Applying Semantic Web Technology in a Mobile Setting: The Person Matcher

William Van Woensel, Sven Casteleyn, and Olga De Troyer

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 up-to-par computing power and screen resolution of new generation smart phones, together with location awareness support (e.g., GPS), sensing possibilities (e.g., RFID) and the omnipresence of wireless internet access, an opportunity presents itself to deliver qualitative, context- and environment sensitive information. Due to the huge amount of information available from the user's surroundings, the main challenge thus becomes to filter and personalize this information, exploiting the aforementioned context data 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 and context-aware data 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 demo, we demonstrate a mobile person matching application built on top of SCOUT. It automatically detects people in the vicinity, performs a detailed compatibility check based on their FOAF profiles and presents the results to the mobile user.

Tools already exist to find and visualize FOAF profiles; however, in contrast to FOAF visualization tools (e.g., WidgNaut [1]) or RDF search engines (e.g., [2]), the Person Matcher runs in a mobile setting, focuses on finding useful relations between two given FOAF profiles, and allows configurable weighting of relations.

B. Benatallah et al. (Eds.): ICWE 2010, LNCS 6189, pp. 506–509, 2010.

2 SCOUT in a Nutshell

The SCOUT framework consists of a layered architecture: each layer (from the bottom up) is shortly explained below. For a detailed description of SCOUT, see [3].

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, and only assumes the detected entity is able to communicate relevant information about itself (usually in the form of a URL pointing to an existing Website or RDF source). Our demo relies on two detection techniques: RFID and Bluetooth. Both techniques retrieve a URL from a nearby entity, which points to relevant information about the entity. In case of Bluetooth, a Bluetooth enabled device communicates this URL on request; in case of RFID, the URL is present on an RFID tag attached to the physical entity, which is then read by an RFID reader.

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. A single proximity strategy is employed 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.

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 (in our demo, the User Model consists of the user's FOAF profile), 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, and exploiting their reasoning capabilities to derive additional information; SPARQL to query the integrated models. The Environment Layer also provides mobile application developers with some basic services that provide access to these 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 and alerts registered applications of specific changes. Our demo application utilizes the Notification Service to be alerted of nearby entities. Furthermore, as only persons are relevant, a condition in the form of a SPARQL query ensures the application is only notified about entities of the type foaf:Person.

The SCOUT framework is written in JavaME. We employ the MicroJena API [4] to programmatically access and manipulate RDF data, and an external query server to handle SPARQL queries.

3 SCOUT Demo Application: 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, based on a comparison between the FOAF profile of the nearby person and the user's own FOAF profile (stored in the User Model). The matching algorithm is grounded in establishing paths between the two FOAF profiles, and is based on a configurable weighting scheme.

The Person Matcher is likewise implemented in JavaME (MIDP 2.0, CLDC 1.1).

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). For our demo, we have provided a weighting scheme that 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, so the weight of the sequence "foaf:made <x> foaf:topic" will be high.

The matching algorithm

The 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, that link a person *directly* to another person (e.g., foaf:knows), and "indirect" linking properties, that connect a person to another person via a number of intermediary resources (the maximum amount is configurable). E.g., 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 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 relevant resources (i.e., intermediary resources or linked persons), denoted by the rdfs:seeAlso property, and examines them for other relevant resources (i.e., similar to existing Semantic Web Crawlers, e.g. [5]), which are also added to the graph. During this process, the algorithm stops exploring link sequences if their total score (see below) falls below a certain threshold, and finishes once a predefined number of iterations is reached. A relevant "connection" is found when a link sequence connects the two initial Person nodes. Note that this algorithm combines data from a range of RDF sources, possibly resulting in new connections for which the data was not present in any single source.

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

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

A connection's score equals the product of the weights of its contained links, while the last factor ensures that the score decreases with the length of the connection.

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: i.e., the links of which it consists and the persons present in the connection (figure 1c).

Fig. 1. Person Matcher screenshots (a) (b) and (c)

4 Conclusion

This demo paper presents the Person Matcher. It relies on the SCOUT framework to detect and retrieve relevant semantic information from nearby persons, and calculates compatibility with these 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.

References

1. WidgNaut, `http://widgets.opera.com/widget/4037/`
2. 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. In: CIKM '04 (2004)
3. Van Woensel, W., Casteleyn, S., De Troyer, O.: A Framework for Decentralized, Context-Aware Mobile Applications Using Semantic Web technology. In: OTM Workshops (2009)
4. MicroJena, `http://poseidon.elet.polimi.it/ca/?page_id=59`
5. Biddulph, M.: Semantic web crawling. XML Europe (2004)