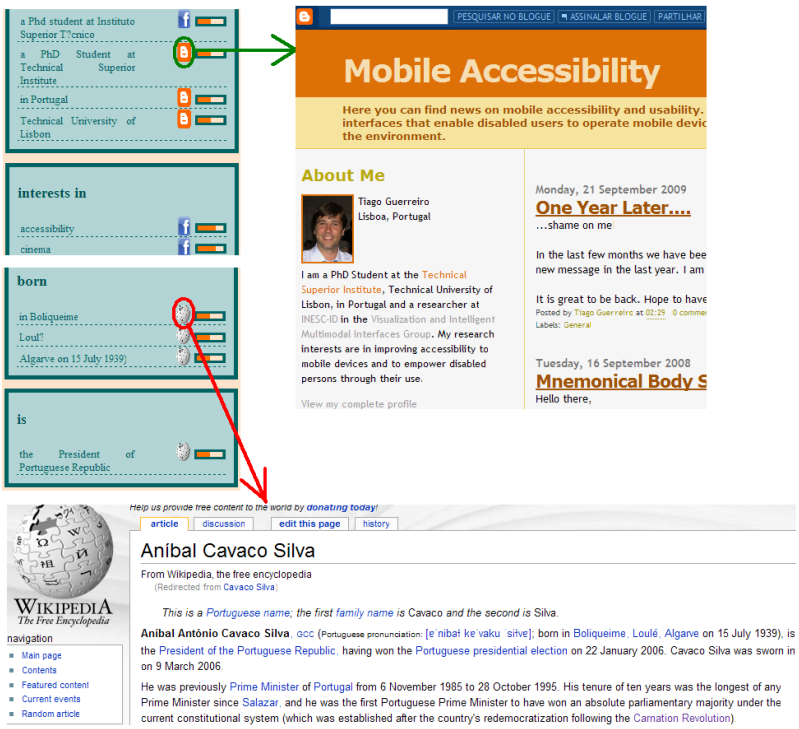

Figure 4.5: Clickable icons and the resulting webpages

Figure 4.6 shows the right column (interactions), showing an example of the received e-mails and its concrete characteristics (subject and date, and if there is an attachment, its name). The type of interaction is defined by an icon above the items corresponding to that type. In this case, it means that the interaction was by mail, and it was received. To represent sent e-mails, the arrow points to the other direction and the color is green. Each mail, in addition to characteristics, has also a bar indicating the confidence GeniusPhone has on that piece of information. It is not necessary to assign a weight for each one of those characteristics because all the e-mail information have the same confidence.

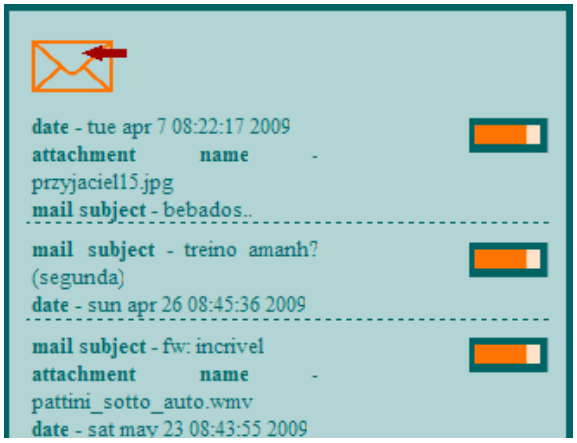

Figure 4.6: GeniusPhone result's interface right column, which presents the interactions

In this section we described an example application which responds to searches about persons. It was built to prove our assumptions and analyze our framework behaviour and results. In the next section we will detail the users' evaluation of our framework (using this application), show

the results and analyze them.

5 Evaluation

A state of the art analysis (Chapter 2) revealed several fragilities respecting to the use of personal information existent on users' devices. Although we found and have described in this document many potentialities to improve users' experiences, using it together with public online sources, our scenarios and contributions are not worthy until they have been evaluated with real users and real information. Thus in order to confirm our assumptions and show that our approach is able to provide useful and relevant responses from the user perspective, further evaluation is required.

To validate our approach, we developed an example application (Chapter 4) that uses our framework to provide user-relevant resumed information about a person. This application enabled us to validate our values, limitations and contributions. We have conducted task-oriented experiments and extensively collected results for further analysis. It is relevant to mention that, to classify the results, we had the users' collaboration, because the results are about them and they know better if it is wrong or right, and what is missing.

Our framework can be used in multiple scenarios. Even considering that it has wider potentialities in a mobile context, its browser interface allowed us to evaluate the application using desktop computers as entry points, since it is more difficult to find users with compatible operative systems. Thus, as a proof-of-concept we have evaluated our approach using a desktop version and a common web browsing interface. The main focus of this evaluation is to validate our contributions and assess its values in respect to information coherency and inter-relation. Moreover, we intend to verify if the information provided is relevant in the user's point of view.

To acknowledge our approach as a success, some research questions need to be answered at the end of our evaluation:

Is it possible to inter-relate information from different sources? There are multiple information sources that can provide knowledge to the user. Analyzing the state-of-the-art we noticed that most existent applications only deal with one type of information, and those who deal with more do not establish any relations between them. Using more than one source, more types of information and detail levels are accessible, and inter-relating them, it is possible to have a general and coherent view of all the information. On our example application, we use four information sources, from different types, and our goal is to store the data retrieved from all of them, providing knowledge, consistently and inter-relating it.

Can we provide useful and relevant results from the user perspective? To be considered a success, this framework has to provide satisfactory results to the users, improving their experience and enlightening them when in doubt or in need for extra information. It is important that the results are influenced by the users' perspective, returning the information that is really relevant to them. Essential to this point is the personal information, on the example application, the mail messages. On this context, it is our intention that the exchanged mails can provide information of the common subjects and guide the results based on them.

Is the system suitable for different scenarios and contexts? The framework was developed with the intention to be useful in several scenarios, either to provide information about persons, events, documents, institutions or subjects. Also, its browser interface allows its use on a mobile or stationary context. Our example application evaluation tasks, though performing searches only about persons on a stationary context, concern to different scenarios. This can be validated by searches of people with different kinds of relationships and analyzing their results separately.

Is it possible to obtain public information, filtering it with personal information? Public online sources can provide a huge amount of information and, without auxiliary help, it is very difficult to find good results. Using personal information as a seed, we try to disambiguate conflicting data when in doubt of which one is correct. Also, when no information is found on some online source, we try to guide searches based on the most credible information, providing results from the user point of view.

5.1. Procedure

The evaluation procedure is divided in two different phases: preparation and execution. Preparation steps include introducing the user to the evaluation, indexing personal information available in the user's computers or online services (webmail) as well as parametrizing other services (like Facebook) to enable users' access to their friends' profiles (using the provided APIs). We present a detailed test preparation script on appendix A1.1, but summarise it here:

- Prepare the users' Facebook accounts, so they can use its API, to turn possible their access to friends' facebook profiles
- Copy the Scribe (personal information plugin) executable to the users' computers, in order to index the personal information therein (i.e., mail messages)

- Parametrize the configuration files
- Run Scribe, indexing the personal information existent on their device
- Send the index folder and other important files generated by Scribe to the evaluation computer
- Change the facebook API parameters in GeniusPhone configuration file, to associate the application created to the user account
- Verify Internet Connection, to take advantage of all information sources.

With all steps completed, all the ground work is set for evaluation! First, we need to have some idea of who the users are. We handed them a questionnaire (appendix A2.1), where they answer some questions related to their age, habilitations, interests or working subjects. These questions can enlighten us about the users' preferences and, with further analysis, check if they are relevantly related with the obtained results. Also, we asked some information about the usage of their personal devices to check if there are differences on results based on that.

The evaluation itself is composed by six (6) searches performed by the users. We asked the users to choose 6 individuals with different relation magnitudes (a public figure vs a good friend or a relative). Besides wanting to evaluate the results' trustworthiness, we also wanted to evaluate how the results match the user's expectations. Thus, for each search/person, we asked them to write, on paper, the information they were expecting to get from each one of them, the relevant information about that person from their point of view. Then, the users perform their tasks and, upon completion, answer a final questionnaire validating the results, evaluating the application by rating several system features with a 5-point Likert Scale and offering subjective feedback (Appendix A3.1). The detailed procedure of the evaluation execution is on the appendix A1.2.

5.2. Users

To evaluate GeniusPhone, and the underlying framework and approach, we performed the evaluation with fourteen (14) users, ten (10) males and four (4) females, with ages comprehended between 22 and 57 years (averaging 28 years old). Most of them are university MsC students (71%), but we also tested it with a few high school teachers and a medical doctor. This selection was based on finding people with multiple uses of mobile and stationary devices but also because this target group was easier to reach. The target group presented different hobbies and interest although some know each other and are somehow related. However, it was enough to guarantee a user group with different Internet usages and personal information patterns. Table 5.1 shows the users' characterization and their personal devices usage.

To get more meaning and truthfulness to the results, we asked a few questions to the users before they made the test to realize what type of users they were and catch their hobbies and interests. The questionnaire and results are on the appendix A2.

In addition, we collected the information about the user environment and conditions (mails indexed and facebook), to check if those matters influence the results. It is natural that the results

	User Characteristics			Devices Information				
User	**Sex**	**Age**	**Habilitations**	**Computer**	**Computer uses**	**Cell Phone**	**PDA**	**Mobile Devices Uses**
LE	F	23	BsC	Yes	Work; mail; Internet	Yes	Yes	Calls; SMS; mail
CG	M	55	BsC - Current Teacher	Yes	Work; mail; Internet; Social Net.	Yes	No	Calls; SMS
FB	F	29	BsC - Current Doctor	Yes	Work; mail; Internet	Yes	Yes	Calls; SMS; work
AP	M	22	BsC	Yes	Work; mail; Internet; Social Net.; chat	Yes	No	Calls; SMS
AC	M	22	BsC	Yes	Work; mail; Internet; Social Net.; chat; Games	Yes	No	Calls; SMS
JS	M	28	BsC	Yes	Work; mail; Internet; chat	Yes	No	Calls; SMS
MM	M	23	12th Grade - Finishing BsC	Yes	Work; mail; Internet; Social Net.; chat	Yes	No	Calls; SMS; MMS
CA	F	23	BsC	Yes	Work; mail; Internet; chat; watch Tv Series	Yes	No	Calls; SMS
RM	M	23	BsC	Yes	Work; mail; Internet; Social Net.	Yes	No	Calls; SMS
JC	F	23	BsC	Yes	Work; mail; Internet; Social Net.; chat; Games	Yes	No	Calls, SMS, GPS; Internet
JA	M	22	BsC	Yes	Work; mail; Internet; Social Net.; chat; Games	Yes	Yes	GPS; SMS
AF	M	22	BsC	Yes	Work; mail	Yes	No	Calls; SMS
AF2	M	23	BsC	Yes	Mail; chat; Social Net, Internet	Yes	No	Calls; SMS
LG	F	57	BsC - Current Teacher	Yes	Mail; chat; Social Net	Yes	No	Calls; SMS

Table 5.1: Users' characterization and personal devices usage

are influenced by the number of mail messages and the existence or not of a Facebook account. So, 86% (12/14) of users have facebook and figure 5.1 presents the range of indexed mails per person. The most frequent is between 251 and 500, but there are also users with a huge quantity of indexed mails. In order to evaluate only in reasonable cases, we restricted the users to persons that have indexed e-mails (the minimum was 40), considering the results would suffer substantially

without that and our work is based on the usage of personal information.

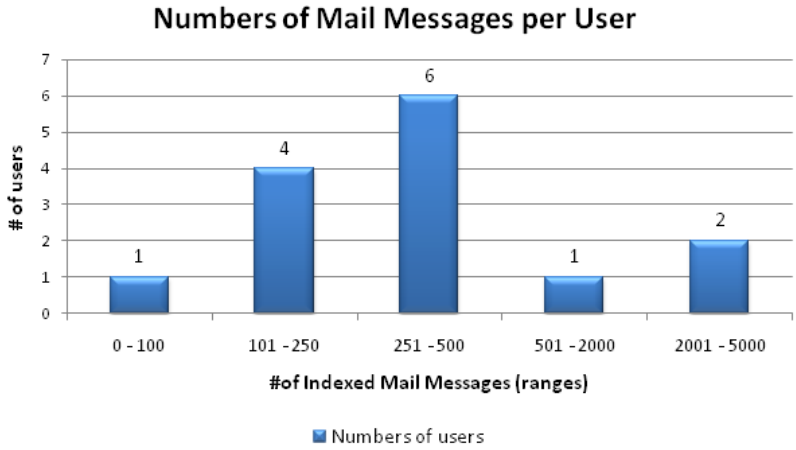

Figure 5.1: Users' range of indexed mail messages

5.3. Evaluation Tasks

Users had to search for some persons in order to obtain information about them. These searches were divided in three types, which differ in the different kinds of relationship the users have with the searched ones. In order to obtain and analyze more information, we asked the users to search for two persons of each type, which mean six searches for each user:

5.3.1. Search for Close Friends

Users had to search for a person they know very well (a close friend or familiar). Being the users the persons who best know who their closest friends are, they are the ones choosing them. This is the most personal search, so it was difficult and it was not necessary that the most users share the person to search (However, when there was a chance we used it).

On this task, it is possible to explore the existence of many shared information (e-mails). Also, with more personal information it is possible to better test the filtering and guidance of public sources. At the same time, users' expectations should be bigger because they know much information about them.

Possible scenarios for a search for a close friend are to find out what were the subjects users had to discuss with their friends, or trying to find a present that suits them for their birthday.

5.3.2. Search for "Known" Persons

Users had to search for someone they know, maybe have had some interaction, but do not know much about. Some examples are a Teacher and a brother or close friend of a friend. In this task,

when different users share known persons, we make use of that, asking them to search them. Thus, we can compare results in order to analyze if there were any differences and why they occurred.

We can explore if different users' interests and personal information generate different results. In this manner it is possible to analyze our success on filtering and guiding searches on public sources. Also, we can compare users' different expectations and their concretization or not. However, if users' had no previous interaction (exchanged e-mails) with that person, the results are handed to public sources.

This task contains the most probable scenario in GeniusPhone, where the users want to know information about someone they recognize but do not know from. "Where I know this guy from?" or "I recognize this name, who is he?".

5.3.3. Search for Famous Persons

Users searched for someone known by everyone. This can be a very important person or a celebrity, and it is important that users do not personally know them. This feature allows us to test public sources without intervention of personal information. Also, we can analyze users' expectations, and satisfaction with results, according to their interests.

For this task, we pre-selected some personalities that every Portuguese citizen knows. We chose Cristiano Ronaldo (a famous soccer player), Cavaco Silva (Portuguese President of Republic) and Herman José (famous Portuguese comedian).

This can be a real scenario, for example, in a different country, when users are in a conversation and somebody refers a name they do not recognize, but assume they should know. If some user goes to France, and somebody refers the finances minister name, I would like to know who he is and have something resumed about him to avoid looking bad.

5.4. Results

To prove our assumptions and answer our research questions, after the users have performed each task, we collected the data that we found helpful to evaluate our system (appendix A1.3). Some of that data was automatically collected (total/per plugin information found and shown), but the remaining one deserved a more careful analysis based on users' opinions, approval and the information they were expecting to get from each search. Appendix A4 presents all the data extracted from each search and user.

As our approach tries to provide relevant results from the users' perspective, those results had to be analyzed by them. They are the ones knowing which information is relevant, irrelevant or garbage for them. Also, we needed help from them to quantify the information they were expecting to get but was not shown. It is important to know if that information was impossible to get, or our framework missed it.

In the next sections we make a detailed analysis of the most important features of our approach trying to answer our research questions.

5.4.1. Evaluating Users' Expectations and Results Relevance

Before performing our tasks, and after selecting the persons to search, the users wrote on a paper the information they were expecting to get from each person. This information is based on features they know about those persons and believe to define them or are somehow related to them. It is important to notice that we have not restricted this process and are not aware, before the experiment, if the information is correct or available in any of the searched sources. This data is very important so we can analyze our results accordingly to user expectations. How were our results accordingly to the users' expectations? To answer this question we have to analyze some different aspects. First, which information from users' expectations was, and was not presented? Figure 5.2 shows the results, in percentage, for the three (3) different search types.

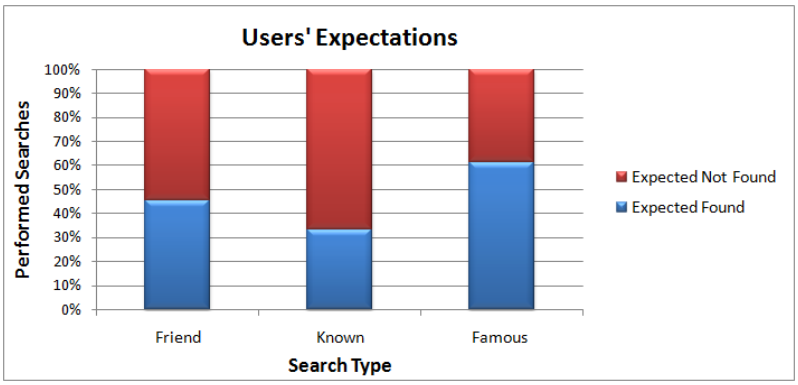

Figure 5.2: Users' expectations and achieved results

From this chart, we can observe that regarding a Friend or Known person search type the values are below 50%, as to Famous people the results are near 60%. Comparing Friends to Known, having more interaction with close friends it is understandable that it will improve the results, giving them a little advantage. On the other side, Famous searches have more available information on the Internet, and on the searches we used, there was always information to be found on our information sources.

Although these results are not outstanding, on most scenarios, mainly respecting to Known and Famous people, some relevant information is enough to enlighten and help the users through their difficulties. Avoid looking bad, or unblock some conversations are some scenarios where having some relevant information can be very useful. These situations might happen even more on a mobile context, where users' may need some quick information to prepare approaching to some person. Besides, it is important to remember that the user's expectations were not preprocessed in any way. Also, we can observe on Figure 5.3 that, from the information that was expected by the users, but was not found, on Friend and Known search types, most was impossible to get through our information sources (near 90% and 85%, respectively).This information was verified with the users in a post-test analysis, by asking them which were the subjects of their exchanged mails or even check the searched person's Facebook profile. This indicates that from

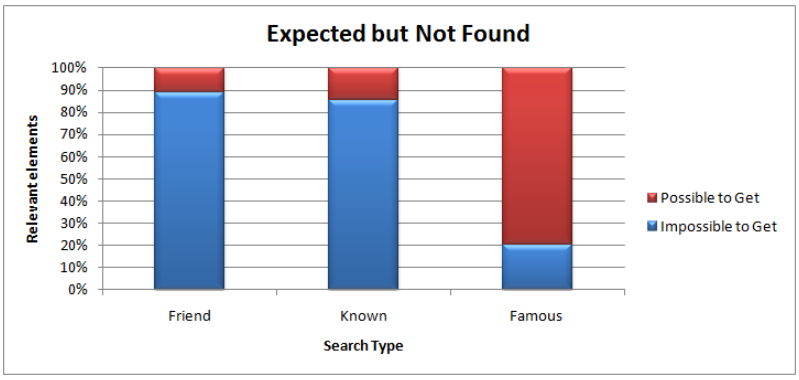

Figure 5.3: Information that was possible or impossible to retrieve

the information users were expecting, only a few was possible for us to retrieve but somehow we missed it (Figure 5.4). Thus, for the personal and public sources considered our framework is able to retrieve the majority of the expected and possible to get information. On the other side, the Famous searches, which gave the best "expected results", had potentialities to almost complete the users' expectations with only near 20% of impossible information.

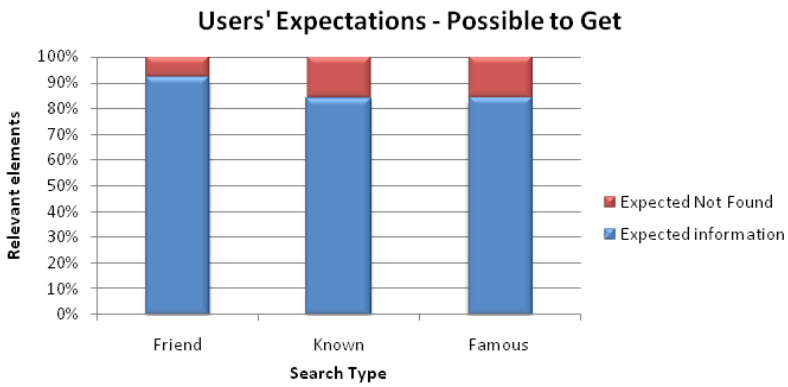

Figure 5.4: Percentage Chart with users' expectations results, considering only the information possible to get

On the first chart (Figure 5.2), we can see the results according to user expectations, but that do not represent all the information relevant presented by our application. Figure 5.5 presents the total number of relevant elements retrieved, including the expected and found information and also information not expected but relevant (average values). This chart suggests that the resulting page presents more relevant information not expected than the number of our total expectations. Although users were not expecting this data, they tagged it as relevant in a post-test analysis and found it to be useful. Also, our approach fits on scenarios that the information users need is the one they do not remember at all, so they could not be expecting it. These results suggest that, mainly respecting to Friend and Known searches, although not presenting all the expected information, the results exceed the expectations and show important/relevant information that the user did not remember.

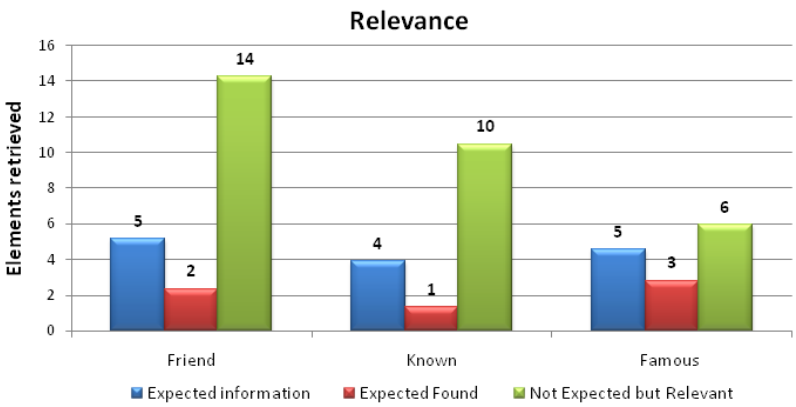

Figure 5.5: Relevant information shown to the user

5.4.2. Analyzing Information Sources

At the same time users were performing tasks, the number of information retrieved and shown per plugin was automatically collected. In a post-test analysis, the users classified the information as right/relevant, wrong or garbage/irrelevant. This classification allows us to assign the quality of each plugin and understand what should be done better. In figure 5.6 we can observe the results, in percentage, for each plugin. On Facebook results are very satisfactory, with more than 90% right/relevant information on Friend and Known searches. The only wrong values on this plugin (near 1.5% on Friend searches) were because those pieces of information were out-of-date, something that is impossible to verify. In Famous searches, since there are no famous friends/"known", there is no retrieved information, which means no analysis to be made.

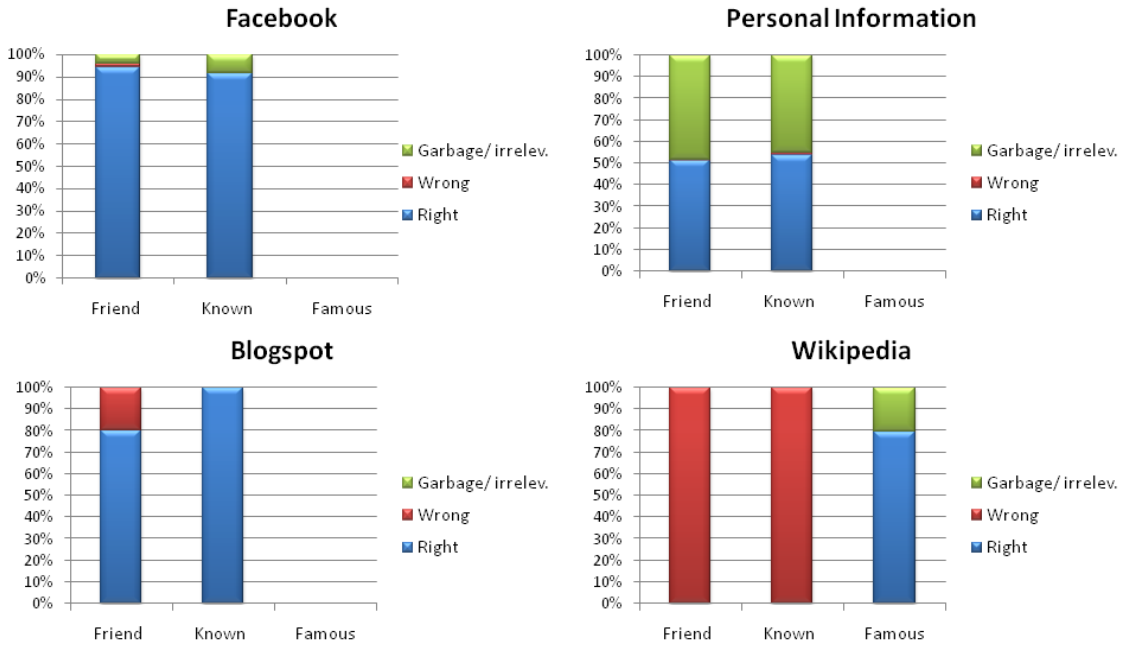

Figure 5.6: Quality results for the four plugins

The personal information plugin (Scribe) results were below our expectations, however we are glad to identify the problem and find a solution to solve it for future work. We can observe that

Friend searches have a few more garbage than Known ones. The main reason is the (over)weight we assign to words on mail subjects. Persons with more mail messages (Friend searches) have more subjects and consequently more unmeaningful keywords. The solution to this problem is to assign a smaller weight to those words, not allowing them to be presented unless some other source corroborates. Again, for Famous searches there are no results.

BlogSpot results were very good, mainly respecting to Known searches. The difference between these and Friend searches (near 20% wrong), is that some blogs were found that did not belong to the searched person, presenting wrong information. Respecting to Famous searches there are no results.

Wikipedia chart seems terrible for Friend and Known searches, however it is not a plugin directed to those searches. That wrong information (100%) occurred when some friend's (or "known") name was the same of a famous person. That happened only one time on a Friend search and other one on a Known search. For Famous searches, this plugin's focus on this application, the results were satisfactory, with near 80% right information. The near 20% garbage is related to some flaws on the Natural Language Module that needs to be improved.

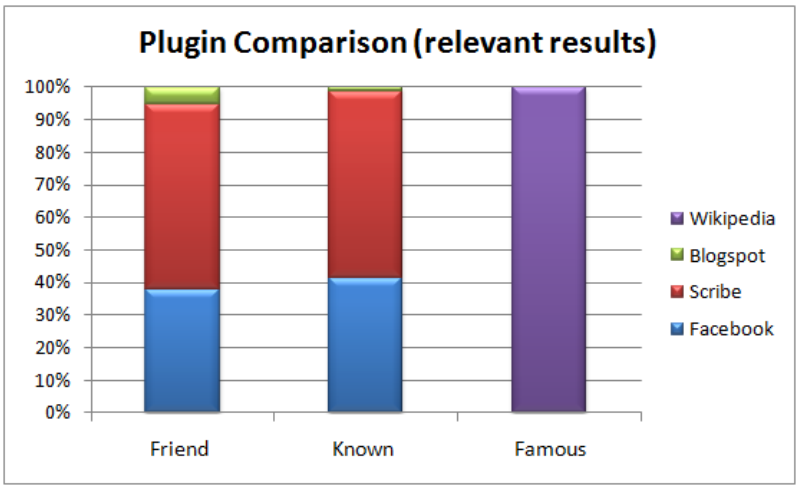

Figure 5.7: Plugins influence on the number of relevant results

Although these results can suggest the plugins' success rate, it is not enough to analyze their relevant information contribution. Figure 5.7 shows that blogs have a minor contribution to the total results, though it has a good success rate. Friend and Known searches are dominated by information from Scribe and Facebook, with a bigger predominance to the first (near 55%). Although its success rate is suggested as average, it provides a lot of useful information to the user. On Famous searches, the relevant results belong totally to Wikipedia. This is perfectly understandable since they did not have any interactions with them. Also, none of the famous persons selected own a blog, so it cannot provide information about them.

5.4.3. Search types

As seen on previous charts, results are different considering the type of search performed. Searches performed for a Famous person present better results when compared to the users' expecta-

tions, but on the other hand, present less relevant information that was not expected. Indeed, the expected information and retrieved one are very similar in this scenario. Also, 80% of the expectations not satisfied were available on the information source (wikipedia) but our plugin missed them. It is comprehensible that all the correct information of these searches is retrieved by Wikipedia.

Friend and Known searches' results suggest many similarities between these two groups. However, it is relevant to mention that Friend searches normally present better results. This search behaved better relating to expected information and presents more relevant information outside the expected range. Also, considering the information not found that was expected by the users, most of the items (a greater percentage in Friends than in Known searches) were not possible to retrieve as they were not available (as verified with the users by post-checking the information sources). This combination of results suggests that when more interaction happens, and consequently more personal information, the results are better and the missing information is most likely impossible to get.

5.4.4. Public information filtering

Since some users had friends or "known persons" in common, we used that to compare their searches and check if their personal information influenced the obtaining of public information, guiding the searches. Table 5.2 shows eight different searches for the same person made by different users. We can observe that those which retrieved most personal information and/or data from social network profiles could guide their searches and obtain some relevant information from blogs. These results could only be obtained using two (2) iterations, since on the first one, for all these searches, nothing was found on blogs. Those which could get relevant personal information used it as context to help filtering on other sources. The result was additional relevant information, which could not be obtained by the users that had no interactions or connections (by social networks) with them. On Famous searches, since there is no personal information involved, this is not applied.

	General Info shown					From Facebook				From Scribe				From BlogSpot			
Type of search	Total info	Right	Wrong	Garbage/ irrelev.	Useful Iterations	Total info	Right	Wrong	Garbage/ irrelev.	Total info	Right	Wrong	Garbage/ irrelev.	Total info	Right	Wrong	Garbage/ irrelev.
friend	74	54	0	20	2	29	27	0	2	37	19	0	18	8	8	0	0
friend	43	20	0	23	2	0	0	0	0	35	12	0	23	8	8	0	0
"known"	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0
"known"	32	12	0	20	2	0	0	0	0	24	4	0	20	8	8	0	0
"known"	51	49	0	2	2	17	17	0	0	27	25	0	2	7	7	0	0
"known"	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0
"known"	61	35	4	22	2	27	27	0	0	26	0	4	22	8	8	0	0
"known"	48	39	0	9	2	29	27	0	2	11	4	0	7	8	8	0	0

Table 5.2: Information retrieved per plugin on searches for the same person, made by different users

5.4.5. Users' Opinions

To quantify the users' satisfaction we chose some main features to be evaluated using a Likert Scale, with values varying from 1 to 5 (1- Strongly disagree; 2- Disagree; 3- Neither agree nor disagree; 4- agree; 5- Strongly Agree). The table 5.3 presents some values for each question (average; standard deviation; mode; inter-quartile range).

	Easy Interface	Atractive WebPage	Provide useful information	Understandable information	Provide expected information	Relevant from user perspective	Source and context perceptive	Wide group of results	Presents the best results
Average	4,7	4,1	4,1	4,0	3,6	4,1	4,1	3,9	3,3
Stdev	0,5	0,8	0,5	0,9	0,6	0,4	0,8	0,9	0,6
Mode	5,0	4,0	4,0	5,0	4,0	4,0	4,0	3,0	3,0
Inter-Quartile Range	0,8	1,0	0,0	2,0	1,0	0,0	1,8	1,8	1,0

Table 5.3: Users' satisfaction ratings, using a 5-point Likert Scale

Although the interface was not our main concern, we are pleased to notice that users found it easy to interact with and attractive (average of 4.7 e 4.1 on a 5-point Likert scale, respectively). More important are the classifications to the usefulness, understandability and relevance of the results, which were rated with average values between 4 and 4.1. These ratings suggest that the users were satisfied and found our application helpful. The lower classification belongs to "provide the best results", which average value is 3.3. However, we recognize that this is a wide area, with many possibilities to explore, and this value, will certainly be improved in our future work.

At the end, we asked some feedback and got some suggestions to improve our framework. The most referred were to improve the keyword filtering mechanism that we have already mentioned, and to add more information sources to have a larger range of information types. Although, we noticed that they were satisfied with our application and most of our searches presented relevant results (Figure 5.8).

Figure 5.8: Chart indicating the percentage of searches which gave results and those who failed

5.5. Discussion

With the results achieved, we are now able to discuss the approach success. In this section, we analyze the results taking into account the four research questions referred at the beginning of this chapter: information inter-relation, relevant results from user perspective, suitability for different contexts and personal information as a filter for public information.

Is it possible to inter-relate information from different sources? Like we observed on almost every chart, except for Famous searches, the results are a combination of information from different sources. Scribe and Facebook are more predominant but that is related to the fact that almost everyone had Facebook and exchanged e-mails with the users. On the other side, almost every person did not have a blog. On Famous searches there is only one useful information source, so there is not a mix of different types of information. However, if there was a friend that was also famous, the inter-relation would be noticed as well.

Figure 5.9: Example of different sources reinforcing the same information

During the users' execution of their tasks it was possible to realize that, besides presenting the information as a whole it was also managed as a whole. This was possible to observe, because the same information, even from different sources, was treated and shown as the same. Figure 5.9 presents an example. Also, this inter-connection allows the information to be reinforced and gain preponderance as a result and as context for further iterations.

Can we provide useful and relevant results from the user point of view? We can observe on figure 5.8 that there were only a few searches that failed (near 10% on Known and near 3% for Friend). Also, analyzing the chart on figure 5.5, it suggests that in average the searches provided great numbers of relevant information. Although some expected information was not shown to the users', for the most cases (on Friend and Known searches) it was impossible to obtain (figure 5.3). Also, the suggestion that it presents more information that was not expected adds great value to our results, since for most scenarios the information users do not remember is the most useful for them.

Analyzing users' satisfaction (table 5.3), we noticed that they classified that our application presents useful and relevant information from the user perspective. Both averages are on 4.1, standard deviation values are small and the inter-quartile range is zero (0) for both. These are good values that suggest that no one classified this features badly.

Is the system suitable for different scenarios and contexts? On every chart we separate the searches

based on the different scenarios to analyze if our framework is suitable for all of them. We can observe that besides the different results on the different charts, the three search types present good results. Friend and Known searches are very similar, with a small advantage for Friend due to their more frequent interaction with the users.

Although values on expected information are smaller on these two types, it repays by adding a good number of other relevant information. On the other side, famous searches presents more expected information, but could expect even more because there were near 80% that were possible to get.

This system is suitable for different scenarios and contexts, since there are plugins useful for each scenario. Also, the plugins need consecutive improvements in order to "expected but not shown" information that was possible to get tend towards zero (0).

Is it possible to obtain public information, filtering it with personal information? We were expecting to get more information about this feature, however we could get some useful information results, which suggest that this is possible to guide searches and that we can make use of the enormous quantity of information spread out over the Internet. Blogs are a good information source to test this; however there were not many searches for people that own blogs. Although little blogs contribution, we are pleased to observe that when the searched person had a blog, the information obtained on previous iterations was extremely helpful to find that blog. On cases that there was not information previously retrieved by the personal information or Facebook plugins (for the same person), no blog was found.

However, a doubt remained respecting to the filtering of public information using the personal one. We realized that the existence of more information sources will be determinant to better evaluate it, making use of the bigger quantity of ambiguities it can provide.

Resuming, we were able to provide positive answers to our research questions. The information presented to the users is, besides being inter-related as coherent whole, is useful and relevant from their perspective in multiple scenarios. Also, we could guide some searches based on the users' personal information.

6 Conclusions

The motivation and main goal of our work was to make use of the personal information existent on user devices, together with filtered information from public online sources. Moreover, and considering the current state-of-the-art's lack of inter-connected information (when there is more than one type), we found extremely important to inter-relate it as a coherent whole, instead of isolated chunks. It is important to have access to a varied number of sources to present different types of information at different levels of detail, accordingly to the user scenario and context.

To accomplish the aforementioned goals we designed a framework that extracts information from multiple sources and continuously seeks for more relevant results from the users' perspective, in an iterative process. It considers the personal information from users' devices, for being more credible and relevant from the user point of view, to work as a filter or guide searches from more ambiguous sources. It gathers all the information extracted on a single inter-related knowledge base, which has mechanisms that allow the information to be weighted and reinforced if replicated on different sources. To manage the information it was imperative that we could find a single structure to represent it, since different sources have different representations and some of them are not structured at all. To deal with the non-structured information we built a Natural Language Module which transforms it on our single representation.

To evaluate our approach, we developed an example application using our framework, which purpose was providing relevant information about persons. The evaluation tasks consisted on searches for persons with different kinds of relationships with the users (close friends, "known" person or famous/celebrity) to analyze its suitability on different scenarios and contexts. The users' contribution to analyze the truthfulness and relevance of the search results was essential to classify them.

The results suggest that our framework can provide useful and relevant information, either it was expected or not by users. It is very good to have good results on information that was not expected, because on many scenarios that fit on our framework the most relevant/helpful information is the one the users do not remember. From the information not found, except on Famous searches, only a few was possible to retrieve but was somehow missed by us. This suggests that with more relevant information sources the results can be improved, covering some part of the information that was impossible to get with the current sources.

Although the results and users' satisfaction level were good, it was possible to identify some limitations that need further improvements. The information retrieved from the personal information source (mail messages) presented a high level of garbage, which was the result of the overweight assigned to keywords that are part of mail subjects.

On the cases that was possible to compare the influence of personal information on the guidance of public sources searches the results were very good. However, and due to a reduced number of sources, there was not enough ambiguous information to extensively evaluate the filtering of that information.

Moreover, considering the current panorama on the usage of personal information to help the users with useful and relevant information, we believe that the work presented in this dissertation add some value and a new approach to this area.

6.1. Future Work

Although we had good results, we believe that the research undertook in this dissertation brought up future work to be developed in different areas, either extending our results or approaching areas that were not our main focus.

Improve plugin results Although the four (4) plugins used presented good results, it was possible to identify some flaws. The most important correction to make is on personal information and its keywords weight calculation, which should diminish drastically the value of garbage on that plugin. Also, other important improvements on keywords filtering are worth to be made.

Add new information sources We have noticed that each information source has better potentialities for different scenarios. With the addition of new sources, our framework can be applied to more scenarios and the current ones can have their results improved. Also, covering more information sources, we are most likely to get more useful information.

Improve information filtering With more information sources it is possible to have more ambiguities and the necessity of filtering the information grows. Mainly with the addition of more online sources, there is the need to guide the searches to the most relevant information to the user. The results our application had, besides not extensive, suggest that personal information fits in this task.

Natural language Natural language was not one of our main concerns on this dissertation, so we

used a simple module capable of moderate good results. However, to improve the results' quality, it certainly needs some attention to better deal with the non-structured information.

Patterns recognition There is a lot of information on the Internet. Either it is on search engines, personal pages or blogs, it is filled with information. It is relevant to recognize patters to identify which information might be relevant or not. It may allow discarding the irrelevant or too detailed data and identifying the most likely relevant, turning the framework internal process easier. It may also occur on personal information that is not totally composed by relevant information. Some signature patters on mail messages are an example of a pattern recognition that could add value to our framework.

Mobile applications Mobile devices are widely distributed and have become essential tools for the most of us. Moreover, and due to their constant use, they fill great part of the scenarios where our framework can be use. It is most likely that users want immediate answers when they need information and it may not be useful when they reach their personal computers. Mobile devices can surpass this limitation, being available at anytime, anywhere.

Searches beyond persons Our framework was built with the intent of organizing information no matter what its nature. Although the example application is dedicated only to searches for persons, it has potentialities to a wider range of scenarios. In future work, some other applications may be explored focusing on other types of information or even on a general one.

Bibliography

Ahn, J. and Pierce, J. S. (2005). Serefe: serendipitous file exchange between users and devices. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 39–46, New York, NY, USA. ACM.

Ankolekar, A., Szabo, G., Luon, Y., Huberman, B., Wilkinson, D., and Wu, F. (2009). Friendlee: A mobile application for your social life. In *In Mobile HCI 09*. ACM New York, NY, USA.

Beach, A., Gartrell, M., Akkala, S., Elston, J., Kelley, J., Nishimoto, K., Ray, B., Razgulin, S., Sundaresan, K., Surendar, B., et al. (2008). WhozThat? evolving an ecosystem for context-aware mobile social networks. *Network, IEEE*, 22(4):50–55.

Begole, J. B., Matsakis, N. E., and Tang, J. C. (2004). Lilsys: Sensing unavailability. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 511–514, New York, NY, USA. ACM.

Cho, K., Hwang, I., Kang, S., Kim, B., Lee, J., Lee, S., Park, S., Song, J., and Rhee, Y. (2008). HiCon: a hierarchical context monitoring and composition framework for next-generation context-aware services. *Network, IEEE*, 22(4):34–42.

Dey, A. and Abowd, G. (2000). CybreMinder: A Context-Aware System for Supporting Reminders. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 172–186.

Eldridge, M., Barnard, P., and Bekerian, D. (1994). Autobiographical memory and daily schemas at work. *Memory*, 2(1):51–74.

Eldridge, M., Lamming, M., and Flynn, M. (1993). Does a video diary help recall? In *HCI'92: Proceedings of the conference on People and computers VII*, pages 257–269, New York, NY, USA. Cambridge University Press.

Falke, E. (2008). The associative pda 2.0. In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 3807–3812, New York, NY, USA. ACM.

Flynn, M., Pendlebury, D., Jones, C., Eldridge, M., and Lamming, M. (2000). The satchel system architecture: mobile access to documents and services. *Mob. Netw. Appl.*, 5(4):243–258.

Gonçalves, D. (2007). *Narrative Interfaces for Personal Document Retrieval*. PhD thesis, Instituto Superior Técnico.

Huberman, B. A., Romero, D. M., and Wu, F. (2009). Social networks that matter: Twitter under the microscope. *First Monday*, 14(1).

Jung, Y., Anttila, A., and Blom, J. (2008). Designing for the evolution of mobile contacts application. In *MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 449–452, New York, NY, USA. ACM.

Lamming, M., Brown, P., Carter, K., Eldridge, M., Flynn, M., Louie, G., Robinson, P., and Sellen, A. (1994). The Design of a Human Memory Prosthesis. *The Computer Journal*, 37(3):153–163.

Lamming, M., Eldridge, M., Flynn, M., Jones, C., and Pendlebury, D. (2000). Satchel: providing access to any document, any time, anywhere. *ACM Trans. Comput.-Hum. Interact.*, 7(3):322–352.

Lamming, M. and Flynn, M. (1994). Forget-me-not: Intimate computing in support of human memory. In *Proceedings of FRIEND21*, volume 94, pages 2–4.

Lugano, G. (2008). Mobile social networking in theory and practice. *First Monday*, 13(11).

Marmasse, N. and Schmandt, C. (2000). Location-aware information delivery with commotion. In *HUC '00: Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, pages 157–171, London, UK. Springer-Verlag.

Milewski, A. E. and Smith, T. M. (2000). Providing presence cues to telephone users. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 89–96, New York, NY, USA. ACM.

Nakanishi, Y., Takahashi, K., Tsuji, T., and Hakozaki, K. (2002). icams: A mobile communication tool using location and schedule information. In *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*, pages 239–252, London, UK. Springer-Verlag.

Oulasvirta, A., Raento, M., and Tiitta, S. (2005). Contextcontacts: re-designing smartphone's contact book to support mobile awareness and collaboration. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 167–174, New York, NY, USA. ACM.

Raento, M., Oulasvirta, A., Petit, R., and Toivonen, H. (2005). ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. *IEEE PERVASIVE COMPUTING*, pages 51–59.

Robbins, D. C., Lee, B., and Fernandez, R. (2008). Tapglance: designing a unified smartphone interface. In *DIS '08: Proceedings of the 7th ACM conference on Designing interactive systems*, pages 386–394, New York, NY, USA. ACM.

Salber, D., Dey, A., and Abowd, G. (1999). The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pages 434–441. ACM Press New York, NY, USA.

Schilit, B., Adams, N., Gold, R., Tso, M., and Want, R. (1993). The PARCTAB mobile computing system. In *Workstation Operating Systems, 1993. Proceedings., Fourth Workshop on*, pages 34–39.

Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, pages 85–90.

Schmidt, A., Stuhr, T., and Gellersen, H. (2001). Context-Phonebook-Extending Mobile Phone Applications with Context. In *Third Int. Workshop on HCI with Mobile Devices*.

Siewiorek, D., Smailagic, A., Furukawa, J., Krause, A., Moraveji, N., Reiger, K., and Shaffer, J. (2003). SenSay: a context-aware mobile phone. In *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, pages 248–249.

Sohn, T., Li, K., Lee, G., Smith, I., Scott, J., and Griswold, W. (2005). Place-Its: A Study of Location-Based Reminders on Mobile Phones. *LECTURE NOTES IN COMPUTER SCIENCE*, 3660:232.

Tang, J., Yankelovich, N., Begole, J., Li, F., and Bhalodia, J. (2001). ConNexus to awarenex: extending awareness to mobile users. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 221–228. ACM New York, NY, USA.

ter Hofte, G., Otte, R., Kruse, H., and Snijders, M. (2004). Context-aware communication with Live Contacts. In *Conference Supplement of CSCW2004, November 6-10, 2004, Chicago, Ill., USA*.

Terry, M., Mynatt, E. D., Ryall, K., and Leigh, D. (2002). Social net: using patterns of physical proximity over time to infer shared interests. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 816–817, New York, NY, USA. ACM.

A1
Evaluation procedure

A1.1. Evaluation preparation procedure

Procedure Before Initializing the test:

1- Create a blank application on the user's account to use facebook's API
   a. http://www.facebook.com/developers/createapp.php
   b. Click on allow
   c. Click on set up new application on the superior right corner
   d. Choose a random name and accept terms and conditions
   e. Click on advanced, and on application type choose desktop
   f. Save the ApiKey, ApiSecret and UserId
   g. Substitute that values in the GeniusPhone configuration file

2- Send the Scribe executable and its necessary files to the user's computer (folder "dist")

3- Depending on the user mail usage, do the following:

GMAIL

- Ask the users to enter on their GMAIL account -> Settings, and check if IMAP is active. If it is not, click on active
- On GeniusPhone Configuration file, uncomment IMAP on General – Dontload
- On file scribe.ini, on folder "dist", ask the person to write his e-mail and password on the correspondent fields on IMAP.
- At the end of the indexation, ask the person to substitute the private data for ******

THUNDERBIRD

- On GeniusPhone Configuration file, uncomment thunderbird on General – Dontload

WINDOWS LIVE MAIL/ HOTMAIL

- Install Thunderbird
- Creating a new account, attention to the following fields:
  - Servidor POP: pop3.live.com (porta 995);
  - SSL necessary: Yes
  - UserName: xxxxxx@hotmail.com
  - Password: **********
- On GeniusPhone Configuration file, uncomment thunderbird on General – Dontload

WINDOWS MAIL

- On GeniusPhone Configuration file, uncomment windows_mail on General – Dontload

OUTLOOK EXPRESS & OUTLOOK

- On GeniusPhone Configuration file, uncomment windows_mail on General – Dontload

- If that does not work:
- Install Thunderbird
    - Install Thunderbird
    - On Tools, choose Import
    - On Outlook Express find the folder on C:\Documents and Setting\UserName\Local Settings\Application Data\Identities\{"some serial-number"}\Microsoft\Outlook Express on Windows XP or similar on Windows Vista
    - On GeniusPhone Configuration file, uncomment thunderbird on General – Dontload

4- Run the file "correr.bat" on dist folder

5- Wait for the final message, "finished preprocessing keywords"

6- Transfer the testIndex folder created to my personal computer, on C:\

7- Transfer the file preprocessed.db, to my scribe folder

A1.2. Evaluation execution procedure

Procedure Executing the Test:

1- Ask the user to fill the initial questionnaire, and fill the sources information

2- Ask the user to choose 2 close friends; 2 known persons (if possible, that other users also know) that the user knows, but not very well; and 2 famous persons (from the list, Cavaco Silva, Herman José and Cristiano Ronaldo).

3- Ask the users to write in a paper, for each one of the 6 persons, what information they know and are expecting to get about them, that defines them as persons.

4- One at a time, ask the user to search for the person

5- Write the automatically collected results and check the list of expected information

6- Together with the user analyze the information retrieved, classifying it

7- Make a last Interview to the user, to classify our application and get some feedback

A1.3. Evaluation results form

Results:

Type of Search:____
(friend/known/celebrity)

Automatically collected:

total information found:_____
total information shown_____
total iterations_____

Facebook:
information found_____
information shown_____
iterations_____

Scribe:
information found_____
information shown_____
iterations_____

Blogspot:
information found_____
information shown_____
iterations_____

Wikipedia:
information found_____
information shown_____
iterations_____

Analyzed with the user:

From the user Expectations:

expected and shown_____
expected but not shown_____
not expected but shown (relevant)_____
not expected but shown (irrelevant\wrong\garbage) _____

From the information that should be found:

Impossible to get _____
Possible to get _____
Found but not shown _____

From the information found

Facebook:
Right_____
Wrong____
Garbage____

Scribe:
Right_____
Wrong____
Garbage____

wikipedia:
Right_____
Wrong____
Garbage____

Blogspot:
Right_____
Wrong____
Garbage____

Notes

A2

User Characterization

A2.1. User Characterization Questionnaire

User Characteristics

1) Name:_____

2) E-mail:_____

3) Telephone:_____

4) Sex: Male

Female

5) Age:___

6) Habilitations: 4th Grade___
9th Grade___
12th Grade___
BsC___
MsC___
PhD___

7) Hobbies: _____

8) Interests: _____

9) Work/Study subjects: _____

Personal Devices

10) Do you have a computer:_____
11) Which uses you most give to it/them: Work___
Navigate in Internet____
Social Networks____
Games____
e-mail____
chat____
….____

12) Do you have a cell phone_____
13) Do you have a PDA_____
14) What you use them for_____

Plugins Information

Nº of Indexed e-mails ____
Facebook profile ____ (yes/no)

A2.2. User Characterization Results

	User Characteristics					
User	**Sex**	**Age**	**Habilitations**	**Hobbies**	**Interests**	**Work/study subjects**
LE	F	23	BsC	Swimming; Dance	Swimming; Dance; TV series	Biomedic - Biomechanic Engineering
CG	M	55	BsC - Current Teacher	Reading; Sports	Politics; Education	Education
FB	F	29	BsC - Current Doctor	Cinema; movies/series; gym; running	Medicine; Pediatric	Medicine
AP	M	22	BsC	Soccer	Physiotherapy	Physiotherapy
AC	M	22	BsC	Basket	Basket; Poker; Soccer	Networks
JS	M	28	Bsc	Reading; Travelling; Photography; Cinema; Soccer;	Informatics; Soccer	Usability; Accessibility
MM	M	23	12th Grade	Read; Write; Cinema	Travelling; Books; Cinema	Telecommunications and Networks
CA	F	23	BsC	Tennis	Series; Theatre; Cinema; Beach, Travelling	Informatics and Computers Engineering
RM	M	23	BsC	Beach; Sports	Soccer; Informatics; Friends	Networks; communications; informatics
JC	F	23	BsC	Sports	Affective computing	Affective computing
JA	M	22	BsC	Cinema; Sports	Tv Series; Music; Movies	Computer Networking
AF	M	22	BsC	Cinema; Soccer; Karts	Soccer; Communications networking	Computer Networking; Managment and Planning of computer networking
AF2	M	23	BsC	Bicycle; Travelling; Go out	AeroPlanes; Soccer; Cars; BTT	Aero-Nautics Sciences
LG	F	57	BsC - Current Teacher	Gardenning; reading; musics	Travelling; Volunteering	English Teacher

Table A2.1: Pre-evaluation characterization results

	Personal Devices					Plugin Information	
User	**Computer**	**Computer uses**	**Cell Phone**	**PDA**	**Mobile Devices Uses**	**Nº indexed mail messages**	**Facebook**
LE	Yes	Work; mail; Internet	Yes	Yes	Calls; SMS; mail	200	Yes
CG	Yes	Work; mail; Internet; Social Net.	Yes	No	Calls; SMS	450	Yes
FB	Yes	Work; mail; Internet	Yes	Yes	Calls; SMS; work	450	No
AP	Yes	Work; mail; Internet; Social Net.; chat	Yes	No	Calls; SMS	150	Yes
AC	Yes	Work; mail; Internet; Social Net.; chat; Games	Yes	No	Calls; SMS	300	Yes
JS	Yes	Work; mail; Internet; chat	Yes	No	Calls; SMS	250	No
MM	Yes	Work; mail; Internet; Social Net.; chat	Yes	No	Calls; SMS; MMS	4500	Yes
CA	Yes	Work; mail; Internet; chat; watch Tv Series	Yes	No	Calls; SMS	200	Yes
RM	Yes	Work; mail; Internet; Social Net.	Yes	No	Calls; SMS	500	Yes
JC	Yes	Work; mail; Internet; Social Net.; chat; Games	Yes	No	Calls, SMS, GPS; Internet	1000	Yes
JA	Yes	Work; mail; Internet; Social Net.; chat; Games	Yes	Yes	GPS; SMS	450	Yes
AF	Yes	Work; mail	Yes	No	Calls; SMS	5000	Yes
AF2	Yes	Mail; chat; Social Net, Internet	Yes	No	Calls; SMS	500	Yes
LG	Yes	Mail; chat; Social Net	Yes	No	Calls; SMS	40	Yes

Table A2.2: Results from personal devices' usage, and some useful plugins information

A3

User subjective evaluation

A3.1. Evaluation Final Questionnaire

Questionário

Classifica as seguintes frases de 1 a 5, tendo cada uma delas o seguinte significado: 1- Discordo bastante; 2 – Discordo; 3 - Não concordo nem discordo; 4- Concordo e 5- Concordo plenamente.

Frase	**Classificação**
A interface é fácil de perceber e navegar	
A página web é atractiva	
O GeniusPhone fornece informação útil	
A informação apresentada é de fácil compreensão	
O GeniusPhone fornece a informação esperada da pessoa em questão	
A informação apresentada é relevante, do ponto de vista do utilizador	
A fonte da informação, e o contexto em que esta informação está, é perceptível	
O GeniusPhone apresenta um vasto grupo de resultados	
O GeniusPhone apresenta os melhores resultados	

Responda às seguintes questões:

Q: Em que situações acha que o GeniusPhone pode ser útil?

Q: Que fontes de informação adicionais acha interessantes para o GeniusPhone?

Q: Quais as suas sugestões/comentários ao GeniusPhone?

A3.2. Evaluation Final Questionnaire Results

User	Easy Interface	Atractive WebPage	Provide useful info.	Understandable info	Provide expected info	Relevant from user perspective	Source and context perceptive	Wide group of results	The best results	Average
LE	5	5	4	4	3	4	4	3	4	4,0
CG	5	5	5	5	4	5	5	5	4	4,8
FB	5	5	4	3	3	4	3	3	2	3,6
AP	4	3	4	4	2	4	3	3	3	3,3
AC	5	3	3	3	4	4	3	4	3	3,6
JS	5	4	4	4	3	4	4	4	3	3,9
MM	4	4	4	3	4	4	4	3	3	3,7
CA	5	4	4	5	3	4	4	3	3	3,9
RM	4	4	4	3	4	4	3	5	4	3,9
JC	5	5	4	5	4	4	5	3	3	4,2
JA	5	4	5	4	4	4	5	5	4	4,4
AF	5	4	4	3	4	4	5	4	3	4,0
AF2	4	3	4	5	4	4	4	4	3	3,9
LG	5	5	5	5	4	5	5	5	4	4,8
Average	4,7	4,1	4,1	4,0	3,6	4,1	4,1	3,9	3,3	4,0
Stdev	0,5	0,8	0,5	0,9	0,6	0,4	0,8	0,9	0,6	
Mode	5,0	4,0	4,0	5,0	4,0	4,0	4,0	3,0	3,0	
Inter-Quartile Range	0,8	1,0	0,0	2,0	1,0	0,0	1,8	1,8	1,0	

Table A3.1: Final evaluation questionnaire results

A4
Evaluation results

A4.1. Users' expectations and results relevance

	User Expectations					
User	Type of search	Expected information	Expected Found	Expected N/ Found	N/ Expected but Relevant	Not Expected; Wrong or Garbage
LE	friend	**5**	**0**	**5**	**8**	**0**
		4	1	3	9	8
	"known"	4	0,5	3,5	11	17
		4	2,5	1,5	5	2
	Famous	5	3,5	1,5	4	3
		3	2	1	5	2
CG	friend	3	2,5	0,5	45	20
		3	1	2	21	25
	"known"	2	0	2	6	1
		3	0	3	7	2
	Famous	3	2	1	5	3
		3	2	1	7	2
FB	friend	8	0	8	1	20
		10	4	6	8	23
	"known"	4	0	4	14	14
		4	0	4	7	12
	Famous	3	2	1	7	2
		5	3	2	4	3
AP	friend	7	3	4	3	0
		6	3	3	14	7
	"known"	4	0	4	11	15
		6	0	6	0	0
	Famous	3	2	1	6	3
		7	3,5	3,5	7	3
AC	friend	8	5	3	26	29
		7	0,5	6,5	13	11
	"known"	4	3	1	10	15
		6	2,5	3,5	1	1
	Famous	5	2	3	7	3
		6	3,5	2,5	3	2
JS	friend	1	0	1	0	0
		3	2	1	9	20
	"known"	1	0,5	0,5	6	14
		2	1	1	14	2
	Famous	6	4	2	2	2
		3	0	3	11	3
MM	friend	4	3	1	23	24
		5	4,5	0,5	17	14
	"known"	5	3	2	26	17
		4	3	1	34	2
	Famous	6	4	2	7	3
		4	2,5	1,5	8	4
CA	friend	5	1	4	9	0
		5	2	3	0	0
	"known"	7	0	7	0	0
		5	1	4	15	13
	Famous	6	4	2	3	2
		5	3,5	1,5	10	3

Table A4.1: Users' expectations and relevant results (User 1 to 7)

	User Expectations					
User	Type of search	Expected information	Expected Found	Expected N/ Found	N/ Expected but Relevant	Not Expected; Wrong or Garbage
RM	friend	8	4	4	2	7
		7	3,5	3,5	22	18
	"known"	5	0	5	15	33
		4	2	2	8	0
	Famous	5	3	2	4	2
		5	2	3	11	4
JC	friend	3	3	0	5	16
		3	2	1	16	30
	"known"	1	0	1	0	1
		3	1,5	1,5	10	15
	Famous	3	3	0	7	3
		2	2	0	4	2
JA	friend	2	1	1	25	4
		2	0	2	25	3
	"known"	3	2	1	12	7
		4	3,5	0,5	22	4
	Famous	4	3	1	5	2
		6	4	2	7	2
AF	friend	3	1,5	1,5	19	28
		3	1	2	20	27
	"known"	3	0	3	10	0
		3	0	3	11	20
	Famous	3	2,5	0,5	5	2
		3	1	2	10	3
AF2	friend	8	4	4	22	20
		8	6	2	11	4
	"known"	5	4	1	25	26
		6	3	3	1	1
	Famous	7	3	4	6	2
		10	6	4	4	3
LG	friend	5	2	3	1	0
		8	5	3	25	11
	"known"	3	1,5	1,5		1
		4	2	2	1	0
	Famous	2	2	0	2	3
		4	3	1	5	3

Table A4.2: Users' expectations and relevant results (User 8 to 14)

		From the Not Found		
User	Type of search	Impossible	Possible	Found but Not Shown
LE	friend	**4**	**1**	**0**
		3	0	0
	"known"	2,5	1	0
		1,5	0	0
	Famous	1	0,5	0
		0	1	0
CG	friend	0,5	0	0
		2	0	0
	"known"	2	0	0
		3	0	0
	Famous	0	1	0
		0	1	0
FB	friend	8	0	0
		6	0	0
	"known"	4	0	0
		4	0	0
	Famous	1	0	0
		0	2	0
AP	friend	4	0	0
		3	0	0
	"known"	4	0	0
		4	2	0
	Famous	0	1	0
		1	2,5	0
AC	friend	2	1	0
		6,5	0	0
	"known"	1	0	0
		3,5	0	0
	Famous	0	3	
		0	2,5	0
JS	friend	1	0	0
		0	1	0
	"known"	0,5	0	0
		1	0	0
	Famous	0	2	0
		0	3	0

		From the Not Found		
User	Type of search	Impossible	Possible	Found but Not Shown
RM	friend	4	0	0
		3	0,5	0
	"known"	5	0	0
		2	0	0
	Famous	1	1	0
		2	1	0
JC	friend	0	0	
		0	1	0
	"known"	0	1	0
		0	1,5	
	Famous	0	0	0
		0	0	0
JA	friend	0	1	0
		2	0	0
	"known"	0	1	0
		0,5	0	0
	Famous	0	1	0
		1	1	0
AF	friend	1,5	0	0
		1	1	0
	"known"	3	0	0
		3	0	0
	Famous	0	0,5	0
		0	2	0
AF2	friend	4	0	0
		2	0	0
	"known"	1	0	0
		3	0	0
	Famous	0	4	0
		1	3	0
LG	friend	3	0	0
		3	0	0
	"known"	1,5	0	0
		2	0	0
	Famous	0	0	0
		0	1	0

Table A4.3: Analysis of information not found

A4.2. Retrieved information analysis

		General Info shown				
User	Type of search	Total	Correct	Wrong	Garbage/ irrelev.	Useful Iterations
LE	friend	8	8	0	0	1
		24	16	0	8	1
	"known"	28	11	0	17	1
		12	10	0	2	1
	Famous	14	11	0	3	1
		11	9	0	2	1
CG	friend	74	54	0	20	2
		49	24	3	22	1
	"known"	13	6	6	1	1
		9	7	0	2	1
	Famous	14	11	0	3	1
		11	9	0	2	1
FB	friend	21	1	0	20	1
		43	20	0	23	2
	"known"	28	14	0	14	1
		20	8	0	12	1
	Famous	11	6	0	5	1
		14	10	0	4	1
AP	friend	9	9	0	0	1
		24	16	0	8	1
	"known"	26	11	0	15	1
		0	0	0	0	
	Famous	11	8	0	3	1
		14	11	0	3	1
AC	friend	61	32	0	29	1
		24	13	0	11	1
	"known"	28	13	0	15	1
		5	4	0	1	1
	Famous	14	11	0	3	1
		11	8	0	3	1
JS	friend	0	0	0	0	
		32	12	0	20	2
	"known"	20	6	0	14	1
		18	16	0	2	1
	Famous	11	9	0	2	1
		14	11	0	3	1
MM	friend	55	31	0	24	1
		39	25	0	14	1
	"known"	48	31	0	17	1
		51	49	0	2	2
	Famous	14	11	0	3	1
		17	13	0	4	1
CA	friend	10	10	0	0	1
		4	3	0	1	1
	"known"	0	0	0	0	
		29	16	0	13	1
	Famous	11	10	0	1	1
		17	14	0	3	1
RM	friend	19	12	0	7	1
		47	29	0	18	1
	"known"	48	15	0	33	1
		10	10	0	0	1
	Famous	13	11	0	2	1
		17	13	0	4	1
JC	friend	40	24	0	16	1
		57	23	4	30	1
	"known"	34	19	0	15	1
		0	0	0	0	
	Famous	13	11	0	2	1
		11	10	0	1	1
JA	friend	32	27	1	4	1
		30	26	0	4	1
	"known"	24	17	0	7	1
		33	29	0	4	1
	Famous	11	10	0	1	1
		13	11	0	2	1
AF	friend	52	24	0	28	1
		52	22	2	28	1
	"known"	10	10	0	0	1
		32	12	0	20	1
	Famous	11	9	0	2	1
		14	11	0	3	1
AF2	friend	25	21	0	4	1
		50	27	2	21	1
	"known"	6	5	0	1	1
		61	35	4	22	2
	Famous	14	11	0	3	1
		11	9	0	2	1
LG	friend	6	6	0	0	1
		48	39	0	9	2
	"known"	7	6	0	1	1
		4	4	0	0	1
	Famous	11	8	0	3	1
		14	11	0	3	1

Table A4.4: General information presented

		From Facebook				From Scribe				From BlogSpot				From Wikipedia			
U	Search Type	Total	Right	Wrong	Garbage irrelev.	Tota	Right	Wrong	Garbage irrelev.	Total	Right	Wrong	Garbage/ irrelev.	Total	Right	Wrong	Garbage/ irrelev.
LE	friend	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		7	7	0	0	17	9	0	8	0	0	0	0	0	0	0	0
	"known"	0	0	0	0	28	11	0	17	0	0	0	0	0	0	0	0
		9	7	0	2	3	3	0	0	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3
		0	0	0	0	0	0	0	0	0	0	0	0	11	9	0	2
CG	friend	29	27	0	2	37	19	0	18	8	8	0	0	0	0	0	0
		13	10	3	0	36	14	0	22	0	0	0	0	0	0	0	0
	"known"	7	6	0	1	0	0	0	0	0	0	0	0	6	0	6	0
		9	7	0	2	0	0	0	0	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3
		0	0	0	0	0	0	0	0	0	0	0	0	11	9	0	2
FB	friend	0	0	0	0	21	1	0	20	0	0	0	0	0	0	0	0
		0	0	0	0	35	12	0	23	8	8	0	0	0	0	0	0
	"known"	0	0	0	0	28	14	0	14	0	0	0	0	0	0	0	0
		0	0	0	0	20	8	0	12	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	11	6	0	5
		0	0	0	0	0	0	0	0	0	0	0	0	14	10	0	4
AP	friend	5	5	0	0	4	4	0	0	0	0	0	0	0	0	0	0
		7	6	0	1	17	10	0	7	0	0	0	0	0	0	0	0
	"known"	0	0	0	0	26	11	0	15	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	11	8	0	3
		0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3
AC	friend	9	9	0	0	52	23	0	29	0	0	0	0	0	0	0	0
		0	0	0	0	24	13	0	11	0	0	0	0	0	0	0	0
	"known"	3	3	0	0	25	10	0	15	0	0	0	0	0	0	0	0
		5	4	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3
		0	0	0	0	0	0	0	0	0	0	0	0	11	8	0	3
JS	friend	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	24	4	0	20	8	8	0	0	0	0	0	0
	"known"	0	0	0	0	20	6	0	14	0	0	0	0	0	0	0	0
		0	0	0	0	18	16	0	2	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	11	9	0	2
		0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3
MM	friend	15	15	0	0	40	16	0	24	0	0	0	0	0	0	0	0
		4	4	0	0	35	21	0	14	0	0	0	0	0	0	0	0
	"known"	9	9	0	0	39	22	0	17	0	0	0	0	0	0	0	0
		17	17	0	0	27	25	0	2	7	7	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3
		0	0	0	0	0	0	0	0	0	0	0	0	17	13	0	4
CA	friend	2	2	0	0	8	8	0	0	0	0	0	0	0	0	0	0
		4	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	"known"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	29	16	0	13	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	11	10	0	1
		0	0	0	0	0	0	0	0	0	0	0	0	17	14	0	3
RM	friend	9	9	0	0	10	3	0	7	0	0	0	0	0	0	0	0
		10	10	0	0	37	19	0	18	0	0	0	0	0	0	0	0
	"known"	0	0	0	0	48	15	0	33	0	0	0	0	0	0	0	0
		10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	13	11	0	2
		0	0	0	0	0	0	0	0	0	0	0	0	17	13	0	4
JC	friend	6	6	0	0	34	18	0	16	0	0	0	0	0	0	0	0
		3	3	0	0	50	20	0	30	0	0	0	0	4	0	4	0
	"known"	0	0	0	0	34	19	0	15	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	13	11	0	2
		0	0	0	0	0	0	0	0	0	0	0	0	11	10	0	1
JA	friend	9	8	1	0	23	19	0	4	0	0	0	0	0	0	0	0
		4	3	0	1	26	23	0	3	0	0	0	0	0	0	0	0
	"known"	0	0	0	0	24	17	0	7	0	0	0	0	0	0	0	0
		0	0	0	0	33	29	0	4	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	11	10	0	1
		0	0	0	0	0	0	0	0	0	0	0	0	13	11	0	2
AF	friend	3	3	0	0	49	21	0	28	0	0	0	0	0	0	0	0
		9	8	0	1	43	14	2	27	0	0	0	0	0	0	0	0
	"known"	10	10	0	0	0		0		0	0	0	0	0	0	0	0
		0	0	0	0	32	12	0	20	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	11	9	0	2
		0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3
AF2	friend	9	9	0	0	16	12	0	4	0	0	0	0	0	0	0	0
		8	7	0	1	40	20	0	20	2	0	2	0	0	0	0	0
	"known"	6	5	0	1	0	0	0	0	0	0	0	0	0	0	0	0
		27	27	0	0	26	0	4	22	8	8	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3
		0	0	0	0	0	0	0	0	0	0	0	0	11	9	0	2
LG	friend	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		29	27	0	2	11	4	0	7	8	8	0	0	0	0	0	0
	"known"	7	6	0	1	0	0	0	0	0	0	0	0	0	0	0	0
		4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Famous	0	0	0	0	0	0	0	0	0	0	0	0	11	8	0	3
		0	0	0	0	0	0	0	0	0	0	0	0	14	11	0	3

Table A4.5: Information retrieved per plugin