Searching and Ranking the Suitable Web Services with the Ontology-Based Measurements

Chun-Jung Lin, T. Chin Lo, Fan Wu*
Institution for Information Industry, Taipei, 101, Taiwan
Department of Information Management, National Chung-Cheng University, Chia-Yi, 621, Taiwan
*Corresponding author. Tel: +886-5-2721500; fax: +886-5-2721501.
Email address: kfwu@mis.ccu.edu.tw (F. Wu).

Abstract: One of the major problems for seamlessly electronic business is how to find a suitable web services. Only the syntax and semantic comparison do not precisely find the suitable web services for they are procedures embedded with a complicated thought. In this paper, we propose an effective approach based on the ontology to solve this problem. With the help of ontology-based metrics, we can measure a web service matching degree to a given request and determine the rank in which the advertisement matches the request. Simulations are also performed, and the results show that our method can have a good precision and recall rate.

Keywords: Web Services, Ontology-based metrics.

I. Introduction

The web services stack of standards is designed to support the reuse and the interoperability of software components on the web [1]. The promise of web services is to enable a distributed environment in which any number of application components can interoperate seamlessly among organizations in a platform-neutral and language-neutral fashion [2]. In the Service-Oriented Architecture (SOA), service providers develop the web services and publish the services to the UDDI (Universal, Description, Discovery and Integration) registry. When service requesters need some services, they search for the web services in the UDDI registry. If the requester finds a suitable web service, he can send SOAP (Simple Object Access Protocol) messages to invoke the web services.

Since there are many web services issued by different corporations, a critical step to find the suitable services is how to discover and find suitable web services against the request. The current services discovery approach in the UDDI registry is based on categories and keywords, as relies on the shared common understanding of services providers and requesters. Generally, the selection of keywords and the classification of categories for a same object are different for different individuals. The found web services based on the above criterion will not always precisely match the request [1]. For example, if we want to find a DogSelling service, it is clear that a PetSelling service does not match the request since the keywords, DogSelling and PetSellling, do not match with each other in syntax. In fact, Dog and Pet have a strong relationship in semantics. But the WSDL (Web Service Description Language) used to describes the characteristics of the web service inherently can not embed the semantics of keywords in its structure.

To embed the semantic of a web service, the Web Ontology Language (OWL) is proposed as a description language of web services [9]. With the emergence of OWL, the web services can be described not only syntactically but semantically. OWL-S is another more powerful descriptive language for the web service, which can represent an OWL-based Web service ontology. The OWL-S can supply Web service providers with a core set of markup language constructs for describing the properties and capabilities of those Web services in an unambiguous and computer-interpretable form [10]. Though the web description is more expressive, the matching between the request and the web provider is not discussed until now. In this paper, we use OWL-S to complement WSDL for web services descriptions and propose an approach to make matchmaking between the requirements and the advertisements of the web services.

A web service can be formally defined as three main elements: {S, C, E}, where S, C and E denote the content descriptions, the capabilities, and the properties of end points respectively [11]. The content descriptions refer to what the web service is commented about such as the text description of the service; the capabilities refer to what the functionalities that the service provides such as what requirement is needed for performing the service and how the service performs; the properties of end points are related to the information and constraints of the end point such as cost, payment ways, response time, etc. The main elements of web services can be described by the service description language such as WSDL or OWL-S. If these main elements of the requested services are similar to those of the advertised services, we can allege that they are matched.

Former researchers consider what requirement is needed for performing the service and how the service performs as the service capabilities [3, 6]. However, the capabilities of web services become more complex and can not be just defined according to all inputs and outputs of the web services. In this paper, a complete web services discovery approach, which compares the degree of similarity between requirements and advertisements according to the whole main elements of web services, is proposed. The proposed approach is a three-metric matching approach, including text matching, construction matching, and parameter matching.

Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 971 - 976.

Each metric is used for examining one kind of main elements of web services between requirements and advertisements and then calculates their matching scores. In addition to the complete consideration of matching criterion, the proposed approach is based on the semantic matching.

II. Previous Work

II. 1 Component Comparison and OWL-S

Since a web service accepts inputs and sends off output after processing, the task to discover web services in a UDDI registry is similar to the task of software component retrieval in a component library. For the automatic software component search and retrieval, the existing approach can be divided into four different methods [7], namely, simple keyword and string search, faceted classification and retrieval, behavioral matching and signature matching. The method of the simple keyword and string search to find the components is based on the frequency of the keywords occurring in the components. The drawback of the method is the lack of precision since the keywords cannot fully delineate the whole ability of the software component. The method of faceted classification and retrieval is to classify components based on taxonomies such as functions the software performs, types of the system, and so on. The method can have more insights about the software. However, it is difficult for software component developers to appropriately classify components based on taxonomies since that some components may overlap several categories. The third method, i.e. behavioral matching, executes each component with random inputs and generates outputs. The comparison between expected outputs and the actual outputs can help to select the matched component. But it can not perform well when the software components have complex behaviors. Precisely, assume that components have many functions. These functions may affect each other, and the behavioral matching will have low precision. The last method, i.e. signature matching, is the comparison of the function types and argument types between the components and the query specified by the user, as is the practical approach. In the area of web services discovery for web services discovery, many researchers also claim that services discovery should base on the match between a declarative description of the service being sought and a description of the service being offered [1, 3, 4, 5, 6]. Note that in this paper, the declarative description of the service being sought is called “requirement” and the one of the service being offered is called “advertisement”.

Ontology defines the common words in a specified domain and the relationships between the words to represent the knowledge of the specified domain. The OWL Web Ontology Language is the W3C recommended language used to define ontology. The purpose of OWL is to describe the knowledge in specified domains and the relationships between words in each domain can be interpreted by a machine without human support. OWL has more facilities for expressing semantics than XML (eXtensible Markup Language) and RDF (Resource Description Framework), and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. There are three sublanguages provided by OWL to define ontologies with three different degrees of expressive abilities. The three sublanguages are OWL Lite, OWL DL and OWL Full. OWL Full is the powerful expressive one and OWL Lite is the simplified expressive one. For each of these sublanguages, the preceding one is the descendant of the rear one. What can be expressed validly in the descendant can also be expressed validly in the elder generation.

OWL-S (Web Ontology Language for Services) is developed by the OWL Services Coalition for describing the ontology of web services. OWL-S defines the overall structure of the ontology and is divided into three parts: Service Profile, Service Process Model and Service Grounding. The part of Service Profile defines the fundamental information about this service such as service name, service text descriptions, service category, provider contact information, etc; the part of Service Process Model contains the information about how the service is constructed; the last part of Service Grounding is to define the details of how to access the service included the communication protocols, the message formats, etc. In this paper, we utilize the Service Profile since it provides the information about content descriptions, capabilities, and end points information and the Service Process Model which provides the information about service construction to make matchmaking.

II. 2 Web Services Discovery and Semantic Matching

Paolucci [3] proposed a semantic matching approach that is based on the service capabilities and defined four matching degrees, namely, exact, plug in, subsume and fail, respectively. For explaining how the approach works, we use Rout and Aout to represent an output of the request and an output of the advertisement. They use four matching degrees, namely, exact, plug in, subsume, and fail, which are in descending order. Precisely, exact is the best matching degree and fail represents not matched.

Wang [1] also proposed the flexible interface matching for web service discovery. The description language used in the proposed approach is WSDL. The approach needs the service requester to provide the ideal service description and then find the matched services in the repository. They first use the vector space model to compare the information within the <documentation> tag described in natural language between the requirement and the advertisement. After the first stage, there are lists of ranked candidate web services returned. Then they use the structure matching to refine the quality of the candidate service set. The criteria used by this approach are the content descriptions, capabilities and service construction. However there is no semantic concept in this approach because of the used description language, WSDL.

Paolucci [3] proposed the semantic web services matching based on services capabilities. Many researchers

have cited the proposed concept for making a semantic web services matchmaking. The criterion used by this approach is the service capability. The proposed approach is good at the matching of the service capabilities but is weak for using only one criterion for matching.

Sirin [6] proposed two filters for selecting semantic web services including the input/output matching and the parameter matching. The proposed approach uses the criteria of capabilities and properties of end points to make a semantic web services matching. However the lack of the criteria including content descriptions and the service construction will reduce the precision and recall of the matching. The following table shows the comparison of these approaches.

Table 1. The comparison of the related research

	Service description language	Criteria	Matching approach
Wang [1] 2003	WSDL	Content descriptions; Capabilities; Construction	Vector-space model; Structure matching
Paolucci [3] 2002	DAML-S	Capabilities	Semantic matching approach
Sirin [6] 2003	OWL-S	Capabilities; End point information	Semantic matching approach; Parameter matching

III. Ontology-based Comparison Approach

This paper proposes a three-metric matching approach for finding suitable web services for requesters according to the criteria of content descriptions, service capabilities, service construction and properties of end points. The three metrics includes text matching, construction matching, and parameter-matching. The first metrics can filter the most irrelevant advertisements in the registry and give the qualified services the ranking. The remaining two metrics, i.e. the construction matching and the parameter matching, are used to refining the initial ranking of the qualified advertisements. Finally, lists of ranked web services are returned to the requester.

Before the matchmaking, the requesters are assumed to figure out what web services they need in an OWL-S format. In other words, the requester should provide an OWL-S file describing the ideal service. Then the proposed approach compares the description with the advertisements in the registry and returns the required web services. The proposed metrics work sequentially when the comparison of the degree of similarities between the requirement and each advertisement are executed. Each metric can produce one matching score. Therefore there are three matching scores produced for each advertisement. Then the relevant web services are returned according to the degree of the similarities. The following is the metrics used in the proposed approach.

III. 1 Text Matching

Many methods for text matching have been proposed in the field of IR. The simple and realistic method is the vector space model [12]. In the Service Profile of OWL-S, there is an optional tag, <profile:textDescription>, in which the service providers can have some comments in natural language for their services and the requesters can describe what their ideal requirements are. According to the vector space model, the part commented in natural language can be treated as a multi-dimensional vector. Each term in the part can represent one dimension in the vector. The weight of each dimension in the vector is directly proportional to the frequency that the term occurs in the document and inversely proportional to the number of documents which contain the term. The weight can be computed according to the following formulas:

$$W_{ij} = tf_{ij} idf_i ;$$

$$idf_i = \log_2 (N / df_i) ;$$

Where W_{ij} represents the weight of the dimension i in the vector. tf_{ij} represents the number of occurrences of the term i in document j. idf_i represents the result of the total number of documents divided by the number of documents which contain the term i. N represents the total number of the documents. df_i represents the number of documents which contain the term i.

For restraining the impact of idf_i, $\log_2$ is used to dampen the effect relative to idf_i [13]. Thus we have two vectors respectively for the requirement and the advertisement, and we can derive the similarity coefficient of the two vectors from the production of them. For each advertisement, our approach will return a similarity coefficient derived from the comparison. The derived similarity coefficient is the matching score of the text matching.

III. 2 Construction Matching

The construction matching is the consideration of which processes the inputs and outputs belong to. In the construction matching, we utilize the information provided by Service Process Model of OWL-S. The comparison of the construction is a multi-step process: it involves the comparison of the processes between the requirement and the advertisement, which is based on the comparison of the compositions of the inputs and outputs within the processes. In the input/output matching, we compare all the inputs and outputs between the requirement and the advertisement. In

the construction matching, we compare all of the inputs and outputs within one process in the requirement to all of the inputs and outputs within one process in the advertisement. Therefore we can get the matching score of each pair-wise process between the requirement and the advertisement. The pair-wise processes can form many kinds of combinations like the combinations in the input/output matching. Then we find the combination with the highest score among all the combinations. Finally we can define the highest score of the construction matching between the requirement and the advertisement as the construction matching score.

III. 3 Parameter Matching

The parameter matching utilizes the information provided by Service Profile of OWL-S to examine the matching score of the properties of end points between the requirement and the advertisement. The parameter matching can be considered as adding the constraints into the matching processes for finding the suitable web services. The parameter in the parameter matching can be divided into two parts. The first part is the OWL-S predefined parameters such as contactInformation and ServiceCategory. The comparison of parameters in this part needs to find the corresponding parameter name between the requirement and the advertisement and then compare the value of each attribute of the parameter. If the value of each attribute of the parameter in the requirement is the same with the value of each corresponding attribute of the corresponding parameter in the advertisement, we define that the two parameters between the requirement and the advertisement are matched. We define the matching scores according to the numbers of the matched parameters. Each matched parameter desires 10. The second part is the self-defined parameters such as the service quality rating. The comparison of parameters in the part needs to first compare the parameter names between the requirement and the advertisement. If the parameter names are the same, we can compare the value of the parameters. Because the value of the parameter is an URI referring to the class in the ontology, we use the semantic matching approach to compare the value of the parameter. As the above mention, the matching degree of the Exact level is defined as score 10; the Plug-in level is defined as score 6; the Subsume level is defined as score 4; the Fail level is defined as score 0. Finally we accumulate each matching scores and the total score represent for the matching degree of the parameter matching.

III. 4 Ranking System

The three metrics introduced above can produce three scores for representing the matching degree between the requirement and the advertisement. First we can utilize the two scores derived from text matching and input/output matching to filter the most irrelevant advertisements. We use the formulation to representing how it works.

$$\text{Relevant Score} = W_1 * S_1 + W_2 * S_2 ,\text{where}\, W_1 \text{ and } W_2$$

are used for normalizing S_1 and S_2 to avoid either of them influence the relevant score too much. S_1 and S_2 denote the scores derived from text matching and input/output matching respectively.

We can set a threshold for the relevant score. If the relevant score between the requirement and the advertisement is higher than the threshold, the advertisement is considered as qualified for the candidates. After we have the candidates, we need to rank them according to their matching degree. We use the following formulation to represent how to do it.

$$\text{Matching Score} = \text{Relevant Score} + W_3 * S_3 + W_4 * S_4 ,$$

where W_3 and W_4 are used for normalizing S_3 and S_4 to avoid that either of them influence the matching score too much. S_3 and S_4 denote the scores derived from construction matching and parameter matching respectively. We can rank the qualified advertisements according to the matching scores in descending. Therefore we can return the suitable ranked services to the service requester.

IV. Evaluation Method

The proposed three metrics can be implemented to develop the matching system. We use Java language to implement the system. The OWL-S API developed by MINDSWAP (Maryland Information and Network Dynamics Lab Semantic Web Agents Project) helps us to parse the OWL-S document [14]. The RACER system [15] is an OWL reasoner system. It helps us to identify the relation of the input/output between the requirement and the advertisement and we can proceed the semantic matching and define the matching score of the input/output matching and the construction matching. At present the general web service description language is WSDL. However, our approach uses the OWL-S for the web service description language. We need to translate WSDL to OWL-S first and then we can use our system to make a matchmaking. The WSDL2OWL-S developed by the Softagents Lab of Carnegie Mellon University can help us to translate the WSDL document to OWL-S document. The whole system architecture is shown in figure 1.

To evaluate the proposed approach, we have to collect lots of WSDL documents of web services as the advertisements. The XMethods [16] is a web site which provides lots of web services for users to give a trial. We can collect the WSDL documents from the web site to form the advertised collection. Then we classify the collection into several categories. We take one advertisement from one of the categories as the requirement. After executing the comparison of the proposed approach, the matching system will return lists of suitable web services. If the returned service is same category with the requirement, we define that it is relevant. Otherwise, the returned service is not

relevant. Based on the definition of the relevance, we can compute the precision and recall for our proposed approach. The way to compute the precision and recall is defined as follows:

Precision = # of Relevant retrieve / # of Retrieve;

Recall = # of Relevant retrieve / # of Relevant.

The two metrics are chosen for the experiments. In the search for the web service, we limit up 5 UDDI server to be accessed. All the retrieval web services are ranked, and only the top 30 web services are returned by our web service search engine through the proposed criterion. The simulation results for the requests of "translation_from_mile_to_kilometer" and "translation_from_French_to_English" are summarized in Figures 2 and 3, respectively. From these two tables, we can find some interesting characteristics. First, we found the retrieval for the request "translation_from_mile_to_kilometer" has higher precision than that for the request the request "translation_from_French_to_English". Since the former request has less descriptive vocabularies than that of the second one. Perhaps the metric of the text matching will let more possible web services to enter the second and third metric comparison. And the second and third metric can have a better rank ability for the simple request. However, the recall rate for the former request is smaller than that of the latter one. It is guessed that the former has less characteristic than the latter. The judge of similarity in the former is more difficult than the latter.

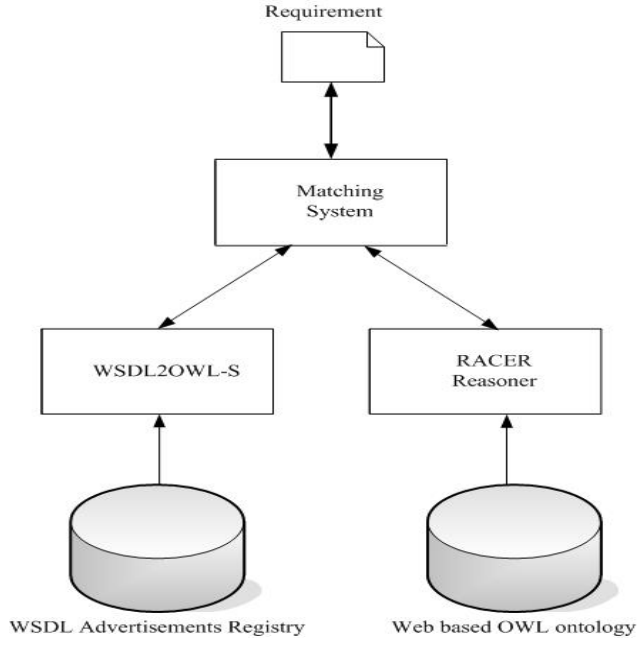
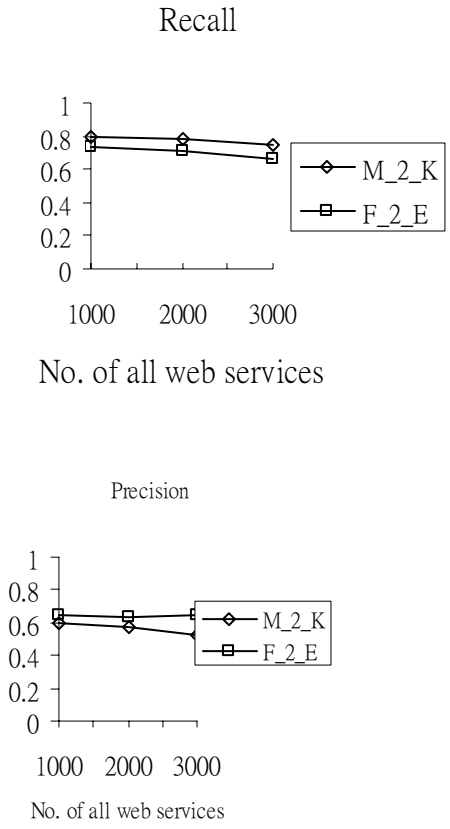


Figure 1. The system architecture



Figures 3 and 4. The recall and precision rates of web service searching for the M_2_K (Mile_to_Kilometer) and F_2_E (French_to_English).

V. Conclusion

As the application of Web services grow exponentially, it has become a crucial problem to provide effective search tools to increase the speed and precision of searching for the suitable ones. The credit assignment for the ranked search can be significantly improved by adequate metrics. In this paper, we propose a method to assign the order of web services for a request. The extensive simulation is performed and shown that our method can effectively estimate the proper suitability of web services.

References

[1] Y. Wang and E. Stroulia, "Flexible interface matching for Web-service discovery," in *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03)*, 2003, pp. 147-156.

[2] D. A. Chappell and T. Jewell, *Java Web Services*, O'REILLY, 2002, pp. 63.

[3] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," in *Proceedings of the First International Semantic Web Conference on The Semantic Web (ISWC)*, 2002, pp. 333-347.

[4] L. Li and I. Horrocks, "A software framework for matchmaking based

on semantic web technology," in *Proceedings of the twelfth international conference on World Wide Web*, 2003, pp. 331-339.
[5] M. Paolucci and K. Sycara, "Autonomous Semantic Web services," *IEEE Internet computing*, vol. 7, no. 5, Sep.-Oct. 2003, pp. 34-41.
[6] E. Sirin, B. Parsia, and J. Hendler, "Filtering and Selecting Semantic Web Services with Interactive Composition Techniques," *Semantic Web Services*, vol. 19 no. 4, July-Aug. 2004, pp. 42-49.
[7] V. Sugurmaran and V. Storey, "A Semantic-Based Approach to Component Retrieval," *ACM SIGMIS Database*, vol. 34, no. 3, 2003, pp. 8-24.
[8] S. A. McIlraith and D. L. Martin, "Bringing semantics to Web services," *IEEE Intelligent Systems*, vol. 18, no. 1, Jan.-Feb. 2003, pp. 90-93.
[9] OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/
[10] OWL-S 1.1 Release. http://www.daml.org/services/owl-s/1.1/
[11] X. Gao, J. Yang, and M. P. Papazoglou, "The Capabilities Matching of Web Services," in *Proceedings of IEEE 4th International Symposium on Multimedia Software Engineering (MSE'2002)*, 2002, pp. 56-63.
[12] D. A. Grossman and O. Frieder, *Information Retrieval Algorithms and Heuristics*, Kluwer Academic Publisher, 1998, pp.13-18.
[13] G. Salton, A. Wong and C. S. Yang. "A vector-space model for information retrieval", *journal of the American Society for information Science*, vol. 18, November 1975, pp. 13-620.
[14] OWL-S API. http://www.mindswap.org/2004/owl-s/api/
[15] V. Haarslev and R. Möller, "RACER System Description," in *Proceedings of International Joint Conference on Automated Reasoning*, 2001, pp. 18-23.
[16] Xmethods. http://www.xmethods.com