Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 4 / ISSUE 2 / AUGUST 2009 / 79-87
© 2009 Universidad de Talca - Chile

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200007

A Method of Web Service Discovery based on Semantic Message Bipartite Matching for Remote Medical System

Yang Zhang[1], Bing-Yue Liu[2] and Hong Wang[3]

Neusoft Institute of Information, China, Dalian
[1] zhangyang@neusoft.edu.cn,[2] liubingyue@neusoft.edu.cn,[3] wanghong@neusoft.edu.cn

Received 13 January 2009; received in revised form 3 June 2009; accepted 15 June 2009

Abstract

The number of Web services are growing at an explosive speed, which brings great challenges to the accurate, efficient and automatic retrieval of target services for users. This paper presents a service discovery method with semantic matchmaking which could be used in remote medical systems. Adding ontology related semantic annotations to service interfaces is considered, and a method of service discovery based on bipartite matching of semantic message similarity is proposed. The method is easy to implement because it is not limited to specific service model. It also contributes to the improvement of service discovery efficiency when service is retrieved in an automatic way.

Key words: Web Service Discovery, Remote Medical Systems, Message Bipartite Matching, Semantic, Similarity

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200007

1 Introduction

Network technologies offer new opportunities for wide adaptation of new medical technologies and development of telemedicine or remote medical systems. By making use of these technologies, we can quickly gather information and process it in various ways in order to assist with making diagnosis and treatment decisions immediately and accurately no matter where the patient may geographically be in the world.

According to the features of remote medical solution over the Internet, Service-Oriented Architectures (SOA) and Web service technologies have been proposed to respond to some interoperability challenges of heterogeneous medical systems. Web service technology incorporates the strengths of distributed computing, grid computing, Extensible Makeup Language (XML) and so on. By adopting XML-based specifications such as Web Service Description Language (WSDL), Universal Description, Discovery and Integration (UDDI) and Simple Object Access Protocol (SOAP), Web service technology is well suited to the integration and implementation of heterogeneous remote medical systems by providing interoperability among diverse applications and platform and language independent interfaces.

The number of Web services is growing at an explosive speed, which brings great challenges to the accurate, efficient and automatic retrieval of target services for users. Some semantic matchmaking approaches have been proposed to support automatic service discovery [1], [5] - [7], [14]. In [1], semantic temporal and security constraints for service discovery are considered. The work in [7] extends existing approaches by supporting semantic approximate matching and IR techniques. The work in [14] proposes (Quality of Service) QoS-based selection of services. In [6], the use of graph transformation rules for specifying both queries and services is proposed. The work in [5] suggests behavior matching for service discovery based on similarity measures of graphs.

2 Example Web Services in Remote Medical System

We motivate our automatic service discovery method through the following example.

A web service, *w*, has typically two sets parameters: $w^i = \{i_1, i_2,\ldots\}$ for SOAP request (as input) and $w^o = \{o_1,o_2,\ldots\}$ for SOAP response (as output). When *w* is invoked with all input parameters, w^i , it returns the output parameters w^o. We assume that in order to invoke w, all input parameters in w^i must be provided (i.e, w^i are mandatory).

Consider the two Web services in table1 illustrated in WSDL notation:

Table1: Example Web Services

Description	[1] <message name="findDoctor_Request"> [2] <part name="disease" type="xsd:string"> [3] <part name="location" type="xsd:string"> [4] <part name="isExpert" type="xsd:boolean"> [5] </message> [6] <message name="findDoctor_Response"> [7] <part name="hospital" type="xs:string"> [8] <part name="phone" type="xs:string"> [9] <part name="email" type="xs:string"> [10] </message>	[11] <message name=" getPhysician _Request"> [12] <part name="sickness" type="xsd:string"> [13] <part name="region" type="xsd:string"> [14] </message> [15] <message name=" getPhysician _Response"> [16] <part name="email" type="xs:string"> [17] <part name="phone" type="xs:string"> [18] </message>
Name	findDoctor	getPhysician

1. Given the disease type, current location and expert or not, findDoctor returns the hospital, phone and email.

2. Given the sickness type and current region, getPhysician returns the email and phone.

Now, consider the following requests:

1. r1: sequential input parameters are "pleurisy, Beijing, true", and sequential output parameters are the hospital, phone and email of a doctor.

2. r2: sequential input parameters are "arthritis, Shanghai", and sequential parameters are the email and phone of a doctor.

3. r3: sequential input parameters are "arthritis, Shanghai, true", and sequential output parameters are the phone and email of a doctor.

Given a request r with input parameters r^i and output parameters r^o, finding a set of web services $w \in W$ such that $r^i \supseteq w^i$ and $r^o \subseteq w^o$ is referred to as the Web Service Discovery(WSD) problem.

To fulfill r1, invoking the web service findDoctor is sufficient, to fulfill r2, invoking the web service getPhysician is sufficient. However to fulfill r3, it seems that either findDoctor or getPhysician is not suitable, because that the names and sequence of parameters between request and service are different. The input parameters of r3 are sickness, region and isExpert, while the input parameters of findDoctor are disease, location and isExpert. They are different literally although they have the same semantic meaning. The ordinal output parameters of r3 are phone and email, while the ordinal output parameters of findDoctor are email and phone. They are different in the sequence.

Assume only syntactic (text-based) matching among web services and requests, input of r3 could not match with input of findDoctor. Assume only sequential matching between parameters, output of r3 could not be fulfilled. So we use our method to solve WSD problem in a combine way, which using semantic message bipartite matching. Web service matching problem is transformed into a semantic message bipartite matching problem.

The semantic message bipartite matching is denoted in figure 1. It shows the matching result of output parameters between r3 and findDoctor, in which output of r3 and output of findDoctor are matched in a bipartite way. Semantic similarity of output parameters between request and service are considered and the bold lines show that O_1^r matched with o2 and o_2^r matched with o3. In the same way, we could get the matching result of input parameters between r3 and findDoctor, in which i_1^r, i_2^r and i_3^r matched with i1, i2 and i3 respectively. As far as the algorithm of semantic message bipartite matching, it would be discussed in section 4.

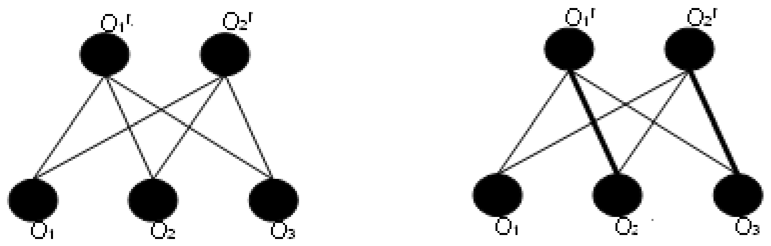

Figure 1: Bipartite Matching and Result

3 Related Work

Using semantic web technologies in a medical consultations domain has its obvious advantages. The traditional Web Service discovery algorithm based UDDI was dependant on the technology of keyword searching. Its efficiency and auto-discovery capability are lower. The matching Web Services returned by these algorithms were not always suitable for user's queries. The semantic Web Service discovery algorithms just can resolve those questions. Machine understandable documents are provided by the development of relevant ontology (shared vocabularies for a domain). If users commit to a particular descriptive regimentation of a domain, the data is not only available to machines for unambiguous interpretation, but widely understood by users of the data.

Much of the recent research has focused on remote medical consultations based on Service-Oriented Architectures and Web service technologies. But only a few researches study on semantic service discovery in remote medical consultations, now they are described as follows.

Nigel Shadbolt develops an application for the MIAKT project to provide support for medical decision making [10]. It is built around a novel distributed architecture which uses web-based services to provide discrete and disparate functionality to a generic client application. Automatic service discovery is supported through the use of remote service repositories and description mining. Available services are automatically mapped to tasks in the task invocation system, thereby publishing them for use to the client application. Services are automatically inserted into the generic client application by examining their input and output roles.

Nadine Fröhlich presents the CASCOM agent-based approach to semantic service discovery and coordination [4]. The main objective of CASCOM was to develop, implement, validate, and trial of intelligent, context-aware agent-based service coordination infrastructure for innovative Semantic Web service discovery, composition, and execution across mobile and fixed peer-to-peer service networks. The CASCOM architecture has been extensively evaluated using an emergency assistance application in mobile eHealth environment. The evaluation of the system within the challenging emergency assistance healthcare domain indicated that it can be efficiently used in practical settings and provided valuable feedback for further improvement.

Cesar Caceres focuses on service discovery for medical-emergency management [3]. A new mechanism for semantic service discovery complements existing approaches by considering relevant parts of the organizational context in which e-health services are used, to improve system usability in emergencies. This approach is reusable in other application areas, especially in the medical field.

The approach to automatic Web Service discovery of Glue is described by Emanuele Della Valle describe and Dario Cerizza[12]. COCOON Glue is a prototype of WSMO Discovery engine for the healthcare field. In WSMO, goals and Web Services can be annotated using different ontologies and the relationships between them can be captured by the mean of wgMediators. This approach is similar to an early work on OWL-S, in which the service discovery problem is formulated as a rewriting problem where requests are attempted to be rewritten in terms of available services, but it takes a much easier approach in coding the matching rules directly in the wgMediator. Applying WSMO in the healthcare field is that the clear separation between the ontologies used by each entity simplifies and speeds up the gathering of consensus, which is always difficult to reach in large groups, and especially in healthcare field.

ARTEMIS [2] is a European Commission supported STREP project in the 6th Framework program that aims to provide the interoperability of medical information systems through semantically enriched Web services [13]. ARTEMIS aims to develop a semantic Web Services based interoperability framework for the healthcare domain. ARTEMIS has a peer-to-peer architecture in order to facilitate the discovery of healthcare web services. In ARTEMIS, healthcare institutes are represented as peers. ARTEMIS peers provide interfaces to the healthcare information systems to enable them to discover and consume the Web services provided by the other peers.

This paper focuses on the discovery method with semantic matchmaking which could ease the way to use UDDI registered Web services. The core of the method is to consider the Web service matching problem as a semantic message bipartite matching problem. Later an algorithm of service discovery based on bipartite matching of semantic message similarity is proposed, which can fully use the semantic information of the interfaces of Web services. The method is easy to implement because of using WSDL standard instead of some specific service languages or models.

The remainder of the paper is organized as follows. In Section 4, a method of Web service discovery based on semantic message bipartite matching is introduced. In section 5 we conduct experiments to validate the efficiency of our service discovery method. In Section 6, we summarize our work and discuss future work.

4 Web Service Discovery Method Based on Semantic Message Bipartite Matching

In this section a method of Web service discovery based on semantic message bipartite matching is introduced. We first outline our approach of adding semantics in WSDL and UDDI. Then we present Basic Definitions for the service discovery. Thereafter, we discuss our semantic discovery algorithm. The method proposes the following phases:

1. Adding Semantic Annotations
2. Basic Definitions
3. Matching Between Web Service and User Query

Each phase within the framework is briefly described in the following subsections. Experiments used to demonstrate this method is in the following section.

4.1 Adding Semantic Annotations

We added semantic annotations to WSDL and UDDI to achieve sufficient expressiveness to automate the discovery process. Our approach involves relating concepts in WSDL to anthologies in Web service description and then providing an interface to UDDI that allows querying based on ontological concepts.

Web Services are described using WSDL descriptions, which provide operational information. Since operation name and message parts are key parts for Web service matching, we only add semantics to operation name, input and output parameters of operations. An omitted WSDL file shown in figure 2 represents a sample Web service and has an operation for getting consultations of a patient. In figure 2, the operation *ByPatientID* is mapped to ontological concepts *ConsultationQuery*, using the attribute operation-concept. In figure 2, the operation input message *ByPatientIDSoapIn* and operation output message *BypatientIDSoapOut* are mapped to ontological concepts *id* and *consultation* respectively. This allows users to search for operations based on ontological concepts.

```
<wsdl:definitions …
  <wsdl:operation name="ByPatientID" SExt:operation-concept="SOnt:ConsultationQuery">
    <wsdl:documentation Use the PatientID with the FetchConsultation method to get a
formatted medical consultation./>
    <wsdl:input message="s0:ByPatientIDSoapIn" SExt:input-concept="SOnt:id"/>
    <wsdl:output message="s0:ByPatientIDSoapOut" SExt:output-concept="SOnt:consultation"/>
  </wsdl:operation>
</wsdl:definitions>
```

Figure 2: Adding Semantic Annotations to WSDL

The approach to store the mapping between operations and ontological concepts in UDDI is similar to the one suggested by [11].During a Web service publication, ontological concepts representing operations and their message parts of the WSDL descriptions of the Web service are stored using UDDI structures, tModels and CategoryBags.

4.2 Basic Definitions

Discovery of Web service is actually a matching between user query and published Web service descriptions. With semantic annotations being added to WSDL and UDDI, we give basic definitions of Web service, operation of service and user query as follows:

Definition 1: Web Service
A Web service *ws* is the 4-tuple: *ws* = {*n, d, q, P*} , where:

1. *n* is the name of the Web service,
2. *d* is the functional description of the Web service,
3. *q* is a set of quality items of the Web service,
4. *P* = {p_1, p_2,…p_m } is a set of operations of Web service.

Definition 2: Operation
An operation *p* is the 4-tuple: *p* = {*n,d,I,O* } , where:

1. *n* is the name of the operation,
2. *d* is the functional description of the operation,
3. *I* = {i_1,i_2,…i_n} is a set of input parameters of the operation,
4. *O* = {o_1,o_2,...o_m} is a set of output parameters of the operation.

Note that $I \cup O$ is a set of input parameters, output parameters and data types.

Since data types used in message parts of Web service are usually very normal such as Integer, String, Date etc, and data type could be usually transformed from one to another as needed, and it is not difficult to tell out the type of given data by the message semantic annotations. In this paper, service matching is based on the name of semantic messages, while not considering the data type.

For $\forall e \in I \cup O$ of the operations is related to an ontological concept, and *n* is related to an ontological concept too.

Definition 3: User Query
A user query *r* is the 4-tuple: *r* = {n,*I*,*O*,λ}, where:

1. *n* is the functional name of user query, it is the desired functional operation,
2. *I* is a set of input parameters of user query,
3. *O* is a set of output parameters of user query,

4. $\lambda(0<\lambda\leq1)$ is a value set by user, when the similarity of a Web service and user query is not less than λ, the Web service is the suitable service. λ could be changed as needed.

Note that for $\forall e \in I^r \cup O^r$ of user query is related to an ontological concept, as well as *n*.

4.3 Matching of Web Service and User Query

Generally, a Web service includes a set of operations. We believe that as long as one of the operations is suited to the user query, the Web service is the suitable service. So problem that matching of Web service and user query could be transformed to matching of operations and user query.

Algorithm: Matching of Operations and User Query
Input: *ws*, *r*
Output: $p_1, p_2, \ldots p_m$

```
1. PSet = φ;//a set of suitable operations
2. for all(p∈P){
3.   if(p.n is not same with r.n in ontological concept ) continue;
4.   if(sim(p,r)≥λ) PSet = PSet∪{p};
5. }
6. return PSet;
```

Notes: The input of the algorithm is a Web service *ws={n,d,q,P}* and a query *r={ n,I,O,λ}*, the output of the algorithm is a set of operations of ws.

Only when *p* and *r* has the same ontological concept in operation name and *sim(p,r)*≥*λ*, then *p* is the suitable operation for the *r.*

And *sim*(*p,r*) represents the integrated similarity of inputs matching and outputs matching between *p* and *r*.

Algorithm: Get Integrated Similarity of Inputs Matching and Outputs Matching between p and r
Input: *p*, *r*
Output: *sim*(*p*,*r*)

```
1. if ($|O^r| > |O|$ or $|I| > |I^r|$) return 0;
2. dose semantic message matching for bipartite graph ($o^r \rightarrow o$) and ensures $\sum Sim(o^r,o)$ reaches its max;
3. dose semantic message matching for bipartite graph ($i \rightarrow i^r$) and ensures $\sum Sim(i,i^r)$ reaches its max;
4. return $a\sum Sim(o^r,o))/|O^r|+b\sum Sim(i,i^r)/|I|$;
```

Notes : the input of the algorithm is an operation *p={n,d,I,O}* and a user query *r={ n,I,O,λ}*, the output of the algorithm is the integrated similarity of message matching between *p* and *r*.

Step 1 show that if the number of outputs of operation $|O|$ is less than the number of outputs of user query $|O^r|$, the result is 0, i.e. the operation is not suitable for the query. The same result returned when the number of inputs of operation $|I|$ is greater than the number of outputs of query $|I^r|$.

Step 2 get the max sum of similarity values in outputs matching $\sum Sim(o^r,o)$.

Step 3 get the max sum of similarity values in inputs matching $\sum Sim(i,i^r)$.

The Step2 and Step3 use the same algorithm with different parameter orders. In Step 2, outputs of user query match outputs of operation, $o^r \rightarrow o$. In step 3, inputs of operation match inputs of user query, $i \rightarrow i^r$.

In step 4, *a* and *b* are coefficient used in outputs matching and inputs matching respectively, *a+b=1*, and they can be changed as needed.

Semantic Message Bipartite Matching

In order to ensure that the Web service could meet all the desired outputs of the user query and user could provide all the inputs that an operation needed. The match between *p* and *r* should be: all the outputs of *r* match with all the outputs of *p,* and all the inputs of *p* match with all the inputs of *r.*

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200007

The Web service matching problem, whether input matching or output matching, could be considered as the problem of optimal matching for bipartite graph, i.e. get an optimal matching for bipartite graph, the set $X = \{x_1, x_2, \ldots x_n\}$ to the set $Y = \{y_1, y_2, \ldots y_m\}$ ($n \le m$), ensuring that $\sum Sim(x,y)$ reaches its max. $Sim(x,y)$ is the similarity of x (an ontological concept related to set X) and y (an ontological concept related to set Y), and $0 \le Sim(x,y) \le 1$.

However the problem of optimal matching for bipartite graph is limited by the condition $|X| < |Y|$, it could not be applied directly when $|X| \le |Y|$. So it is necessary to extend the definition of optimal matching for bipartite graph: a weighted bipartite graph $G = (X,Y,E)$, and M is one of the suitable matches of G, *and* $|X| \le |Y|$, then M is the optimal extended match of G only when: (1) M is the optimal match of G when $|X| < |Y|$, (2) M consists of all the vertices of X and M is the match of G with the sum of weights reaches its max when $|X| < |Y|$. The definition could be proved easily, and the proof is omitted here.

KM(Kuhn-Munkres)[8] is a classic algorithm applied to getting optimal match for bipartite graph. Our algorithm of semantic message bipartite matching is based on KM and is outlined as follows:

Algorithm: Semantic Message Bipartite Matching

Input: $G = (X,Y,E)$; // $|X| \le |Y|$
Output: M; // M is the optimal match of G

```
1.  if($|X|<|Y|$)$G(X,Y,E) \rightarrow G'(X',Y,E')$, $X' = X \cup Z, Z = \{z_1, z_2, \ldots, z_k\}\ (k = |Y| - |X|)$ and $E' = E \cup \{<z_i, y_j> | 1 \le i \le |Z|, 1 \le j \le |Y|,
    W_{z_i y_j} = 0\}$ ; //bipartite graph G to complete bipartite graph G'
2.  w[i][j] ($i \in X', j \in Y$) are assigned with the similarity of vertex i in X' and vertex j in Y;
3.  Lx[i] = max{w[i][j]}($i \in X', j \in Y$); Ly[j]=0;
4.  matched[i] = 0;
5.  for all($i \in X'$){
6.  while(true){
7.  if(i has extending alternating path) break;
8.  delta=min{Lx[i]+Ly[j]-w[i][j], $i \in S$, $j \in$ not T} //S is a set of vertices of X' while in extending path now, and T
    is a set of vertices of Y while in extending path now;
9.  Lx[i]=Lx[i]-delta($i \in S$), and Ly[j]=Ly[j]+delta($j \in T$);
10. }
11. }
12. if($|X|<|Y|$) { $M' \rightarrow M$ by eliminating $E' = \{<z,y> | z \in Z\}$ from M', and M' is the optimal match of the complete
    bipartite graph G';
13. }
14. return M;
```

Notes: the input of the algorithm is the bipartite graph G, the output of the algorithm is M, the optimal matching of G.

In step 2, the approach of similarity computing is similar to the one suggested by [9].

Remarks: Since KM has computation time $O(n^3)$, so the forward searching procedure has computation time $O(m \times 2n^3)$, and m is the number of operations of the Web service, n is the number of parameters of inputs or outputs.

Consequently, the two significant dimensions to determine the performance of the matching algorithm are: (1) the length of sequence operations in the Web service, (2) the inputs and outputs parameter set size.

5 Experiments and Analysis

To validate the efficiency of our algorithm, we conducted experiments while considering Precision, Recall Rate and Scalability of our service matching method. Precision is the rate of suitable services that matching method returned to the total services that being matched. Recall rate is the rate of suitable services that matching method returned to the suitable services in the test set. Scalability is response time affected by the increasing of service number.

We load ontological concepts from WordNet [15] randomly to mapping with user query, operation name and message parts respectively. Groups of Web services with different service number were built and used as data sets for experiments. And it is followed that inputs and outputs are absolutely different when Web services are created with random semantic mappings. As well as the creation of user query, it ensures that inputs and outputs are semantically different.

Test Recall Rate and Precision of our semantic message bipartite matching. We conducted a comparison with the no bipartite matching method. We search for 10 Queries over 100 services, there would be a total of 10 x 100 = 1000 matching, and get the average precision and recall rate. Here we set *λ*=0.9, *λ is* appeared in the definition of user query and it could be changed as needed.

From figure 3 we can see that when the max dispersion of parameters set size between service and query is zero, the recall rate is the same whether or not using bipartite matching method. With the dispersions increased one by one, the recall rate is not affected with bipartite matching and the recall rate dropped rapidly with no bipartite matching. The recall rate with bipartite matching is much higher especially when the dispersion is greater than 1. As far as precision, it improved only a little by using the bipartite matching.

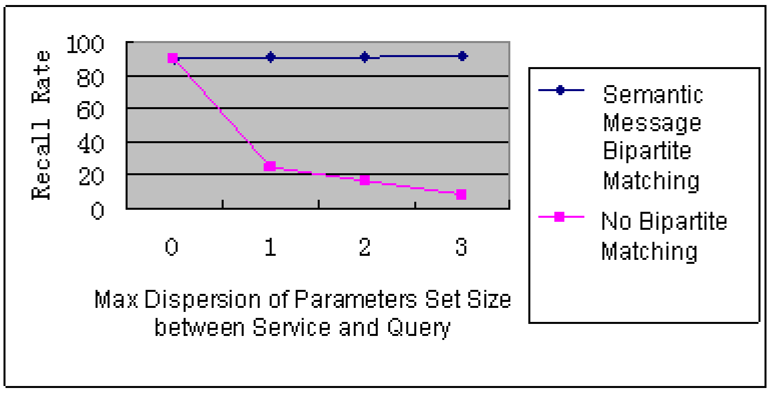

Figure 3: Recall Rate Comparison

Test Scalability of our Semantic message bipartite matching, we load 4 groups of Web services, total number of services are 200,400,600 and 800 plus 10 service queries to test the service matching and calculate the average response time. The experimental results are shown in figure 4. We can see in figure 4 that the response time increased with the number of services grew up, and it is linearity.

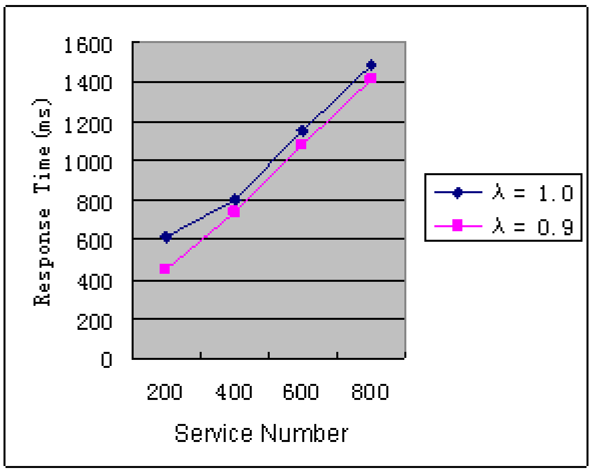

Figure 4: Response Time of Semantic Message Bipartite Matching

6 Conclusions and Future Work

Existing Web service specifications and protocols are limited to the syntactic level, which cannot express semantic information, thus limiting the dynamic discovery ability of Web services. This paper proposes a discovery method with semantic matchmaking which could ease the way to use UDDI registered Web services in remote medical systems. Extending WSDL by adding ontology related semantics which are mapped to necessary operation interfaces of Web services, this method is easy to implement because of using WSDL standard instead of some specific service languages or models. Then a method of service discovery based on bipartite matching of message semantic similarity is proposed. The experiments illustrates that the method improves the efficiency in service discovery, and gets fine scalability. It is conclusive that it contributes to the improvement of service discovery efficiency when service is retrieved in an automatic way.

This paper discusses the discovery method of Web services without considering non functional properties. The next step is to improve the algorithm proposed in this paper by adding more non functional properties, such as QoS. It would be taken into account that using QoS in the service matching method to improve the efficiency of service discovery.

Acknowledgments

This work is partially supported by the Youth Science and Research Foundation of Neusoft Institute of Information, grant No. NEUSOFTIIT2007A0001.

References

[1] S. Agarwal and R. Studer. Automatic Matchmaking of Web Services, IEEE International Conference on Web Services, ICWS, USA, September 2006.
[2] ARTEMIS. (1995, June). [Online]. Available: http://www.srdc.metu.edu.tr/artemis.
[3] Cesar Cáceresc, Alberto Fernández, Sascha Ossowski, Carlos Matteo Vasirani, Agent-Based Semantic Service Discovery for Healthcare: An Organizational Approach, IEEE Intelligent Systems, vol. 21, no. 6, 2006, pp. 11-20.
[4] Nadine Fröhlich, Heikki Helin, Heimo Laamanen, Thorsten Möller, Thomas Schabetsberger, Heiko Schuldt, and Christian Stark, Semantic Service Co-Ordination for Emergency Assistance in Mobile e-Health Environments, Workshop on Semantic Web in Ubiquitous Healthcare, collocated with the 6th International Semantic Web Conference (ISWC2007), 2007.
[5] D. Grigori, J.C. Corrales, and M. Bouzeghoub. Behavioral Matchmaking for Service Retrieval, IEEE International Conference on Web Services, ICWS, USA, September, 2006.
[6] J.H. Hausmann, R. Heckel and M. Lohman. Model-based Discovery of Web Services, IEEE International Conference on Web Services, ICWS, USA, 2004.
[7] M. Klusch, B. Fries, and K. Sycara. Automated Semantic Web Service Discovery with OWLS-MX, International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Japan, 2006.
[8] L. Lovasz, and M. D. Plummer. Matching Theory. Amsterdam: North-Holland, 1986.
[9] RiTa.Wordnet a Wordnet library for Java/Processing. (2008). [Online]. Available: http://www.rednoise.org/rita/wordnet/documentation/index.htm.
[10] Nigel Shadbolt, Paul Lewis, Srinandan Dasmahapatra, David Dupplaw, Bo Hu, and Hugh Lewis, MIAKT: Combining Grid and Web Services for Collaborative Medical Decision Making, AHM2004 UK eScience All Hands Meeting, 2004, pp.784-792.
[11] Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, and John Miller, Adding Semantics to Web Services Standards. 1st International Conference on Web Services (ICWS'03), Las Vegas,NV(US), June 23-26, 2003.
[12] Emanuele Della Valle, and Dario Cerizza. The mediators centric approach to automatic Web Service discovery of Glue. CEURWorkshop Proceedings, vol. 168, 2005, pp. 35–50.
[13] Emanuele Della Valle, Dario Cerizza, Veli Bicer, Yildirak Kabak, Gokce Banu Laleci, and Holger Lausen, The Need for Semantic Web Service in the eHealth, W3C workshop on Frameworks for Semantics in Web Services, 2005.
[14] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A QoS-Aware Selection Model for Semantic Web Services, 4th International Conference on Service Oriented Computing, ICSOC, USA, December, 2006.
[15] WordNet a lexical database for the English language. (2006). [Online]. Available: http://wordnet.princeton.edu/obtain.