Automated Retrieval of Semantic Web Services: A Matching Based on Conceptual Indexation

Hadjila Fethallah and Chikh Amine
Computer Science Department, UABT University - Tlemcen, Algeria

Abstract: Web services are taking an important place in the distributed computing field, as well as in the electronic business. In this work we present an initial research which deals with the issue of automated service retrieval. For that, we propose an approach which exploits the service interface (inputs/outputs) and the domain ontology, in order to index conceptually the web services, after that, we compute a similarity score between the request and the indexed web services through the cosine measure. An experimentation based on the OWLTC test collection is shown to evaluate the system. The obtained results are very encouraging and confirm the satisfiability of the solution.

Keywords: Web service research, similarity measure, ontologies, OWLS, information retrieval.

Received October 17, 2010; accepted January 3, 2011

1. Introduction

The web service technology is a concrete instance of the SOA paradigm; It is noteworthy to mention that the majority of the existing web applications is planned to undergo a reengineering process [8], in order to profit from the advantages of SOA (reusability, interoperability, integration...).

The web services are based on a set of recommendations that include Universal Discovery Description Integration (UDDI) [27], Web Services Description Language (WSDL) [10] and Simple Object Access Protocol (SOAP) [28]. SOAP is designed to provide a transport mechanism for the XML based message, WSDL is model used to describe the service interface. However, neither SOAP nor WSDL allow for the automatic location of web services on the basis of their capabilities. UDDI is a registry that describes businesses by their characteristics such as name, and other public informations. The registry describes also the services through the industrial categories. In addition, UDDI descriptions are augmented by a set of attributes, called Tmodels, which describe additional features such as classification of services within taxonomies, abstract specifications...etc. Since UDDI does not represent service capabilities, it can’t help the functional search for services.

A drawback that surrounds XML based standards, such as those mentioned above is their lack of explicit semantics by which two identical XML descriptions could mean totally different things, depending on the context of their use. The opposite case is also possible (i.e.: the presence of two xml documents that have same meaning and different structures). These shortcomings limit the capability of matching Web services. To resolve this problem, we must add semantic knowledge to support the identification of the most suitable service for a particular task.

The integration effort of semantics into Web services started with the RDF [20] language and evolved with the creation of RDFS [9], OWL [14] and OWLS [15]. Different approaches were developed to resolve the web service searching problem [5, 16, 19, 25]. In this paper we propose, an approach based on the owls content, we exploit, more precisely the “profile:hasInput”, and the “profile:hasOutput” elements to index the services and the requests. Furthermore we employ a set of domain ontologies to enhance the semantic of these elements. This process is called “conceptual indexation”, we will detail it in the fourth section.

The searching process of a web service can be summarized as follow:

For each web service S of the base, we build 02 concepts vectors Si and So which model S in the matching step.

Si contains the service inputs and their subsumers, which are extracted from the domain ontology, we note also that each subsumer is associated with its frequency. For that we use the domain ontology associated to S.

So contains the same thing, except that it works on outputs, rather than inputs.

The same thing is applied to the requests, and we have as result, two vectors Ri and Ro (a request is modeled as an owls document).

The matching process is implemented by the cosine measure which computes a similarity score between the services vectors and the requests vectors. A threshold θ (fixed by the user) is used to filter the results, according to their closeness degree.
Finally we evaluate the system quality through the recall and the precision metrics. We have chosen the measure cosine (which is a space vector similarity measure), because it is one of the most prominent measures of the information retrieval domain [11]. The rest of the paper is structured as follows: Section 2 presents some background on service retrieval approaches; the section 3 presents the OWLS specification. In section 4 we present our proposal for the web service retrieval problem. Section 5 shows the experimentation results; we discuss the obtained results in section 6. Finally in section 7, we expose some open issues and conclude this paper.

2. State of the Art

Different Web service matchmakers have been developed in the literature, such as the MAMA [12], HotBlu [13, 18], OWLS-MX [19], RACER [21], SDS [22], and OWLSUDDI matchmaker [25]. A lot of them use the subsumption test and the description logics to compare the (I/O) of the profile with the user’s request. We mention also another set of approaches [1, 7] which use only the WSDL interface as an entry (without semantic interfaces like owls). The SecDisc approach [1] utilizes two Algorithms: the level-based matching is used to compare syntactically the WSDL concrete parts, and the sequence-based schema matching approach [2] is used to compare the WSDL abstract parts, in the same optic, the approach quoted in [23] uses a linear discrimination function associated with the wordnet thesaurus [29], to compare a set of WSDL files. The system proposed in [6] adopts a service process-model matching. To reinforce the retrieval scalability, the approach developed in [3] makes a peer to peer discovery of web services; Bansal and Vidal [4] apply a recursive tree matching to achieve the retrieval problem. The project [17] Proposes a set of mediators to make a semantic discovery of WSMO based web services, the matching is done through the Web Service Modeling Language (WSML). The METEOR-S approach [26] enhances the WSDL standard, with semantic descriptors in order to make an efficient search. LARKS [24] and OWLSMX [19] provide a hybrid search, in the sense that they exploit both explicit (formal) and implicit (informal) semantics by complementary means of logic based and approximate matching. The OWLS-MX [19] matchmaker proposes four variants of the matching Algorithm, the first is purely logic and the others are hybrids (i.e, they combine the content based metrics used in the information retrieval in addition to the logic subsumption). We note that each variant uses a set of filters and produces seven scores of matching. The purely logic-based variant OWLS-M0 (the first Algorithm) is similar to the OWLS-UDDI matchmaker [25] but differs from it as follows: Firstly, the latter adopts a different notion of plug-in matching, and does not provide additional subsumed-by matching.

3. OWLS (Ontology Web Language Services)

OWLS [15] is a high level ontology, which allows an abstract description of a service. It is composed of a root class named “Service”, and it directly corresponds to the actual service which is described semantically (every service that is described, corresponds to an instance of this concept). The “Service” class is linked with three other classes. The first is “ServiceProfile”, it specifies the functional properties as well as the QOS based attributes of a service, it gives also an informal description about the exposed capabilities, the second is “ServiceModel”, it is an orchestration part, that specifies the data flow and the control flow of the service. The third class is named “ServiceGrounding”, its role is to define the manner to access a service. It shows also the equivalent element in the WSDL model, for each atomic process Figure 1 Shows the upper ontology OWLS.

4. Contribution

The data used in our experimentations are sampled from the OWLS-TC\(^1\) corpus version 2.2.1 this base is developed by the German research center for artificial intelligence (http://www.dfki.de/scallops). This base uses the owls documents, to describe a set of web services. These documents involve in their profile part, a set of elements named « profile: hasinput » and « profile: hasoutput ». These elements are used during the conceptual indexation of the web services. The owls documents are segmented in seven classes: economy, communication, education, food, weapon, medical care, and travel. In this paper we have used only two requests (these requests, correspond to the classes that have the greatest number of documents).

\(^1\) http://www.dfki.de/scallops
We note that the request is modeled as a web service, i.e., it owns an OWLS document with a «profile: hasinput» and a «profile: hasoutput» elements.

The searching process follows these consecutive stages:

The indexation stage:
1. For each service, we extract its inputs and outputs.
2. We build a vector Vi which contains the inputs and their subsuming concepts in the ontology (each concept will have a number which represents its frequency in the vector). For example if the inputs are \{c1, c2\}, and c1 has \{c3, c4\} as subsumers, and c2 has \{c5, c4\} as subsumers, then \(v_i = \{c1/1, c2/1, c3/1, c4/2, c5/1\}\) (because c4 has two occurrences).
3. We make the same thing for the outputs (we compute Vo).
4. The same treatment of indexation (step 2 and 3) is applied to the request (we compute Ri and Ro).

The matching stage:
1. We compare the indexed request with the indexed services, by using the cosine similarity measure (as given by the following formula): \(\cos(A, B) = \langle A, B \rangle / (||A||.||B||)\), where \(\langle A, B \rangle\) denotes the scalar product of A and B. A and B are two vectors of concepts. This step is realized as follow:
   a. Score1 = \(\cos(Ri, Si)\). Score1 stands for the similarity between Ri and Si.
   b. Score2 = \(\cos(Ro, So)\). Score2 stands for the similarity between Ro and So.
   c. Then, Score = (Score 1+Score 2)/2.
2. Thereafter, we sort, the results (from the greatest score to the weakest score).
3. We retain only the services whose score exceeds a certain threshold \(\theta\) (the threshold \(\theta\) is a numerical value chosen by the user, \(\theta \in [0, 1]\)).

Finally, we evaluate the results through the criteria of recall and precision.

5. Experimentation

The following paragraph defines the criteria of recall and precision, which are used in the results evaluation.

The precision P can be thought of as the 'signal to noise' ratio, it is the fraction between the pertinent results retrieved by the system and the totality of results. And more formally:

\[P = \frac{Tp}{Tp+Fp}\]

With:

- \(Tp\): the number of pertinent results which are positively classified
- \(Fp\): the number of impertinent results which are positively classified.

The recall R, can be thought of as the 'hit ratio', it is the fraction between the pertinent results retrieved by the system and the totality of the pertinent results and more formally:

\[R = \frac{Tp}{Tp+Fn}\]

with:

- \(Fn\): the number of pertinent results which are negatively classified.

In this experimentation we consider two requests R1 and R2 (see Table 1):

<table>
<thead>
<tr>
<th>Name</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car_price_service</td>
<td>Car Price</td>
<td></td>
</tr>
<tr>
<td>Groceries_store_food_service</td>
<td>Grocerystore Food</td>
<td></td>
</tr>
</tbody>
</table>

The Table 2 shows a part of the results associated to the first request R1, in this example we set \(\theta\) to 0.8. The 7th column represents the decision taken by our system, while the 8th column stands for the decision taken by the human expert.

<table>
<thead>
<tr>
<th>Service Name</th>
<th>The service inputs</th>
<th>The service outputs</th>
<th>Score1</th>
<th>The service outputs</th>
<th>Score2</th>
<th>Score</th>
<th>The system decision</th>
<th>The user’s decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>3wheeledcar_price</td>
<td>3wheeledcar</td>
<td>Price</td>
<td>0.953</td>
<td>Price</td>
<td>1.0</td>
<td>0.976</td>
<td>Accept</td>
<td>Accept</td>
</tr>
<tr>
<td>1personbicyclecar_price</td>
<td>4WheeledCar, _1personbicycle</td>
<td>Price</td>
<td>0.809</td>
<td>Price</td>
<td>1.0</td>
<td>0.904</td>
<td>Accept</td>
<td>Reject</td>
</tr>
<tr>
<td>car_price</td>
<td>Car</td>
<td>Price</td>
<td>1.0</td>
<td>Price</td>
<td>1.0</td>
<td>1.0</td>
<td>Accept</td>
<td>Accept</td>
</tr>
<tr>
<td>3wheeledcar_year_Recommendedprice</td>
<td>3WheeledCar, Year</td>
<td>Recomended price</td>
<td>0.725</td>
<td>0.5</td>
<td>0.612</td>
<td>Reject</td>
<td>Accept</td>
<td></td>
</tr>
<tr>
<td>_3WheeledAuditCarprice</td>
<td>_food_Exportservice</td>
<td>Price</td>
<td>0.0</td>
<td>1.0</td>
<td>0.5</td>
<td>Reject</td>
<td>Accept</td>
<td></td>
</tr>
<tr>
<td>drugstore tea</td>
<td>Drugstore</td>
<td>Tea</td>
<td>0.463</td>
<td>0.0</td>
<td>0.231</td>
<td>Reject</td>
<td>Reject</td>
<td></td>
</tr>
<tr>
<td>grocerystore_butterquantity</td>
<td>Grocery Store</td>
<td>Butter, quantity</td>
<td>0.463</td>
<td>0.0</td>
<td>0.231</td>
<td>Reject</td>
<td>Reject</td>
<td></td>
</tr>
<tr>
<td>grocerystore_food_service</td>
<td>Grocery Store</td>
<td>Food</td>
<td>0.463</td>
<td>0.0</td>
<td>0.231</td>
<td>Reject</td>
<td>Reject</td>
<td></td>
</tr>
<tr>
<td>retailstore_foodquality_service</td>
<td>Retail Store</td>
<td>Food, quality</td>
<td>0.467</td>
<td>0.0</td>
<td>0.233</td>
<td>Reject</td>
<td>Reject</td>
<td></td>
</tr>
</tbody>
</table>
Table 3. The results of R2.

<table>
<thead>
<tr>
<th>The service name</th>
<th>The service inputs</th>
<th>Score1</th>
<th>The service outputs</th>
<th>Score2</th>
<th>Score</th>
<th>The system decision</th>
<th>The user’s decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>3wheeledcar_price</td>
<td>3wheeledcar</td>
<td>0.442</td>
<td>Price</td>
<td>0.0</td>
<td>0.221</td>
<td>Reject</td>
<td>Reject</td>
</tr>
<tr>
<td>1personbicycle_car_price</td>
<td>4WheeledCar, 1personbicycle</td>
<td>0.625</td>
<td>Price</td>
<td>0.0</td>
<td>0.312</td>
<td>Reject</td>
<td>Reject</td>
</tr>
<tr>
<td>car_price</td>
<td>Car</td>
<td>0.463</td>
<td>Price</td>
<td>0.0</td>
<td>0.231</td>
<td>Reject</td>
<td>Reject</td>
</tr>
<tr>
<td>3wheeledcaryear_Recommendedprice</td>
<td>3WheeledCar, Year</td>
<td>0.336</td>
<td>Recommended price</td>
<td>0.0</td>
<td>0.168</td>
<td>Reject</td>
<td>Reject</td>
</tr>
<tr>
<td>3WheeledAudiCarprice_</td>
<td>-</td>
<td>0.0</td>
<td>Price</td>
<td>0.0</td>
<td>0.0</td>
<td>Reject</td>
<td>Reject</td>
</tr>
<tr>
<td>food_ExpertService_</td>
<td>-</td>
<td>0.0</td>
<td>Food</td>
<td>1.0</td>
<td>0.5</td>
<td>Reject</td>
<td>Reject</td>
</tr>
<tr>
<td>drugstore_tea</td>
<td>Drugstore</td>
<td>0.985</td>
<td>Tea</td>
<td>0.932</td>
<td>0.958</td>
<td>Accept</td>
<td>Accept</td>
</tr>
<tr>
<td>grocerystore_butterquantity</td>
<td>Grocery Store</td>
<td>1.0</td>
<td>Butter, quantity</td>
<td>0.745</td>
<td>0.872</td>
<td>Accept</td>
<td>Accept</td>
</tr>
<tr>
<td>grocerystore_food_service</td>
<td>Grocery Store</td>
<td>1.0</td>
<td>Food</td>
<td>1.0</td>
<td>1.0</td>
<td>Accept</td>
<td>Accept</td>
</tr>
<tr>
<td>retailstore_food_quality_service</td>
<td>Retail Store</td>
<td>0.992</td>
<td>Food, quality</td>
<td>0.912</td>
<td>0.952</td>
<td>Accept</td>
<td>Accept</td>
</tr>
</tbody>
</table>

The Table 3 shows a part of the results associated to the second request R2, in this example we set θ to 0 to 0.8. The 7th column represents the decision taken by our system, while the 8th column stands for the decision taken by the human expert.

6. Discussion

The following Figures (2 and 3), show the relation between the performance (precision/recall) and the threshold θ. The more the threshold θ is large, the more the precision is good, and the more the recall is mediocre. For R1 we notice that a threshold comprised between 0.5 and 0.9, induces a reduction in the precision and the recall. This situation is caused by the degradation of Tp and the increasing of fn. We notice also that, If θ = 1, then the precision = 1 and the recall is minimized.

For R2, we note an increasing in the precision, which will be equal to 1 beyond 0.6, this situation is caused by the elimination of Fp. In a general manner we can note that the range [0.7, 0.8] is a good compromise (for the precision and the recall). The main difficulty of the approach is to set the threshold θ, this problem can be resolved by setting θ heuristically, or by learning the parameter, from the precedent experiences (or the user’s feedback).

Another drawback is the time needed to build the model, associated to the services/or the requests, this drawback has a strong link with the selected similarity measure.

It is worth observing that, the main advantage of the system, in comparison with the other approaches is:

1. The adaptation of the best space vector similarity measure to the semantic context.
2. The minimization of Fp and Fn: the adopted process of indexation eliminates the ambiguity in the computation of the similarity score, however the bipartite graph matching approaches such as [24], can’t easily bypass this problem.

7. Conclusions

In this paper, we have presented a method that exploits OWL-S to build a web service searching system. This method uses the service inputs/outputs and the ontologies to index semantically, the web services,
then it exploits the cosine similarity measure to select the relevant solutions. Future work will mainly consist of augmenting the requests number, that have to be tested, and furthermore, an interesting aspect to deal with, is the study of the similarity measures performance, the term performance will include the precision, the recall and the time.

References


**Hadjila Fethallah** received his MS degree in computer science from the UABT University of Tlemcen in 2003. Now he is currently a researcher in the Department of Computer Science, UABT University Tlemcen, Algeria. His research interests include: web service discovery, service composition, ontology matching, and artificial intelligence.

**Chikh Amine** received his PhD degree in 2005. He became an Assistant Professor in the Department of Computer Science, UABT University of Tlemcen since 2006, His research interests include: decision aiding, artificial intelligence, biomedical applications.