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. 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. 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 about someone, either they were expecting it or not, resorting both to personal and public information.

Keywords
Personal Information, Information Extraction, Public Information, Inter-Related Information, Knowledge Base.

INTRODUCTION
Not so long ago, our personal information, whether documents, agendas, notes or interaction registers, like letters, 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 immense universe of 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 reachable 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. Never have we had so much personally relevant information at our disposal and the ability to access it almost instantaneously. For instance, mobile phones are widely used and have become essential tools for most of us. They contain information about the users, their habits and daily interactions, as no other person or device. These tools have long surpassed their original role as simple communication tools, being always available, gathering enormous amounts of information, and being able to easily communicate with other personal devices or public information sources (homepages, social networks,...). The mobile phone is the perfect witness to each user’s personal interactions and the perfect candidate to help the user whenever needed. However, they are still of little use when the user needs to meaningfully access personal information, created both on the phone and in other devices, in a related, synergistic way. Nowadays, the usage of mobile device acquired knowledge is restricted to word dictionaries, recent contact recall or other somehow basic functionalities. Even projects that try to go beyond these basic functionalities are restricted to a limited set of information, ignoring the enormous amount of personal and public information possibly available on and from a mobile device and its inter-relation.

Overall, 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 [3,5] 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.

We present an approach and underlying framework that goes beyond the state of the art by inter-relating the users’ personal information and interactions with others from their computers and mobile devices, using it to gather additional data from online public sources. The information retrieved from the personal devices, due to its personal and trustable character, helps us filter the information retrieved from other less trustable and structured sources. While laying the ground work for applications in several contexts and domains, we believe that the developed framework is likely to have great benefits in a mobile context, one where, nowadays, the user is highly personal information-deprived and where that information is likely to be required on time.

In this document, we present the approach and the concrete framework as well as an example web application, GeniusPhone, using the developed framework to retrieve information about someone.

**RELATED WORK**

There are some applications that improve the user's daily life and relation with his surrounding environment by using some kind of contextual or personal information. As an example, "Forget-Me-Not" [4] records social interactions, together with sending/receiving documents, and can recall that information upon user request. However, most of the research in this area tries to use personal or social interaction focusing on the users or their contacts' context, resorting mostly to sensors. Context information often considered is related to location, contacts social interactions [6], instant messaging application status, device state [2], idle time, schedule information, among others. These applications work only as long as the users allow and want their friends to be aware of their context information. Another way to obtain information about someone is explored in "WhozThat?" [1], by resorting to online social network profiles like Facebook or MySpace. Also in interest-matching is the Social Net project [7], which records the time and duration of encounters with other users. With this information, it tries to find patterns of physical proximity between people over time, to assume they may have interests in common. If someone has two friends with similar interests, he is suggested to introduce them.

What all the above approaches lack is a usage of personal information, from different sources, taking advantage of all the knowledge the devices have about their owners. Further, little attempt is made in trying to combine such information with that from public sources. This knowledge could be used to help the user anytime, anywhere, relating, summarizing and providing important data when it is required, in a personally relevant way. Some applications might automatically present the essential information for a meeting or collect, automatically, by proximity, context or demand, all the relevant information about someone (according to our own personal experience with that someone). 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?". Our system is able to provide answers to those questions by gathering and interrelating information from the user’s devices enriched with other public information sources, to offer the user context- and personally-sensitive information when it is needed.

**MANAGING PERSONAL INFORMATION**

There is an immense 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.

**Detailing our approach**

To provide a better understanding of our challenges and goals, we outline a set of possible scenarios as well as the main concepts underlying our approach.

**Scenarios**

The framework developed in this thesis’ context aims to help the user in several different situations. We describe 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 at 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 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 his knowledge on the area.

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.

Personal Content-Filtering of 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.

Inter-related and Coherent Information
Our approach makes place for several possible information sources. 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 a unified integrated index, I could search only for CHI Conference, immediately obtaining information from all the above sources inter-related as a coherent whole.

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 process) 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 section.

![Figure 1 – Framework High-Level Architecture](image)

### Architecture

The framework architecture is based on three main components: Plugins, Plugin Manager and Coordinator (Figure 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, v2.6.

### 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 between 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).

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. 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. 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.

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.

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. Each plugin has to register in the Plugin Manager informing what kind of information they can obtain.

### 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, the 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. 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 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.

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.

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.

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 2. In this case frame, it 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.

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 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.

There is the need to access the information represented in the Knowledge Base, either to present 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(["?node", Literal('subject'), "?person"], 
                    ("?node", Literal('relation'), Literal("interests in")),
                    ("?node", Literal('object'), Literal("soccer")),
                    ("?node", Literal('weight'), "?wei")]
relationForWeight = query(spqgraph, selectMei, whereMei)

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

Iteratively seeking for information

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 constritions. 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")))
```

Updating knowledge

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 "Machinery", and in his blog says to be interested in "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"), as we are interested in the concepts and contexts and not an exact copy of the original data.

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.

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:
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:

\[ 1 - \text{OldWeight} [0.8] = 0.2 \]

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

\[ \text{AddWeight} = 0.2 \times \text{ArrivingWeight}[0.5] = 0.1 \]

The calculated value is added to the old value:

\[ \text{NewWeight} = \text{OldWeight} + \text{AddWeight} = 0.9 \]

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 iterating, 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 has the same value (12). The expression is:

\[ \text{Value to Decrease} = \ln((\text{months}/10) + 1) \]

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.

EXAMPLE APPLICATION

To prove that our framework can provide relevant information to the user and satisfactory results, we developed an example application, called GeniusPhone, with the goal of obtaining information about a person (Figure 4). 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 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.

**Application platform**
There are many contexts where our framework can be used. Using mobile devices as an entry point it is possible to use our applications 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 our framework 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.

**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.

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 5).

**EVALUATION**
To acknowledge our approach as a success, some research questions need to be answered: 1) Can our framework inter-relate information from different sources?; 2) Can this framework provide useful and relevant results from the user point of view?; 3) Is the system suitable for different scenarios and contexts?; and, 4) Is it possible to obtain public information, by filtering and guiding searches with personal information?

**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).

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, 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.

**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 comprehend between 22 and 57 years (averaging 28 years old).

**Tasks**
Users had to search for some persons in order to obtain information about them. These searches were divided in
three types (Close person, Know person, Famous person), 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.

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. 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.

Figure 6 - Users’ expectations and achieved results
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.

Evaluating Users’ Expectations and Results Relevancy.
Before performing our tasks the users described what 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. To answer this question we have to analyze some different aspects. First, which information from users’ expectations was, and was not presented? Figure 6 shows the results, in percentage, for the three different search types.

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 them through their difficulties. Besides, it is important to remember that the users’ expectations were not pre-processed in any way. Also, we can observe on Figure 7 that, from the information that was expected 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. This indicates that, from the information users were expecting, only a few was possible for us to retrieve but somehow we missed it. 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
Figure 8 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.

Analyzing information sources. Figure 9 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.

Search Types. Searches performed for a Famous person present better results when compared to the users' expectations, 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. 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 type 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 (greater percentage in Friends than in Known) were not possible to retrieve as they were not available (as verified with the users by post-checking the sources). This combination of results suggests that when more interaction happens, and consequently more personal information is shared, the results are better.

User's Opinions. 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 relevancy of the results, which were rated with average values between 4 and 4.1. These ratings suggest that 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.

CONCLUSIONS
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. However, the available amount of information is enormous and to be useful it must be contextualized and summarized. Our framework gathers personal information from the user's devices, and use it as a filter to the information available in public sources like search engines and social networks. After an iterative process of searching, renewing and improving the information retrieved, from the user point of view, it is able to present contextualized structured information. An example application, evaluated with users, was presented as valuable reaching the aforementioned personal information access, inter-relation and coherency goals. The plug-in based architecture allows us to easily extend the framework, so in the future we plan to explore new information sources. Particularly, we will extend our web search plug-ins to be able to recognize relevant chunks of information about persons, particularly personal web pages, or events. Using this platform, there are several scenarios and applications that can be explored, so we will focus also on trying to find new uses to it, mainly in mobile contexts.

REFERENCES