# A Pervasive Environment for Location-Aware and Semantic Matching Based Information Gathering

Riza Cenk Erdur[1], Oguz Dikenelli[1], Ata Önal[1], Özgür Gümüs[1], Geylani Kardas[2], Özgün Bayrak[1], Yusuf Engin Tetik[1]

[1] Ege University, Department of Computer Engineering, Bornova, 35100 Izmir, Turkey
{erdur, oguzd, onal, gumus}@staff.ege.edu.tr, {bayrak, tetik}@bornova.ege.edu.tr
[2] Ege University, International Computer Institute, Bornova, 35100 Izmir, Turkey
geylani@bornova.ege.edu.tr

**Abstract.** The main motivation of this paper is to integrate the semantic matching capability into the pervasive computing environments. In this context, we have developed an environment that provides a semantic matching based information gathering capability for mobile users. An important feature of the developed environment is its domain independence. Domain independence is realized by first transferring the concepts of a specific domain's ontology in XML format from the server to the mobile device, and then parsing that XML file for dynamically creating a visual interface, using which users can enter requests. The generic design of the semantic matching engine also contributes to domain independence, since a generic matching engine can accept inputs and return outputs using concepts from any ontology. To show the effectiveness of the architecture, a case study was implemented in a campus area. In this case study, mobile users can find closest places to reside or eat something.

## 1 Introduction

Advances in the enabling technologies for pervasive computing [9] have already established the infrastructure for mobile users to access the information from anywhere and anytime. Based on this infrastructure, different applications have been developed for information access and service provisioning in pervasive environments. On the other hand, Semantic web [1] describes a new vision for web computing in which knowledge is represented and processed semantically using ontologies. An ontology defines the concepts within a domain, describes the properties of each concept, defines the relations between concepts, and rules can be constructed for reasoning about concepts. Semantic matching is kind of an ontology-based information search. A semantic matching engine takes concept(s) from an ontology as input and then it returns knowledge which semantically matches the input concepts. The advantage of semantic matching is that when an exact match is not found, semantically related results can be returned to the user. In this paper, our primary motivation is to take the advantage of semantic matching based information gathering in pervasive environments.

In the pervasive computing and semantic web literature, there are pioneering studies that use semantic web technologies. For example, there are pervasive applications that use ontologies for context information modeling or semantic service discovery. Below, we will summarize the previous works by comparing them with the system that we have developed so that we can show in what ways our work is different from them.

There are studies that extend the existing service discovery infrastructures using semantic web technologies: Chakraborty, et. al. [2] implemented a semantic service discovery infrastructure that uses Darpa Agent Markup Language (DAML) to describe the services. The infrastructure contains a Prolog based reasoning engine. Masuoka at. al. [7] has taken the semantic service discovery in pervasive environments one step further and developed an application, where semantically discovered services can also be composed to achieve more complex tasks. In addition to semantic service discovery, there are studies that use ontologies for modeling context information. Wang, et. al. [10] extends the basic context information modeling by proposing an OWL (Web Ontology Modeling) encoded context ontology for pervasive environments. Chen et. al. [3] describes an ontology called as SOUPA (Standard Ontology for Ubiqutious and Pervasive Applications). Using OWL, SOUPA defines vocabularies to represent intelligent agents' beliefs, desires and intentions, time, space, events, user profiles and actions and policies for security and privacy. Although these studies use ontologies for both semantic service discovery and context information modeling, the capability of semantic matching does not exist in these systems.

On the other hand, there are many classical location-based information search services in mobile environments [4]. For example, there are systems where users with mobile phones can be directed to the nearest local restaurants, shops, etc. These systems can be considered as standard information search services for mobile users. There are two features, which make our work different from them. The first feature is being domain independent or opennes. This means that new domains can be added at any time. The visual user interfaces, which are necessary to prepare requests for querying these domains, are created dynamically at run-time by first transferring and then parsing the XML file containing the concepts belonging to that domain's ontology. Supporting semantic matching is the second feature where the system that we have developed differs from them. Using semantic matching, a result list ranked by the degree of semantic match can be presented to the user in response to his/her request so that he/she can have the option of accessing to the most semantically related information. So, we take the previous works one step further by integrating semantic matching capability into the information gathering process in pervasive environments and by modeling the system in a way that it supports domain independence.

The rest of the paper is organized as follows: Section 2 gives an example scenario to show usefulness of the proposed system. The architecture of the developed system is discussed in section 3. The semantic matching engine component is discussed in section 4. Section 5 gives an example case study, which demonstrates how the sample scenario above is realized. Section 6 includes the conclusion.

## 2 Motivating Example

As the motivating example, we give below an example scenario to illustrate how the system that is proposed in this paper can be useful:

John is a student who has been admitted to a university in a foreign country. In his new university campus and the nearby areas, there is a pervasive computing infrastructure where an information search service is provided for different domains such as accommodation, eating out, and shopping etc. He has not arranged an accommodation before, so when he visits the campus area for the first time, he asks his mobile device to list the available services given to mobile users around. First of all, he has to arrange an accommodation and chooses the accommodation domain. Whenever he chooses it, a visual user interface is created automatically and dynamically so that he can enter requests about accommodation. He chooses the concept of dormitory from the interface. He also enters other filtering criteria such as price or room capacity (e.g. he requests rooms for only one person). He was a bit late, so when he looked at the results presented, he saw that no dormitory could be found, but that the semantic matcher returned him names and addresses of some pensions with the desired characteristics and ranked with respect to distance from him. After visiting these pensions to see whether they are suitable to stay, he feels that he is hungry and chooses the eating domain from his mobile device, which results in the dynamic creation of a visual interface to prepare eating out requests by entering some concepts. He selects *RedMeatRestaurant* concept from the interface and the semantic matcher returns two red meat restaurants at top of the list and also one kebab restaurant since *KebabRestaurant* concept is semantically related with *RedMeatRestaurant* concept. Since the returned kebab restaurant is closer than red meat restaurants to him, he decides to go to the kebab restaurant to eat some Turkish kebab. The same way of information gathering is realized as he chooses other domains such as shopping, culture, etc. Finally, the same scenario can also be applied not only in campus area, but also for travelers arriving in a city, citizens looking for real estate to buy or hire in a specific city area, and such.

In this paper, we introduce a software architecture to develop a pervasive system that provides the requirements of above example scenario. The critical points of this scenario are that openness of the system architecture in terms of addition of new service domains, and the ability to gather semantically related services upon request.

## 3 System Architecture

The application that we have developed consists of three basic components. These are the mobile client component, the server component and the semantic matching engine component. The internal modules of the client and server side components, and the semantic matching engine component are shown in Fig. 1 and will be discussed in more detail in the following subsections.

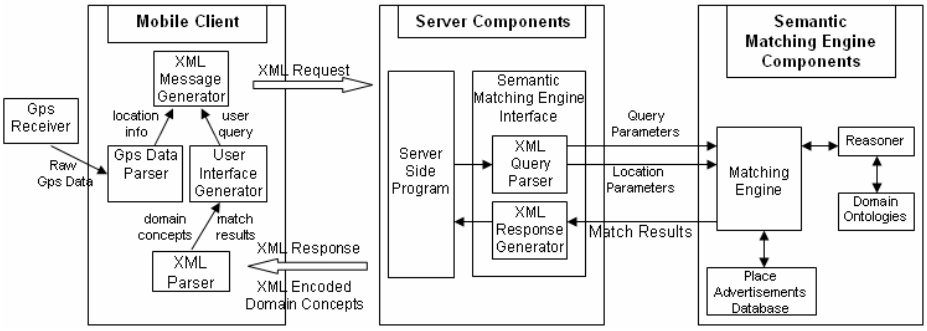


**Fig. 1.** Internal modules of client and server components

### 3.1 Mobile Client Component

The client side component is responsible for getting the GPS data, providing the interface for specifying the user requests and displaying the results, sending the request to the server in XML format and parsing the results received in XML format. The GPS data is received using the GPS receiver. The connectivity of the mobile device and the GPS receiver is provided using a Bluetooth transceiver.

The "GPS Data Parser" module is responsible for parsing the location data retrieved from the GPS receiver. Connecting the GPS receiver with a mobile device using Bluetooth starts a stream of GPS data flowing over the Bluetooth serial port. To get the most current position of the client, the GPS data parser module reads periodically the GPS message strings, which are updated every second. After getting GPS data stream, this module parses it and gets the Latitude and Longitude coordinates.

"User Interface Generator" module is critical, since the creation of visual interfaces dynamically at run-time is the responsibility of this module. The XML file containing the concepts belonging to a specific domain's ontology is transferred from the server first. Then, the transferred XML file is parsed and a visual interface is created. Hence, a user interface where users can enter their requests is created independently at run-time for each different domain. In fact, the ontologies are represented in Web Ontology Language (OWL) in the semantic matcher component. However, since mobile devices are resource limited, we simplified and represented these ontologies in simple XML format to make the parsing process efficient in the mobile device. Otherwise, the mobile device should execute the code necessary to parse OWL documents.

After the user selections are collected, they are converted into XML together with the GPS location data by the "XML Message Generator" module. We preferred the requests and results to be transmitted in XML format, since it is a well-known web standard. The "XML Message Generator" module then sends the request in XML format using a GPRS Http network connection. To send the data via GPRS Http network, the client device needs to activate the GPRS settings. As an example, if the

user selects *SeaFoodRestaurant* concept, corresponding request XML document is shown below:

```
<request>
   <PlaceType>SeaFoodRestaurant</PlaceType>
   <gpsData>
      <latitude>22.333E</latitude>
      <longtitude>52.444N</longtitude>
   </gpsData>
</request>
```

The “XML Parser” module parses the XML document that includes domain concepts initially and the incoming XML formatted results that are received from the Http server response. The incoming results are in the form of a collection, which is organized as an XML document. The parser iterates over the collection and passes the data to the “User Interface Generator” module to print the results on the user screen. As an example, a part of the result XML document is shown below. This result tells the user that a restaurant, whose name is Blue Ocean, exactly matches what he/she requests. Other details about the place such as its address, telephone, opening hours, geographical position (GPS data) and distance (in meters) from him/her are also included in the result.

```
<matchResults>
   <result>
      <name>Blue Ocean</name>
      <matchDegree>EXACT</matchDegree>
      <tel>+90-232-1111111</tel>
      <address>Bornova Street 1</address>
      <openingHours>08:30 AM - 11:00 PM</openingHours>
      <gpsData>
         <latitude>27.229E</latitude>
         <longtitude>38.455N</longtitude>
      </gpsData>
      <distanceToClient>350m</distanceToClient>
   </result>
   :
   :
</matchResults>
```

Mobile devices have a limited memory. For this reason, the “XML Parser” module has been designed to be small and light. The pull parser technique, in which the software drives the parsing, has been used. In this technique, only some part of a XML document is read at once; hence, it does not need a large memory size. The application drives the parser through the document by repeatedly requesting the next piece. Our application can process and display information as it is parsed after being downloaded from the server. In this case it basically iterates over the XML tree and finds the items. The parsed data is then passed to the “User Interface Generator” module to be printed on the screen of the mobile device.

### 3.2 The Server Component

A server side program, which in our case a Java Servlet component, meets the user request and passes it to the “Semantic Matching Engine Interface” module. The “Query Parser” sub module of this interface module decomposes the request into a

format that the semantic matching engine can understand and then sends the request to the semantic matching engine as shown in Fig. 1. It then waits for the results from the semantic matching engine. The matching results are sorted by the degree of match and location knowledge and are inserted into a collection. This collection is converted into an XML message by the "XML Response Generator" sub module. The formed XML message is then forwarded to the servlet component so that it can be sent to the requester as an Http response.

### 3.3 The Semantic Matching Engine

Semantic matching engine is a registry to keep records of knowledge about advertised places in a specific domain. It can be searched for the closest place to eat something, to reside, to buy some clothes etc. using domain dependent information. So, a place that gives a service in a specific domain, must advertise itself to the matching engine using concepts in predefined domain ontologies. For example, a restaurant that serves sea foods, must advertise itself using the *SeaFoodRestaurant* concept defined in eating place ontology given in Fig. 2.

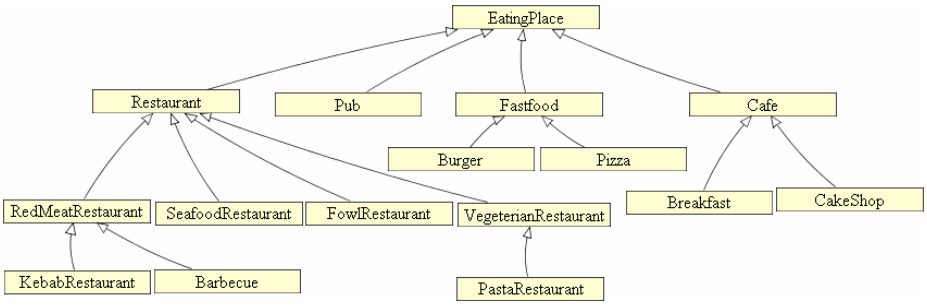


**Fig. 2.** An example *eating place* ontology to show the taxonomy of places in a domain

The basic idea behind the matching process is to find the advertised concepts that are identical to the requested one. However, the advertised and requested concepts can be semantically related with each other but are not directly identical. In this case, a semantic matching process is required. Semantic matching process is a matching process that can identify the semantic relationships between the advertised and requested concepts. Semantic matching engine is the software module that executes this process. For example, if a restaurant advertises itself to the matching engine using meat restaurant concept, then a request that searches a meat restaurant is matched with this advertised restaurant. With semantic matching process, a request that searches a kebab restaurant is also matched with the same advertised restaurant if there is a relationship between meat restaurant and kebab restaurant concepts.

In the literature, there are several studies [6], [8] proposing algorithms especially for discovery of semantic web services. We have adapted the matching algorithms for web services proposed in previous works and redesigned it for discovering places in a

specific domain. In our system (Fig. 1), mobile users can obtain the list of the *most suitable* places based on their request and global position. Requested place type in a specific domain and global position are input to our engine to realize semantic match. Each place is advertised to the engine using predefined domain ontologies to specify its type and GPS data to specify its location.

Place types are ontology classes with defined namespace URIs. In this study, we have defined a concept named as "Place" to advertise the places in different domains to the matching engine. One of the attributes of this concept is the "Place Type" which takes value from different domain ontologies and specifies the domain dependent value of the advertised places. For example, when a restaurant that serves sea foods, wants to advertise itself to the matching engine, it fills the "Place Type" attribute with *SeaFoodRestaurant* concept from the eating place ontology shown in Fig. 2. The other critical attribute is the "GPS data" that defines the geographic position of the advertised place and it takes value from GPS ontology. The other attributes are name, address, telephone and opening hours. The "Place Advertisements Database" module of the semantic matching engine stores all the places in a particular area (e.g. campus) with their GPS Data.

The "Matching Engine" module realizes matching of mobile user's request and advertised places and produces the list of the suitable places. Consider the types of requested and an advertised place are represented with $C_1$ and $C_2$ respectively. According to the matching algorithm in [8], the "Matching Engine" module determines four types of match degree between these two concepts:

- *exact match* when $C_2$ and $C_1$ are equal or $C_1$ is subclass of $C_2$
- *plug-in match* when $C_2$ is more generic than $C_1$ ($C_2$ subsumes $C_1$)
- *subsumes match* when $C_2$ is more specific than $C_1$ ($C_1$ subsumes $C_2$)
- *fail* when neither of the conditions above satisfies

The semantic matching engine has a "Reasoner" module for determining ontology class relations. It gives the superclass distance of the two ontology classes with given URIs using the relations between them. For doing this, it parses OWL ontologies and finds subsumption relations. For example, consider the following simple ontology class tree in Fig. 2. According to the ontology model, the "Reasoner" module finds superclass distances as -1, 0, 1 and 2 in (SeaFoodRestaurant, CakeShop), (SeaFoodRestaurant, SeaFoodRestaurant), (SeaFoodRestaurant, Restaurant) and (SeaFoodRestaurant, EatingPlace) ontology class pairs respectively. In case of a multiple inheritance, there will be different paths from a subclass to its superclass. In this case, it returns the shortest distance as a result by performing a depth-first search on the ontology tree. Calculated ontology class distances are cached as instances in the "Reasoner" to optimize performance. When the same distance query is received multiple times they are all responded (excluding the first one) via it's cache.

The "Matching Engine" module determines the match degree between the requested and advertised place types using the superclass distances found between corresponding ontology classes. This is realized with the following pseudo code in the "Matching Engine" module:

```
If distance = 0 or distance = 1 then EXACT match
If distance > 1 then PLUGIN match
```

```
If distance < 0 then calculate reverse_distance
   (reverse_distance means parameters in reverse order)
   If reverse_distance > 0 the SUBSUMES match
   Else FAIL in match is determined
```

The scoring function of the "Matching Engine" is ordered as exact > plug-in > subsumes > fail. The engine sorts out advertised places according to their semantic degree of match with the given requested type. The GPS data of the places are used as tie-breaker to sort the places that have the same match degree. So, the *semantically equal* match results are sorted according to their distances to the user. To keep implementation and test simple, only latitude and longitude tags of a GPS data are evaluated and distance between two geographic points is calculated using a formula considering those points' latitude and longitude values [5].

## 5 Case Study

As a case study, we have implemented a scenario like the one illustrated in the section 2. To realize the test scenario, we have walked through the campus with the GPS receiver, detected the GPS data for the major places for eating and accommodation. Then, we have advertised these places to the semantic matching engine using "Place" concept. For the "Place Type" attribute of this concept, we have defined two example ontologies using OWL; one for accommodation and the other for eating places (Fig. 2). We also mapped a simplified version of these ontologies to XML format and stored in the server so that whenever a specific domain is selected, this XML file can be transferred to the mobile device for the creation of the visual interface at run-time. Finally, we have formed different requests using the mobile phone in various places of the campus and observed what the matching engine returned us.

For identifying user locations, a GPS receiver has been used. Since the application is initially implemented for campus area, which is an open area, the GPS receiver satisfied our needs. However, different location identification technologies such as cell-id based identification can be added to our system whenever a need arises. The connectivity of the mobile phone and the GPS receiver is provided via Bluetooth technology.

User screens are usually limited in mobile devices; hence, the user interfaces created are not so complex. The primary user interface window provides view of all data received along with user-selectable menu choices for controlling the application. When the user starts application, he/she first selects a domain and then a place type using the taxonomy of places within this domain. The terms representing the types of the selected place come from that domain's ontology. Then, the system will return results sorted by the degree of semantic match. Also, the system can give detailed information about returned places when the user selects a matched result.

First of all, the client application on mobile phone connects to the server and gets the available service domains. In parallel with the scenario, let us assume that the user selected the eating domain, then following the taxonomy of places within this domain, he/she selected *RedMeatRestaurant* concept to find places that instances of this concept. Screen snapshots showing user's selection of his/her request are given in

Fig. 3. So, the *RedMeatRestaurant* concept and the position of the user send to the server and the semantic matching engine.


**Fig. 3.** Screen snapshots showing user's selection of his/her request

Again assuming that there are suitable restaurants, after the semantic matching process, the engine returns a list of three restaurants sorted by match degree and distance from the user. Then, the user selects returned results to see detailed information about them. Screen snapshots showing match results list and details of each returned place are given in Fig. 4.


**Fig. 4.** Screen snapshots showing match results list and details of each returned place

As shown in Fig. 4, first two of the returned restaurants have degree of exact match with the user's requested place type. However, assume that the user is very hungry and he/she selects the third one though it has degree of subsumes match. But, it's closer to the user than the other restaurants.

## 6 Conclusion

We think that integrating semantic web technologies into pervasive environments will help to implement more novel services for mobile users. In this respect, we have implemented a semantic matching based information search service for mobile users. Using the application that we have developed, we have observed that having the

ability of finding the most semantically related information, the semantic matching process increases the quality and relevance of the information presented to the users.

As an immediate future work, we plan to present the results in different colors according to the degree of match on a map in the mobile device. Towards this direction, we have already begun to prepare a map of the campus.

## References


1. Berners-Lee, T., Hendler, J. and Lassila, O.: The Semantic Web, Scientific American, 284(5), (2001), 34-43
2. Chakraborty, D., Perich, F., Avancha, S. and Joshi, A.: Dreggie: Semantic Service Discovery for M-commerce Applications, in the 20th Symposiom on Reliable Distributed Systems, (2001)
3. Chen, H., Perich, F., Finin, T. and Joshi, A.: SOUPA:Standard Ontology for Ubiquitous and Pervasive Applications, In the Proc. of the First Annual Conference on Mobile and Ubiqutious Systems: Networking and Services, Boston, MA, (2004)
4. Deitel, "Wireless Internet and Mobile Business: How to Program", Prentice Hall, (2002)
5. Green, R. M.: Textbook on Spherical Astronomy, 6th ed. Cambridge, England: Cambridge University Press, (1985)
6. Li, L. and Horrocks, I.: A Software Framework for Matchmaking based on Semantic Web Technology, In the proc. of WWW'2003, Budapest, Hungary, (2003), 331-339
7. Masuoka, R., Labrou, Y., Parsia, B. and Sirin, E.: Ontology-Enabled Pervasive Computing Applications, IEEE Intelligent Systems, Vol.18, No.5, (2003), 68-72
8. Paolucci, M., Kawamura, T., Payne, T. R. and Sycara, K.: Semantic Matching of Web Services Capabilities, In the proc. of the first international semantic web conference (ISWC), Sardinia (Italy), (2002)
9. Satyanarayanan, M.: Pervasive Computing: Vision and Challenges, IEEE Personal Communications, vol.8, no.4, (2001), 10-17
10. Wang, X. H., Zhang, D. Q., Gu, T. and Pung, H. K.: Ontology based Context Modeling and Reasoning using OWL, In the Proc. of Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, (2004)
11. The OWL Services Coalition: Semantic Markup for Web Services (OWL-S), (2004), available at http://www.daml.org/services/owl-s/1.1/