Applying Semantic Web technology in a Mobile Setting: the Person Matcher

William Van Woensel[1], Sven Casteleyn[1], Olga De Troyer[1]

[1] Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium
{William.Van.Woensel, Sven.Casteleyn, Olga.Detroyer}@vub.ac.be

Abstract. In a mobile setting, users use, handle and search for online information in a different way. Two features typically desired by mobile users are tailored information delivery and context awareness. In this paper, we elaborate a demo application that is built upon the existing SCOUT framework, which supports mobile, context-aware applications. The application illustrates the use of intrinsic mobile features, such as context- and environment-awareness, and combines them with the use of Semantic Web technologies to integrate and tailor knowledge present in distributed data sources.

Keywords: Mobile Web, Semantic Web, Context Awareness

1 Introduction

With the increased market penetration of new generation smartphones, mobile application developers may now benefit from the additional capabilities of these devices. Because of their up-to-par computing power and screen resolution, location awareness (e.g., through GPS), identification and sensing possibilities (e.g., RFID) combined with the omnipresence of wireless internet access, a realistic opportunity presents itself to deliver qualitative and context and environment sensitive information. In a mobile setting, users are easily overwhelmed with the huge amount of information available from their surroundings, most of which is not relevant for the given user at a given time. The main challenge is thus to filter and personalize the available information, according to the context and the specific needs of the user.

SCOUT is a mobile application development framework that focuses on data acquisition from different, decentralized sources, with the aim of personalized, context-aware delivery. It supports different sensing technologies to become aware of the surrounding environment, and is primarily based on Web technologies for communication and data delivery, and Semantic Web technologies for integrating and enriching the knowledge present in the decentralized data sources. In this paper, we shortly recap the SCOUT framework, and demonstrate a mobile person matching application built on top of SCOUT. This application automatically detects other people in the vicinity, performs a detailed compatibility check based on their FOAF profiles, and notifies the user when people are encountered that could be of interest for him.

2 SCOUT in a Nutshell

The SCOUT framework aims to offer application developers support for building environment- and context-aware mobile applications. Such applications are aware of the user's context, their current (physical) environment and the objects in it, and utilize this information to offer personalized information or services to the user. The SCOUT framework consists of a layered architecture: each layer (from the bottom up) is shortly explained below. More information on the framework can be found in [1].

The Detection Layer is responsible for detecting identifiable physical entities in the vicinity of the user. The framework abstracts from the actual detection techniques employed (e.g., RFID, NFC, Bluetooth), and only assumes the detected entity is able to communicate relevant information on the entity to the framework (usually in the form of a URL, pointing to an existing Website or RDF data source).

The Location Management Layer receives raw detection information from the Detection Layer, and conceptualizes it by creating positional relationships: when an entity is determined to be nearby, a positional relation is created; when the entity is no longer nearby, the positional relation is invalidated. Determining proximity (i.e., remoteness and nearness) is done using proximity strategies, which may differ depending on the available detection data and the specific detection technique used.

The Environment Layer combines several models and services geared towards mobile context- and environment-aware application development. The Environment Model offers an integrated view on the data associated with (currently or past) nearby entities. It encompasses the User Model, which stores the user's characteristics, needs and preferences, and the Relation Model, which stores the (time-stamped) positional relationships provided by the Location Management Layer. In the Environment layer, Semantic Web technologies are exploited to represent and integrate data: RDF(S) / OWL to store and integrate the different data sources, additionally enriched by using external (semantic) sources (e.g., the Wikipedia RDF representation) and RDF(S) / OWL's reasoning capabilities; SPARQL is used to query the integrated models. The Environment Layer also provides mobile application developers with some basic services that provide access to the aforementioned models: pull-based data retrieval, where arbitrary SPARQL queries are issued over the different models using the Query Service, and push-based data retrieval, where a Notification Service monitors changes in the environment model and alerts registered applications of specific changes.

The SCOUT framework is written in JavaME (MIDP 2.0, CLDC 1.1). We employ the MicroJena API (http://poseidon.elet.polimi.it/ca/?page_id=59) to programmatically access and manipulate RDF data, and an external query service to handle SPARQL queries.

3 SCOUT Demo Application: the Person Matcher

The Person Matcher application is built upon the SCOUT framework, and utilizes some of its key features: entity identification, proximity detection, content filtering, enabling personalized data delivery. A basic implementation of SCOUT is already available (see [1]); here we shortly elaborate the implementation of the framework

features relevant for our demo, and subsequently discuss the implementation of the Person Matcher application itself.

3.1 SCOUT implementation

In the Detection Layer, two detection techniques are provided to support our Matching application: Bluetooth and RFID. Both techniques retrieve a URL from a nearby entity (which points to relevant information on the entity). In case of Bluetooth, a Bluetooth device (or the entity's Bluetooth-enabled mobile device) communicates this URL on request; in case of RFID, the URL is present on an RFID tag present nearby the physical entity, which is read by an RFID reader. In the Location Management layer, a simple proximity strategy is implemented responsible for determining when an entity is (no longer) nearby. This strategy is applied for both RFID and Bluetooth: as the detection range of the employed Bluetooth-enabled devices and RFID readers is relatively small, entities are considered nearby whenever they are in range, and no longer nearby when they move out of range. In the Environment Layer, the Notification Service is used to notify the Person Matcher application when new entities become nearby. However, in our case, only nearby persons are relevant; to achieve this data filtering, the Notification Service is additionally provided with a condition (in the form of a SPARQL query), which makes sure the application is only alerted in case the nearby entity is of the type foaf:Person. By querying the Relation Model, (all) past encounters also become accessible and available for matching. Finally, the Matcher application implements a personalized content delivery, taking into account the nearby person's FOAF profile and the user's Entity Model (containing the user's own FOAF profile), to present the user with personalized matching results. Details on this process are provided in the next section.

3.2 The Person matcher

As the mobile user is walking around, the Person matcher application is thus continuously provided with FOAF profiles of persons in his vicinity. The application subsequently calculates a "compatibility" score for each found FOAF profile, based on a comparison between the FOAF profile of the nearby person, and user's own FOAF profile, stored in the Entity Model. The algorithm used to calculate compatibility is grounded in establishing paths between the two FOAF profiles, and is based on a configurable weighting scheme which allows it to be deployed for different purposes.

The weighting scheme

The user can employ the Matcher application for different reasons (e.g., finding colleagues to collaborate with; looking for new friends); depending on this reason, some (sequences of) FOAF properties will become more important in the matching process, while others become less important or even irrelevant. For this purpose, the Matcher application can be configured with different weighting schemes (specified in RDF format), identifying relevant FOAF property sequences and their importance (between 0 and 1) in the matching process. In other words, these schemes denote *which* connections between the two persons should be searched for. We have provided a weighting scheme which is used in our demo and is aimed at finding

colleagues to collaborate with. For instance, two persons having created (foaf:made) documents with the same subject (foaf:topic) are potentially interested in collaborating together, so the weight of the sequence "foaf:made <x> foaf:topic" will be high. The user can also create his own weighting scheme, or tweak an existing one.

The matching algorithm

The actual matching algorithm looks for paths, consisting of properties identified in the weighting scheme, between the user's FOAF profile and the FOAF profile of the nearby person. These properties can be subdivided into two categories: "direct" linking properties and "indirect" linking properties. Direct linking properties *directly* link the person in question to another person, e.g., foaf:knows. Indirect linking properties connect the person to another person, via a number of intermediary resources (the maximum amount is configurable). For instance, foaf:made links a person to a resource he made, to which other persons may also link using foaf:made.
The matching algorithm constructs a graph step-wise in a breadth-first manner, starting concurrently from both persons' FOAF profiles. The nodes in this graph correspond to Persons, edges to direct or indirect links. The algorithm is able to construct some links immediately, from data found in the initial two FOAF profiles. Subsequently, the algorithm retrieves the RDF sources of encountered resources (i.e., intermediary resources, or linked persons), denoted by the rdfs:seeAlso property. These RDF sources are then examined for other relevant resources, which are also added to the graph (in the form of direct or indirect links). During this process, the algorithm stops exploring link sequences if their total score (see below) falls below a certain threshold. Thus, the algorithm recursively extends both graphs, until a predefined number of iterations are reached. A relevant "connection" is found when the two graphs get connected, i.e. when a link sequence connects the two initial Person nodes. Note that the algorithm combines information from different RDF sources, possibly resulting in new connections for which the relevant information was not present in any single source.

The compatibility score between two FOAF profiles is the sum of the individual scores of found connections between the two graphs. Each connection's score is calculated as follows (*j* is the number of direct or indirect links in the connection):

$$\text{score(connection)} = \left(\prod_{0<i\leq j} weight_i\right)\left(1 - \frac{j}{10}\right)$$

A connection's score is the multiplication of weights of its individual links. The last factor ensures that the score (additionally) decreases with the length of the path.

The user interface

The person matcher is a mobile application that, once started, continuously runs in the background. The following overviews are available: 1/ last 5 persons matched, 2/ best 5 matches of the day (figure 1a), 3/ the complete matching history. In these overviews, a detailed view of each match can be retrieved, which shows the total compatibility score, the name of the matched person, his profile picture (if available in the FOAF profile), and an overview of the connections linking the user with this person (figure 1b). Furthermore, the details on each connection can be obtained; namely, the links of which it consists and the persons present in the connection (figure 1c).

Figure 1 Person Matcher screenshots (a) (b) and (c)

The Person Matcher application utilizes the functionality provided by the SCOUT framework, and is also implemented in JavaME (MIDP 2.0, CLDC 1.1).

Related work

There already exist a number of generic Semantic Web crawlers, which follow rdfs:seeAlso properties in order to crawl a set of interconnected RDF files: e.g., RDF Crawler [2]. However, there does not exist an implementation (at least, not to our knowledge) for a mobile, JavaME environment. Tools for visualizing FOAF networks (e.g., WidgNaut, see http://widgets.opera.com/ widget/4037/) and search engines for RDF datasets (e.g., Swoogle [3]) exist, but do not focus on finding connections between two given FOAF profiles in a mobile setting, and cannot be configured to emphasize certain relationships (e.g., colleagues).

4 Conclusion

This demo paper presents the Person Matcher, a mobile application build on top of the SCOUT framework for mobile context-aware application development. It relies on the SCOUT framework to detect and retrieve relevant semantic information from nearby persons, and calculates compatibility with these nearby persons based on their FOAF profile. Both SCOUT and the Person Matcher are built utilizing Web technology for communication and content delivery, and Semantic Web technology for integrating and tailoring information for the mobile user.

References

1. Van Woensel, W., Casteleyn, S., De Troyer, O.: A Framework for Decentralized, Context-Aware Mobile Applications Using Semantic Web technology. OTM 2009 Workshops Proceedings, LNCS 5872, pp. 88-97, Springer-Verlag, Portugal (2009)
2. M. Biddulph: Semantic web crawling. In XML Europe, The Netherlands (2004)
3. Ding, L, Finin, T., Joshi, A., Pan, R. ,Cost, R.S., Peng, Y. , Reddivari, P., Doshi, V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. CIKM'04 (2004)