Semantic Integration and Language Access to Mobile Data

Raimondas Lencevicius
Nokia Research Center Cambridge
3 Cambridge Center
Cambridge, MA 02142
Raimondas.Lencevicius@nokia.com

Alexander Ran
Nokia Research Center Cambridge
3 Cambridge Center
Cambridge, MA 02142
Alexander.Ran@nokia.com

ABSTRACT
Real-world data can significantly enhance the functionality of mobile services. For this, real-world data needs to be collected, stored and integrated with other information available on mobile devices. A flexible and user-friendly interface to the data is also needed. This paper describes an experience in collecting real-world data and integrating it into a semantic data repository. We use an innovative Natural Query language and engine to automatically connect the resulting repository to the natural language user interface. The resulting system on S60 mobile platform successfully answers user questions about Personal Information Management (PIM) data extended with real-world data.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**], H.5.2 [**User Interfaces**]: *Natural language*

Keywords
Data access, query language, natural language.

1. INTRODUCTION
Mobile devices make a perfect user interface to the real-world environment. They are constantly carried with the user [2] enabling gathering of user location information. Mobile devices are equipped with more and more sensors including GPS receivers, Bluetooth transmitters and receivers, RFID receivers and others. They also receive and store information about such real-world events as messages, phone calls, meetings, application usage and access to digital services. It is therefore natural to expect that this real-world data should be collected and made accessible on mobile devices. However, there are some open questions that need to be resolved in order to make this kind of data useful and accessible both to the programs and to the mobile device users. Collected real-world data must be structured and integrated with other information available on mobile devices such as, for example, the information found in the user's phone book or calendar. There also needs to be an intuitive interface that allows flexible access to collected information.

In this paper we present a framework that collects real-world data, structures it according to an extended PIM ontology, augments and integrates it with earlier collected data, and stores it in an RDF repository.

To access the data, an intuitive interface is needed. Natural language based information access is increasingly viewed as a promising alternative to graphical user interfaces (GUIs), especially in the domain of mobile devices. We have developed a Natural Query language and engine that can automatically map meaning representation produced by language systems into formal database queries. This enables us to provide a natural language interface to the integrated real-world and on-device data.

The paper describes our data gathering framework (Section 2) and explains our solution to data storage (Section 3). Natural Language Interface to the stored data is simplified via using Natural Query system (Section 4). Then we describe our experience with the system (Section 5). We finish with the related work and conclusions.

2. DATA GATHERING
Data collection on mobile devices is an active field of research [3][8]. We have extended one of the frameworks available within Nokia to collect events that occur on a mobile device: phone calls, SMS messages, nearby Bluetooth devices, and GSM locations. All of these events are tagged with a timestamp when they occur. For phone calls the device records the phone number called (or the phone number that called the phone user) and the call length. For messages, the phone number and the message text is recorded. A GSM location change event is recorded when the cell tower associated with the phone changes. Finally, the phone periodically scans for Bluetooth devices in its vicinity and records their names and IDs.

Although the real world data gathered is interesting by itself, it becomes even more important when connected to the data already available inside the device. Mobile devices store a rich set of structured information. The address book or phone book application contains names, phones, addresses and affiliations of personal contacts. The calendar application contains entries for meetings with participants, meeting location and time. All these data are related. Retrieving these data based on their relation could be very useful for device owners. With such retrieval capabilities they could learn who called them when they were in California, or when is their next meeting with Ann from Accenture. Unfortunately, the relations between different data items are not explicit when the events occur or information is entered in some application. Therefore it is important to integrate the collected data by explicating its relation to data available on the device. To achieve this goal, we have developed an extended PIM ontology that covers all relevant types of information available on the mobile device: from observed events, information from external data stores, to on-device data from several mobile applications. Once the data was structured and augmented with relations, it is stored in RDF [10] repository.

3. STRUCTURED DATA STORAGE AND ONTOLOGY
We created the PIM ontology to cover all data available in the device. We considered using such standard ontologies as W3C

foaf [4] and vcard [15]. However, the information available on the mobile device was richer than the types supported by standard ontologies. Main classes in our ontology are *Person, Organization, CalendarEntry, EmailAddress, Location, Observation, Message, Call, and PhoneNumber*. Part of the ontology is shown in Figure 1.

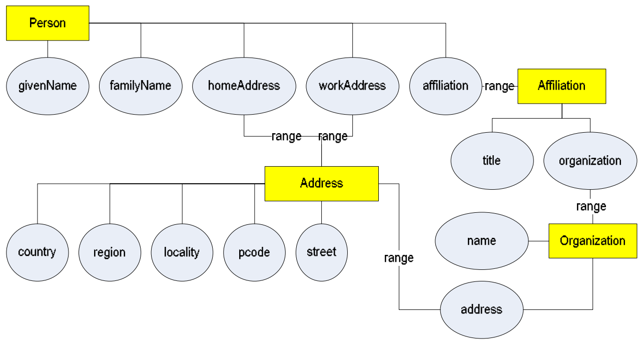


Figure 1. Part of Mobile PIM ontology

There are about ten more secondary classes and class attributes. The real-world data observations are stored in the objects of *Observation* subclasses: *BTDeviceObserved, CallObserved*, *MessageObserved*, and *LocationObserved*. The attributes of these objects connect with other objects of the repository. For example, the *phoneNumber* attribute of a *CallObserved* is of type *PhoneNumber*, which is also used in the attribute *phoneNumber* of a *Person* or *Organization* class. Therefore the gathered real-world data directly integrates with the on-device data.

For some other data, programs or users have to add information to facilitate integration. For example, a Bluetooth device ID and name attributes have to be added to the *Person* class and filled in with concrete values in order to associate the *BTDeviceObserved* observation to a specific person carrying a Bluetooth device.

Another area where observed data integrates with on-device data is the location information. A significant part of ontology deals with locations at various granularity levels: from meeting rooms, to office buildings, cities, and countries. We use the *partOf* relation between different objects to represent geographic or organizational inclusions. For example, a relation can indicate that Boston is a part of Massachusetts, which in turn is a part of the USA. This attribute is also used to describe the GSM location containment within a certain geographical object. Since GSM locations are somewhat imprecise, we have chosen to associate them with town or city level geographical entities. This provides sufficient information in most cases.

Overall, we found that our RDF repository is significantly more flexible than a relational database. For example, it naturally supports multiple classes of contacts, multiple affiliations per person, and supports a sophisticated typing system.

4. NATURAL LANGUAGE INTERFACE

Although the repository of integrated real-world and in-device data can be used in variety of ways, for example, via querying it using SPARQL [14], we were interested to provide an intuitive and flexible user interface to it. We decided that a general natural language interface to a rich data set could be more effective than a GUI based application.

As a rule, information bases and language systems are developed independently of each other. Therefore information bases are not designed for interaction using natural language and their integration process is mostly ad hoc, manual process. Figure 2 is a sketch of a typical architecture that is used to provide a natural language interface to databases and other back-end or native services.

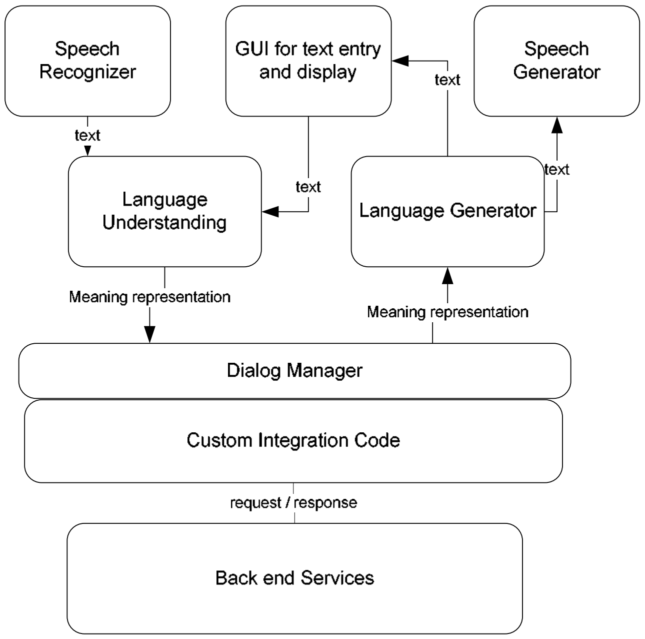


Figure 2. Architecture sketch of Natural Language Interface to Services

The speech recognition and generation components translate between text and speech modalities. The language understanding component converts the text into a formal representation of meaning sometimes called *semantic frame* [12]. The language generation component converts the formal meaning representation to a natural language text [1]. The dialog manager uses the context of conversation to complete frames received from the language understanding module or created by the custom integration code from responses of backend services. The custom integration code also translates meaning representation frames it receives from the dialog manager into a standard database query or backend specific API requests.

We have designed and implemented the Natural Query (NQ) language and engine [9] that removes the need for custom integration code. NQ can automatically map meaning representation produced by language systems into precise database queries. NQ employs two mechanisms: language tags and data graph search to return requested data using only the information in the meaning representation of the user request.

Language tags are words, expressions, and linguistic tokens attached to database elements such as classes and properties. Multiple tags can be attached to a single element and a single tag can be attached to multiple elements. Language tags are the names of the corresponding categories used by the language system(s).

Figure 3 illustrates language tags associated with a part of our PIM ontology. A generalization like "*Contact*" can be attached to specific classes like "*Person*" and "*Organization*". Tags like "*in*" can be attached to all location elements. A general reference like "*Name*" can be attached to multiple elements like "*givenName*", "*familyName*", and so on. In our RDF repository of real-world and in-device data, we added language tags to the RDF objects using a subproperty of RDFS *label* field.

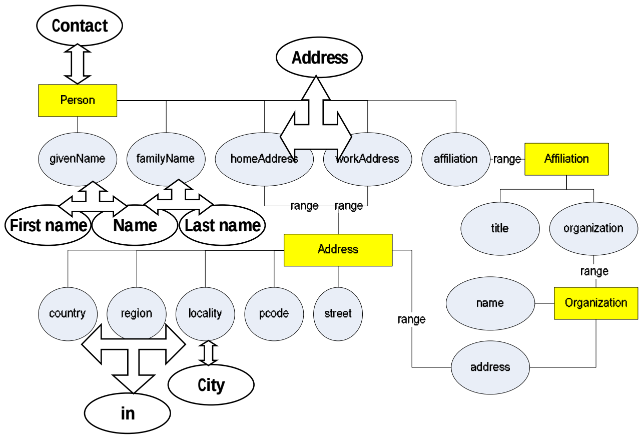


Figure 3. Language tags for database elements

While ad hoc integration needs to have the information about the organization of the database, NQ avoids the need for such information by using a graph search to achieve the same objective. Given a question "*Who are my contacts at IBM in Ulm?*", NQ finds paths connecting the nodes known from the meaning representation, such as *"Person"*, *"name"*, *"Organization"*, *"City"*, *"Ulm"*, and *"IBM"*. One of such paths is highlighted in Figure 4.

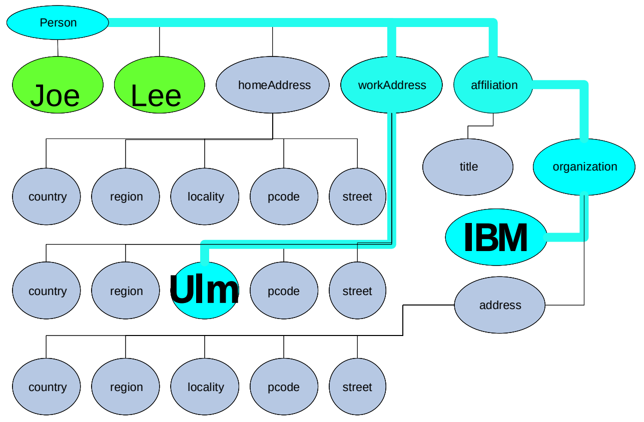


Figure 4. Answering query via graph search

We have created a proof of concept implementation of NQ in Python [7] that runs on S60 [11] mobile phones. Full description of the Natural Query system is presented in [9].

5. EXPERIENCE WITH THE SYSTEM

We tested our system on a PIM test data set containing 550 contacts with about 150 meetings and 250 phone calls, which is normal for executives with a lot of contacts and meetings. The repository contained over 11000 RDF triples. We asked over 50 natural queries corresponding to over 600 parameterized questions. We did not count various parameters, such as different cities or names, since these numbers can be made arbitrarily large and the resulting number of questions does not really reflect the capabilities of the system.

The system can answer questions ranging from "*What is the email of John?*" to "*Where does Ann work?*" to "*My meetings next week with John*" and "*Who called me yesterday*". Some of these questions would convert to quite complex relational or SPARQL queries. For example for the query "*Who called me yesterday*", we need to find all telephone numbers of calls that occurred yesterday and then find all people who have these telephone numbers. NQ query for this is very simple: "*fromClass = 'Person', select = ['givenName', 'familyName'], where = [(["ReceivedPhoneCall", 'start'], TimeInterval ('yesterday'))]*".

If we classified questions according to domains, one domain would contain questions about the personal information data from an address book application, for example "*Who works as a real estate broker?*". Another set of questions is about meetings, for example, "*When are my meetings next month at MIT?*". Yet another set is about calls and messages, for example, "*Who called me last Friday?*". Finally there are questions spanning multiple domains, for example, "*What are emails of people who participated in a meeting on Monday?", "Who called me when I was in Finland*?", and so on.

All these types of queries (Figure 5) were successfully created and executed on the extended PIM data store.

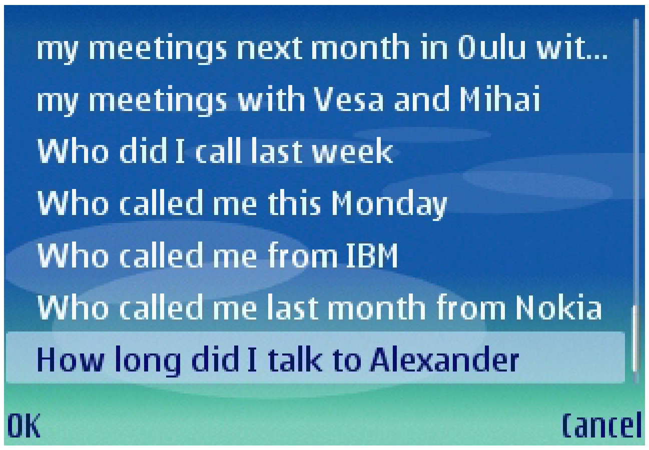
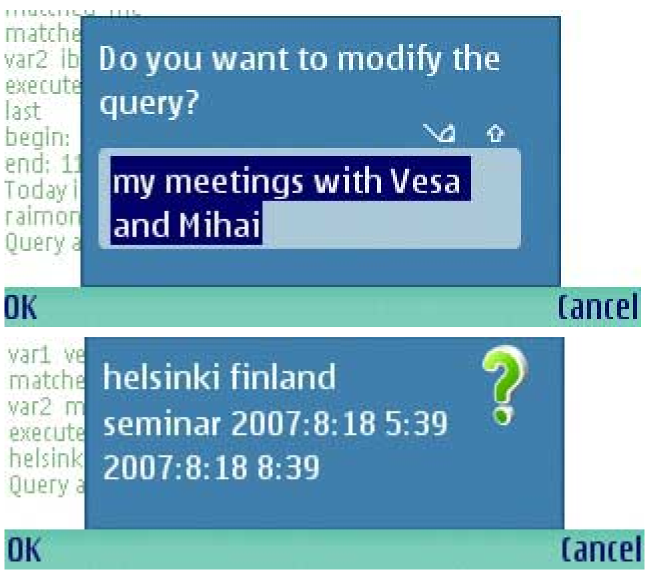

my meetings next month in Oulu wit...
my meetings with Vesa and Mihai
Who did I call last week
Who called me this Monday
Who called me from IBM
Who called me last month from Nokia
How long did I talk to Alexander

Figure 5. Example questions

We found out that we could easily ask questions both about the in-device data and the collected real-world data. Integration of the two enhanced our question answering capability significantly, allowing such questions as "*Who called me when I was in Helsinki?*", "*Which messages did I receive during the meeting with Juha?*". Although the detection of someone's Bluetooth device is a weak indication the phone user met other person with a Bluetooth device, in our experiments we assumed such implication. This allowed us to ask questions such as "*Who did I meet last week?*".

Test NQ queries mostly returned expected answers (96% recall, 92% precision) (Figure 6) including the approximate answers where the exact answers were not available. For example, the question "*When was my meetings with Sam last month?*" had no exact answers, so the system returned approximate answers of meetings with Sam that did not occur last month as well as the meetings that occurred last month, but did not include Sam.

The performance of the system was acceptable with answers taking from less than a second to several seconds. The system implementation is a prototype in Python that was not optimized for memory or speed. The detailed evaluation of system performance is outside the scope of this paper. We are planning to optimize the system performance in the near future.


Do you want to modify the query?

my meetings with Vesa and Mihai

helsinki finland
seminar 2007:8:18 5:39
2007:8:18 8:39

Figure 6. Example question and answer

6. RELATED AND FUTURE WORK

Real world data has been gathered on mobile devices by a number of projects including Context [8] and Reality Mining [3]. In our work, we have extended one of the data gathering frameworks available at Nokia.

Mobile data storage in RDF repositories is investigated by ConnectingMe [5] project at Nokia Research Center. We are collaborating with ConnectingMe in the ontology and repository development.

We have not discovered any research directly corresponding to the Natural Query approach. The Precise system by Popescu et al. [6] attaches language tokens to database elements in a way very similar to language tags of NQ. Also the query derivation approach of Precise is based on database graph search. NQ uses a more flexible data model, supports incomplete answers, and collects data for explanations.

In the future, we plan to connect our system to such natural language and speech systems as TINA [12] and Galaxy [13]. We plan to perform user trials to evaluate our system and its user interface to real world data. We will collect additional data such as email messages, songs listened, and pictures viewed and taken. We will also optimize the current prototype implementation.

7. CONCLUSIONS

Mobile devices are now able to continuously collect real world data and present it to the users. In addition to real world data, mobile devices host structured and semi-structured information bases. We have demonstrated integration of such data with the collected real world data using a flexible and powerful RDF repository and a common ontology. We have designed and implemented a query language and engine NQ that can automatically map meaning representation produced by language systems into formal database queries. We have used NQ to access extended PIM (Personal Information Management) data on mobile phone. Our experience indicates that real-world data gathering and integration with in-device data, together with a natural language interface is a valuable addition to capabilities of mobile devices.

8. REFERENCES

[1] Baptist L. and S. Seneff, "Genesis-II: A Versatile System for Language Generation in Conversational System Applications," Proc. ICSLP '00, Vol. III, pp. 271-274, Beijing, China, Oct. 2000.

[2] Chipchase, J., "Why do People Carry Mobile Phones?", http://www.janchipchase.com/blog/archives/2005/11/mobile_essentia.html, 2005.

[3] N. Eagle, "Machine Perception and Learning of Complex Social Systems", Ph.D. Thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology, June 2005.

[4] FOAF Vocabulary Specification 0.9, http://xmlns.com/foaf/0.1/, 2007.

[5] Lassila, O. et al, "ConnectingMe", http://research.nokia.com/research/projects/connectingme/index.html, 2007.

[6] Popescu, A., Etzioni, O., and Kautz, H. 2003. Towards a theory of natural language interfaces to databases. In Proceedings of the 8th international Conference on intelligent User interfaces (Miami, Florida, USA, January 12 - 15, 2003). IUI '03. ACM Press, New York, NY, 149-157.

[7] Python for S60, http://sourceforge.net/projects/pys60, 2007

[8] Mika Raento, "Context software - A prototype platform for contextual mobile applications". In Proceedings of the International Proactive Computing Workshop. University of Helsinki, 2004.

[9] Ran, A., and Lencevicius, R., "Automating Access to Structured Data from Natural Language", Submitted for publication, 2007.

[10] Resource Description Framework, http://www.w3.org/RDF/ , 2007.

[11] S60 platform, http://www.s60.com, 2007

[12] S. Seneff, "TINA: A natural language system for spoken language applications," *Computational Linguistics*, vol. 18, no. 1., pp. 61-86, March 1992.

[13] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "GALAXY-II: A Reference Architecture for Conversational System Development," *Proc. ICSLP 98*, Sydney, Australia, November 1998.

[14] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/, 2007.

[15] Vcard, http://www.w3.org/TR/vcard-rdf, 2007.