Web Services Semantic Searching enhanced by Case Reasoning

Ling Wang             Jinli Cao
Department of Computer Science & Computer Engineering
La Trobe University, Melbourne Australia
l15want@students.latrobe.edu.au, j.cao@latrobe.edu.au

Abstract

Web services integrate various business application systems to provide worldwide platform to serve customers directly over the Internet. As the increasing number of business applications joining into the integration, the service activities are involving more and more complex situations. The enormous workload and the complex business processes involved by a request cannot be dealt with the simple request-answer model. It is highly desirable that Web services system can support vast clients’ requests efficiently, effectively and promptly. This paper provides a searching mechanism to serve clients intelligently. It utilizes an efficient matchmaker to discover clients’ requests by reusing past experiences. The matchmaker is enhanced by an automated discovery algorithm. It uses not only OWL-S for identifying different matching levels by related domain ontologies but also is enhanced with reuse mechanism of case-based reasoning (CBR) with a formula for similarity. The proposed matchmaker takes advantages of semantics and CBR techniques to improve the efficiency and effectiveness of Web services searching.

Keywords: Web services, Matchmaker, Similarity, Case-based reasoning.

1. Introduction

With advances in the service-oriented technologies service providers can offer services to clients in various domains such as e-business, healthcare and entertainment etc on the Internet. However, the efficient and effective processing and management of the information services in a large scale distributed and complex environment create new challenges for the research community of the Web services. A Web Service is any service that is accessible over the Internet. The Web services with their current form WSDL (Web Services Description Language) [1] service descriptions and UDDI (Universal Description, Discovery and Integration) [2] search provide some mechanisms for integration of individual Web service providers.

As the number and variety of services continue to grow, the complex business processes involved by a request cannot be dealt with the simple request-answer model of Web applications. Comparing the services provided by participating service providers based on certain rating criteria in order to provide an optimized composition of service is a crucial task. The existing discovery approaches used in UDDI are ineffective [14, 15] since search and matching lack of semantic knowledge. Web services should be able to understand and interpret data into useful information without human involvement. It is desirable that UDDI is not only used for registration, publication and discovery but also provides the best matching and the optimal service solution to clients. Ontology Web Language (OWL)-based Web service ontology (OWL-S) facilitates the automation of Web service tasks [4]. Many researches are working on the OWL-S/UDDI using ontology to automate Web service discovery, execution, composition and interoperation [3, 8, 9, 10, 11, 16, 17, 21]. Researchers [3, 8, 16, 17] tried to extend the existing UDDI registry in order to augment existing discovery mechanism with semantics. The work in [8, 17] presented a semantic search mechanism enabling clients to specify semantic inquiries based on web services capabilities. They proposed a matching algorithm with a planning functionality.

However, utilization of OWL-S may provide effective solutions while the processing and management of the Web services such as selection, composition, and interoperation of service providers to perform some complex tasks are very expensive and time consuming. They have failed the inefficiency and feasibility of matchmaker in their proposals. The completed optimized compositions of services are not utilized. Many different clients ask for the similar
services following the 80-20 rules. That is, 80% of clients ask for the same requests on 20% of the composite of Web services. This gives a suggestion that we should reutilize the previous results generated by the OWL-S/UDDI Matchmaker to avoid wastes of resource and improve the response time.

To address the problem, this paper concentrates on the investigation of an efficient matchmaker CBR/OWL-S to discover clients’ requests by reusing past experiences. In order to target desired Web services, the matchmaker CBR/OWL-S utilizes ontology technology for the Semantic Web search and a numerical evaluation formula for case-based reasoning. In addition, this paper also illustrates a travel case to show that the ideas can be realized in a real world situation.

The rest of the paper is organized as follows. Section 2 reviews UDDI discovery and discusses an OWL-S/UDDI Matchmaker from related work. Then the section 3 proposes a CBR/OWL-S Matching Engine within OWL-S/UDDI Matchmaker for efficient retrieval of best matching to client’s request. In section 4, a case of traveling will be demonstrated by using the proposed CBR/OWL-S Matching Engine. The testing program to simulate the proposed algorithm is presented. The last section summarizes the main contribution of this paper with discussions on further issues.

2. Related work

This section discusses the techniques used for UDDI searching and OWL-S/UDDI matching.

Web services discovery is provided by facilitating a keyword search on UDDI registries and using the reasoner component to examine the discovered services capabilities. UDDI is a standard for registration, navigation and location of Web services. Web services provider advertises their services with the UDDI registry. The registry stores the service information for future use. Requester searches clients’ requests from the UDDI registry. The UDDI registry compares the request with advertisements to match to allow requesters select the Web services provider that best fits their needs.

However, it is less developed with semantic capability and is limited to value sets and direct matching of values [3, 4, 8, 17]. To address the situation, OWL-S has been integrated with UDDI for semantic discovery.

The existing Matchmaker’s architecture [3, 7, 8] combines OWL-S and UDDI into one. The UDDI is used to store advertisements and retrieve information about Web services. OWL-S exploits the ontology and match capabilities. UDDI is a platform-independent, centralized warehouse, and registry for businesses worldwide services on the Internet. OWL-S profile descriptions are stored inside the UDDI registry with mappings between the OWL-S profiles and the UDDI records. The OWL-S/UDDI Matchmaker uses RACER [6] to perform OWL inferences [7].

![Fig.1 The architecture of the OWL-S/UDDI Matchmaker [8]](image)

The matchmaker shown in Fig. 1 receives advertisements and requests through the communication module. If a message is an advertisement it is passed to the OWL-S/UDDI Translator. The UDDI records the advertisement and the capability description of the advertisement is sent to the OWL-S Matching Engine. By identifying a request, the communication module sends it to the OWL-S Matchmaker Engine for capability matching. The result of the matching is the selected advertisement of the providers and a reference to the UDDI service record.

The OWL-S Matching Engine consists of four components. Advertisements are stored in the Advertisements Database component and indexed using ontologies. By receiving a request, the Matching Engine component selects advertisements that are relevant for the current request, and then uses the OWL-S Reasoner and ontologies to compute the degree of match and select the advertisements that match the request best.

For example, a service provider sells pet food, and a requester is looking to buy dog food. A UDDI registry would be unable to match the request because keyword matching is not enough to identify the relationship between pet food and dog food. However, an implicit relationship existed between pet and dog. Using the OWL-S Reasoner, it infers that dog is a type of pet, and the OWL-S/UDDI Matchmaker could recognize a semantic match between the request and the advertisement.
3. CBR/OWL-S Matching Engine

In this section we further develop the Matching Engine described in previous section by introducing an extra component namely case-based reasoning (CBR) which is named as CBR/OWL-S Matching Engine. CBR/OWL-S Matching Engine utilizes CBR technique with an algorithm for efficient retrieval of best matching to client’s requests.

CBR is a problem solving technique that reuses past cases and experiments to find a solution to the problems [9]. CBR is used to firstly retrieve the similar solutions from the case base according to the situation when the problem occurred. Secondly, considering the differences between the new problem and the previously-solved problem, it adjusts the solving strategy of the previously-solved problem to fit the new problem. The basic idea is to retrieve and adapt previous cases when solving a new problem [10].

3.1 Architecture of CBR/OWL-S Matching Engine

The new matching engine CBR/OWL-S Matching Engine will replace the OWL-S matching engine in OWL-S/UDDI Matchmaker discussed in section 2. The architecture of CBR/OWL-S Matching Engine is depicted in Fig. 2.

![Fig. 2. CBR/OWL-S Matching Engine](image)

The core of the CBR/OWL-S Matching Engine is the Case Reasoning Engine component with an embedded OWL-S reasoner. By receiving a request, the matching engine selects the cases that relevant to the current request from the Case Knowledge Base component, and the Case Reasoning Engine will compute the degree of matching. There may have two situations of matching:

If the request is matched, the matched case will be selected from the Case Knowledge Base and sent back to the requester. This matched case was optimized composition of Web services which is the best service to the client.

If the request can not be matched, the matching engine has to start the searching for a new case of Web services to satisfy the request:
- Select advertisements that are relevant for the current request.
- Call the OWL-S Reasoner using ontologies to compute the degree of match.
- Select the highest match degree to the request in order to get an optimized composition plan for the required services and send back to the client.

The Case adjust/add Engine will add the new case or modify old cases in the Case Knowledge Base after getting all matched services information such as contact information, business categorization, identifier, etc. from UDDI registry. The update of cases enriches Case Knowledge Base that will extend the capability of the Case Knowledge Base and improve the semantic searching more effectively. Therefore, it can provide better quality and more effective services to clients.

The Case Knowledge Base and Advertisements Database both take advantages of the ontologies Database to index cases and advertisements for fast retrieval at matching time [3, 8, 17].

3.2 The Algorithm for Case Reasoning Engine (ACRE)

To find the best match case, the case reasoning engine computes the similarity for the degree of match between the requested case and old cases stored in case knowledge base (CKB). The algorithm of the case reasoning engine adopts REMIND’s [12, 13] formula to compute the similarity of the cases with the utilization of ontologies of related domains for similarity comparisons.

For each case \( f \) we define \( n \) dimensions that can accurately describe the case’s properties. Every dimension of the case has been assigned a weight \( W \) which is depending on the importance level of the dimension for the case. A dimension is called critical if the dimension can decide the fate or failure of a case. The critical dimensions can be declared by clients or by semantics of the case. For example, a travel plan to overseas could not be implemented if one cannot find a transport even the accommodation can be booked. Therefore, the critical dimensions have the highest weight. The similarity of a new requested case and old case from CKB is calculated using the formula (1). The formula of computing similarity is adopted and modified from [12, 13]:

```latex
\text{Similarity} = \sum_{i=1}^{n} w_i \cdot (1 - \text{distance}(x_i, y_i))
```

where \( w_i \) is the weight of the dimension \( i \), \( x_i \) and \( y_i \) are the values of the dimension \( i \) in the reference case and the new case, respectively.
similar( \( f^I, f^R \) ) = \( \left( \frac{\sum_{i=1}^{m} W_i \times \text{sim} ( f^I_i, f^R_i )}{\sum_{i=1}^{m} W_i} \right) \)

Where

- \( f^I \): input case,
- \( f^R \): retrieved case,
- \( W_i \): weight of the \( i \)-th dimension of cases \( f \) or \( f^R \),
- \( f^I_i \): dimension \( i \) of input case,
- \( f^R_i \): dimension \( i \) of retrieved case,
- \( \text{sim}(f^I_i, f^R_i) \): similarity of a single dimension \( i \) between \( f \) and \( f^R \), and
- \( m \): the total number of dimensions that \( \text{sim}(f^I_i, f^R_i) \) \( \neq 0 \). This is different with the formula from [12] where the number of total \( n \) dimensions was used.

The aggregate of match score is computed by summing the products of the importance of each dimension multiplied by the degree of match of dimensions in the case. The algorithm of Case Reasoning Engine (ACRE) is described as follows:

The algorithm ACRE(\( f^I, [f^I]_{n} \), index of CKB, \{ rules \})

Input: \( f^I \) a requested case,
\([f^I]_{n} \), a vector of \( n \) dimensions of \( f^I \)
the index of the case knowledge base
\{ rules \} a set of matching rules based on ontologies of each dimension for computing the \( \text{sim}(f^I_i, f^R_i) \)

Output: \( s \): similarity of input case versus retrieved case by formula (1)

BEGIN

\( s=0; i=0; m=0; \)

WHILE ( \( i < n \) ) {

Call function \( \text{sim}(f^I_i, f^R_i) \) to calculate similarity of dimensions using \{ rules \}
IF \( \text{sim}(f^I_i, f^R_i) = 0 \) and dimension \( i \) is critical
RETURN \( s=0; \) // failure of similarity, no more calculation
ELSE
IF \( \text{sim}(f^I_i, f^R_i) = 0 \)
\( W=\sum_{j=1}^{n} W_j - W_i; \)
ELSE
\( s = s + W_i \times \text{sim}(f^I_i, f^R_i); \)
\( m++; i++; \)
\} // end of while

Compute similarity \( (f^I, f^R) \) \( s = s/ W \)

return \( s; \)

END

The similarity result of formula is depending on the result of \( \text{sim}(f^I_i, f^R_i) \). That is, for each pair of dimensions \( I \) we must define the degree of its match in order to compute the similarity of the cases. The embedded OWL-S Reasoner in Fig. 2 is responsible for computing the similarity of each dimension \( i \) \( (f^I_i, f^R_i) \). The rules are based on OWL ontologies for reasoning the level of the match. Inspired by [8] we have classified 5 degrees of comparison for \( \text{sim}(f^I_i, f^R_i) \):

Exact - If dimension \( f^I_i \) and dimension \( f^R_i \) are equivalent, called an Exact match.

Subclass - If dimension \( f^I_i \) is subclass of \( f^R_i \), that is, the concept \( f^I_i \) is the descendent of \( f^R_i \) in the ontology relationship (described in OWL-S for each dimension domain), called a subclass match.

Superclass - If dimension \( f^I_i \) is superclass of \( f^R_i \), that is, the concept \( f^I_i \) is the ancestor of \( f^R_i \) in the ontology relationship, called a Superclass match.

Overlapping - If dimension \( f^I_i \) and dimension \( f^R_i \) are overlapping, called the Overlapping match

Fail - If dimension \( f^I_i \) and dimension \( f^R_i \) are not relevant, called a fail match.

Let us assume that the Exact is valued as 1.0, and Subclass corresponds to 0.8, which Superclass corresponds to 0.6, that Overlapping corresponds to 0.4, and Fail corresponds to 0. Therefore the function \( \text{sim}(f^I_i, f^R_i) \) can be defined as follows:

Function \( \text{sim}(f^I_i, f^R_i) \)

\{ 
RETURN 1 if dimension \( f^I_i = f^R_i \)
RETURN 0.8 if dimension \( f^I_i \subset f^R_i \)
RETURN 0.6 if dimension \( f^I_i \supset f^R_i \)
RETURN 0.4 if dimension \( f^I_i \cap f^R_i \neq \emptyset \)
RETURN 0 if dimension \( f^I_i \cap f^R_i = \emptyset \)
\}

4. Similarity Evaluation

To valid the proposed algorithm we have simulated the traveling services with Java programming. The case knowledge base is setup with 500 traveling cases that is used for matching search. The geographical ontology, transportation ontology and reasoned tour timetable and activities of traveling are predefined. We have tested with over 100 travel requests. The matching results are outperformed.

The program Similarity computes the similarity of requesting cases and retrieving cases from two input files to find the most matched old case with the highest similarity. The matching of similarity is 100% accurate with the predefined ontologies if the cases are in the CKB.

The case knowledge base (CKB) starts with the help from domain experts. We have extended and augmented the CKB by recording the not matched
cases. It is also noticeable if some of cases are only partially matched the partial result should still be applied to new cases. This will save the expensive searching on the OWL-S/UDDI. In the worse case, the original OWL-S/UDDI matchmaker will be called. However, the resulted new case will be added to CKB.

5. Conclusion & Future direction

In this paper, we proposed a Web services matchmaker that plays a role like an OWL-S/UDDI Matchmaker but is augmented with case-based reasoning, matching and ranking capabilities according to past experiences. The features of the matchmaker are utilizing the reuse capabilities of CBR with a formula for similarity; and relative domain ontology to identify different matching levels. Finally, a travel study case has demonstrated the feasibility and effective of the approach.

Compared with related researches on Web services matching, the proposed Web services matchmaker utilizes CBR and OWL-S to improve searching efficiency and effectiveness of Web services. The designed matching algorithm is tested and demonstrated with good result.

Reference