Li, Maozhen, Yu, Bin, Sahota, Vijay and Qi, Man (2009) Web services discovery with rough sets. International Journal of Web Services Research, 6 (1). pp. 69-86. ISSN 1545-7362

This version is available at: http://eprints.mdx.ac.uk/4015/

Copyright:

Middlesex University Research Repository makes the University's research available electronically.

Copyright and moral rights to this work are retained by the author and/or other copyright owners unless otherwise stated. The work is supplied on the understanding that any use for commercial gain is strictly forbidden. A copy may be downloaded for personal, non-commercial, research or study without prior permission and without charge.

Works, including theses and research projects, may not be reproduced in any format or medium, or extensive quotations taken from them, or their content changed in any way, without first obtaining permission in writing from the copyright holder(s). They may not be sold or exploited commercially in any format or medium without the prior written permission of the copyright holder(s).

Full bibliographic details must be given when referring to, or quoting from full items including the author's name, the title of the work, publication details where relevant (place, publisher, date), pagination, and for theses or dissertations the awarding institution, the degree type awarded, and the date of the award.

If you believe that any material held in the repository infringes copyright law, please contact the Repository Team at Middlesex University via the following email address:

eprints@mdx.ac.uk

The item will be removed from the repository while any claim is being investigated.

See also repository copyright: re-use policy: http://eprints.mdx.ac.uk/policies.html#copy
Table of Contents

EDITORIAL PREFACE

Web Services Discovery, Composition, and Quality
Liang-Jie Zhang, IBM T.J. Watson Research Center, USA

RESEARCH ARTICLES

WSBen:
A Web Services Discovery and Composition Benchmark Toolkit
Seog-Chan Oh, General Motors R&D Center, USA
Dongwon Lee, The Pennsylvania State University, USA

USDL:
A Service-Semantics Description Language for Automatic Service Discovery and Composition
Srividya Kona, The University of Texas at Dallas, USA
Ajay Bansal, The University of Texas at Dallas, USA
Luke Simon, The University of Texas at Dallas, USA
Ajay Mallya, The University of Texas at Dallas, USA
Gopal Gupta, The University of Texas at Dallas, USA
Thomas D. Hite, Metallect Corp., USA

Using Web Service Enhancements to Establish Trust Relationships with Privacy Protection (Extended and Invited from ICWS 2006 with id 47)
Zhengping Wu, Weaver, University of Virginia, USA
Alfred C. Weaver, University of Virginia, USA

Web Services Discovery with Rough Sets
Maozhen Li, Brunel University, UK
Bin Yu, Level E Limited, UK
Vijay Sahota, Brunel University, UK
Man Qi, Canterbury Christ Church University, UK

A Model-Based Approach for Diagnosing Fault in Web Service Processes
Yuhong Yan, Concordia University, Canada
Philippe Dague, University Paris-Sud 11, France
Yannick Pencolé, LAAS-CNRS, France
Marie-Odile Cordier, IRISA, France
ABSTRACT

Web services are emerging as a major technology for building service-oriented distributed systems. Potentially, various resources on the Internet can be virtualized as Web services for a wider use by their communities. Service discovery becomes an issue of vital importance for Web services applications. This article presents ROSSE, a Rough Sets based Search Engine for Web service discovery. One salient feature of ROSSE lies in its capability to deal with uncertainty of service properties when matching services. A use case is presented to demonstrate the use of ROSSE for discovery of car services. ROSSE is evaluated in terms of its accuracy and efficiency in service discovery.

Keywords: OWL-S; Rough Sets; Service Matchmaking; Web Service Discovery

INTRODUCTION

Web services are emerging as a major technology for developing service-oriented distributed systems. Potentially, many resources on the Internet or the World Wide Web can be virtualized as services for a wider use by their communities. Service discovery becomes an issue of vital importance for Web service applications. As shown in Figure 1, discovered services can either be used by Web service applications or they can be composed into composite services using workflow languages such as BPEL4WS (Andrews Curbera, Dholakia, Goland, Klein, Leymann et al., 2003). UDDI (Universal Description, Discovery and Integration, http://www.uddi.org) has been proposed and used for Web service publication and discovery. However, the search mechanism supported by UDDI is limited to keyword matches. With the development of the Semantic Web (Berners-Lee, Hendlet, & Lassila, 2001), services can be annotated with metadata for enhancement of service discovery. The complexity of this metadata can range from
simple annotations, to the representation of more complex relationships between services based on first order logic.

One key technology to facilitate this semantic annotation of services is OWL-S (Martin, Paolucci, McIlraith, Burstein, McDermott, McGuinness et al., 2004), an OWL (Web Ontology Language, http://www.w3.org/TR/owl-features/Reference) based ontology for encoding properties of Web services. OWL-S ontology defines a service profile for encoding a service description, a service model for specifying the behavior of a service, and a service grounding for invoking the service. Typically, a service discovery process involves a matching between the profile of a service advertisement and the profile of a service request using domain ontologies described in OWL. The service profile not only describes the functional properties of a service such as its inputs, outputs, pre-conditions, and effects (IOPes), but also non-functional features including service name, service category, and aspects related to the quality of a service. In addition to OWL-S, another prominent effort on Semantic Web services is WSMO (Roman, Keller, Lausen, Bruijn, Lara, Stollberg et al., 2005), which is built on four key concepts—ontologies, standard WSDL based Web services, goals, and mediators. WSMO stresses the role of a mediator in order to support interoperation between Web services.

However, one challenging work in service discovery is that service matchmaking should be able to deal with uncertainty in service properties when matching service advertisements with service requests. This is because in a large-scale heterogeneous system, service publishers and requestors may use their pre-defined properties to describe services, for example, in the form of OWL-S or WSMO. For a property explicitly used in one service advertisement, it may not be explicitly used by another service advertisement within the same service category. As can be seen from Table 1, the property $P_1$ used by the service advertisement $S_1$ does not appear in the service advertisement $S_2$. When services $S_1$ and $S_2$ are matched with a query using properties $P_1$, $P_2$ and $P_3$, the property $P_1$ becomes an uncertain property when matching $S_2$. Similarly, the property $P_3$ becomes an uncertain property when matching $S_1$. Consequently, both $S_1$ and $S_2$ may not be discovered because of the existence of uncertainty of properties even though the two services are relevant to the query.

**Figure 1. A layered structure for service-oriented systems**
It is worth noting that properties used in service advertisements may have dependencies, for example, both $P_1$ and $P_3$ may be dependent properties of $P_2$ when describing services $S_1$ and $S_2$ respectively. Both $S_1$ and $S_2$ can be discovered if $P_1$ and $P_3$ (which are uncertain properties in terms of the user query) can be dynamically identified and reduced in the matching process. To increase the accuracy of service discovery, a search engine should be able to deal with uncertainty of properties when matching services.

In this article, we present ROSSE, a Rough Sets (Pawlak, 1982) based Search Engine for Web service discovery. One salient feature of ROSSE lies in its capability to deal with uncertainty in service properties (attributes) when matching service advertisements with service requests. Experiment results show that ROSSE is more effective in service discovery than existing mechanisms such as UDDI keyword matching and OWL-S matchmaking.

The remainder of this article is organized as follows. The ROSSE Design section presents the design details of ROSSE. The ROSSE Case Study section gives a case study to demonstrate the use of ROSSE for discovery of car services. The ROSSE Implementation and Evaluation section evaluates ROSSE from the aspects of accuracy and efficiency in service discovery. The Related Work section discusses some related work, and the Conclusion and Future Work section concludes the article.

### ROSSE Design

ROSSE considers input and output properties individually when matching services. For the simplicity of expression, input and output properties used in a service request are generally referred to as service request properties. The same goes to service advertisements.

Figure 2 shows ROSSE components. The Irrelevant Property Reduction component takes a service request as an input (step 1), and then it accesses a set of advertised domain services (step 2) to remove irrelevant service properties using the domain ontology (step 3). Reduced properties will be marked in the set of advertised domain services (step 4). Once invoked (step 5), the Dependent Property Reduction component accesses the advertised domain services (step 6) to discover and reduce indecisive properties which will be marked in advertised domain services (step 7). Invoked by the Dependent Property Reduction component (step 8), the Service Matching and Ranking component accesses the advertised domain services for service matching and ranking (step 9), and finally it produces a list of matched services (step 10).

In the following sections, we describe in depth the design of ROSSE components for service matchmaking and discovery. Firstly, we introduce Rough sets for service discovery.

### Rough Sets for Service Discovery

Rough sets method is a mathematic tool that can deal with uncertainty in knowledge discovery. It is based on the concept of an upper and a lower approximation of a set as shown in Figure

<table>
<thead>
<tr>
<th>service advertisements</th>
<th>property</th>
<th>property</th>
<th>property</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$P_1$</td>
<td>$P_2$</td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td></td>
<td>$P_2$</td>
<td>$P_3$</td>
</tr>
</tbody>
</table>

Table 1. Two service advertisements with uncertain service properties
3. For a given set \( X \), the yellow grids (lighter shading) represent its upper approximation, and the green grids (darker shading) represent its lower approximation. We introduce Rough sets for service discovery in the following way.

Let

- \( \Omega \) be a domain ontology.
- \( U \) be a set of \( N \) service advertisements, \( U = \{s_1, s_2, ..., s_N\}, N \geq 1 \).
- \( P \) be a set of \( K \) properties used in the \( N \) service advertisements, \( P = \{p_1, p_2, ..., p_K\}, K \geq 2 \).
- \( P_A \) be a set of \( M \) properties used in service advertisements which are relevant to a service request \( R \) within the domain ontology \( \Omega \), \( P_A = \{p_{A1}, p_{A2}, ..., p_{AM}\}, P_A \subseteq P, M \geq 1 \).
- \( X \) be a set of service advertisements relevant to the service request \( R, X \subseteq U \).
- \( \bar{X} \) be the lower approximation of the set \( X \).
- \( U \) be the upper approximation of the set \( X \).

According to the Rough sets theory, we have

\[
\bar{X} = \{x \in U : [x]_{P_A} \cap X \neq \emptyset\} \tag{2}
\]

For a property used by a service request \( p \in P_A \), we have

- \( \forall x \in \overline{X}, x \) definitely has property \( p \).
- \( \forall x \in \overline{X}, x \) possibly has property \( p \).
- \( \forall x \in U - \overline{X}, x \) absolutely does not have property \( p \).

The use of “definitely,” “possibly” and “absolutely” are used to encode properties that cannot be specified in a more exact way. This is a significant addition to existing work, where discovery of services needs to be encoded in a precise way, making it difficult to find services which have an approximate match to a query.

Advertised domain service properties may be irrelevant (having no effect on service matching) or relevant (having an impact on service matching). Certain properties used by advertised services may be redundant which can be reduced without losing essential classificatory information. The concept of the reduct is fundamental for Rough sets theory (Winiarski, 2001). Service property reduction can be considered as a process of finding a smaller (than the original one) set of properties with the same or close classificatory power as

---

Figure 2. ROSSE components

![Figure 2. ROSSE components](image)
the original set. For a service query, the most relevant properties of advertised services can be determined after property reduction.

**Reducing Irrelevant Properties**

When searching for a service, a service request may employ some properties which are irrelevant to the properties used in a service advertisement within one domain ontology. These irrelevant properties used in service advertisements should be removed before the service matchmaking process is performed.

Let

- $p_R$ be a property used in a service request.
- $p_A$ be a property used in a service advertisement.

Following the work proposed in (Paolucci, Kawamura, Payne, & Sycara, 2002), we define the following relationships between $p_R$ and $p_A$:

- **exact** match, $p_R$ and $p_A$ are equivalent or $p_R$ is a subclass of $p_A$.
- **plug-in** match, $p_A$ subsumes $p_R$.
- **subsume** match, $p_R$ subsumes $p_A$.
- **nomatch**, no subsumption between $p_R$ and $p_A$.

For each property used in a service request, the Irrelevant Property Reduction component uses Algorithm 1 to remove irrelevant properties from advertised services. For those properties used in service advertisements that have a nomatch result, they will be treated as irrelevant properties. Service advertisements are organised as service records in a database. Properties are organised in such a way that each property uses one column to ensure the correctness in the following reduction of dependent properties. As a property used in one service advertisement might not be used in another one, some properties may have empty values. For a service request, a property with an empty value in a service record becomes an uncertain property. If a property in an advertised service record is marked as nomatch, the column associated with the property will be marked as nomatch. As a result, all properties within the column including uncertain properties (i.e., properties with empty values) will not be considered in service matchmaking.

**Reducing Dependent Properties**

Properties used by service advertisements may have dependencies. Dependent properties are indecisive properties which have no effect on service matching. Building on the work
Algorithm 1. Reducing irrelevant properties from service advertisements

1: for each property $p_a$ used in service advertisements
2:    for all properties used in a service request
3:      if $p_a$ is nomatch with any $p_R$
4:         then $p_a$ is marked with nomatch;
5:      end if
6:   end for
7: end for

proposed in (Jensen, Shen, & Tuson, 2005), we designed Algorithm 2 to reduce dependent properties from advertised services.

Let

- $\Omega, U, P, P_A$ be defined as in the Rough Sets for Service Discovery section.
- $P_R^D$ be a set of $L_D$ decisive properties for identifying service advertisements relevant to the service request $R$ in terms of $\Omega$.
- $P_A^{IND}$ be a set of $L_{IND}$ indecisive properties for identifying service advertisements relevant to the service request $R$ in terms of $\Omega$.
- $IND()$ be an indiscernibility relation.
- $f$ be a mapping function from a property to a service advertisement.

Then

$$IND(P_A^{IND}) = \{(x, y) \in U : \forall p_R^{IND} \in P_A^{IND}, f(x, p_R^{IND}) = f(y, p_R^{IND})\}$$

(3)

$$P_A^D = P_A^{IND} - P_A$$

(4)

For a service request, the Dependent Property Reduction component uses Algorithm 2 to find the decisive properties in service advertisements. Specifically, service advertisements with the maximum number of nonempty property values are used in the algorithm as targets to find indecisive properties. The targeted services can still be uniquely identified without using these indecisive properties. All possible combinations of individual indecisive properties are checked with an aim to maximally remove indecisive properties which may include uncertain properties whose values are empty. In the mean time, the following service discovery process is speeded up due to the reduction of dependent properties.

Computing Match Degrees

The Service Matching and Ranking component uses the decisive properties to compute the match degrees of advertised services related to a service request.

Let

- $\Omega, U, P, P_A$ be defined as in the Rough Sets for Service Discovery section.
- $P_R$ be a set of $M$ properties used in a service request $R$, $P_R = \{P_{R_1}, P_{R_2}, ..., P_{R_M}\}$, $M \geq 1$.
- $P_A^D$ be a set of $L_D$ decisive properties for identifying service advertisements relevant to the service request $R$ in terms of $\Omega$, $P_A^D = \{p_{A_1}^D, p_{A_2}^D, ..., p_{A_{L_D}}^D\}$, $L_D \geq 1$.
- $m(p_{R_i}, P_{Aj})$ be a match degree between a property $P_{R_i}$ and a property $P_{Aj}$ in terms of $\Omega$, $P_{R_i} \in P_R, 1 \leq i \leq M, P_{Aj} \in P_A^D, 1 \leq j \leq L_D$.
- $v(P_{Aj})$ be a value of the property $P_{Aj}$, $P_{Aj} \in P_A^D, 1 \leq j \leq L_D$.
- $S(R, s)$ be a similarity degree between a service advertisement $s$ and the service request $R$, $s \in U$.
Algorithm 3 shows the rules for calculating a match degree between a property used in a service request and a property used in a service advertisement. A decisive property with an empty value has a match degree of 50% when matching each property used in a service request. A property used in a service advertisement will be given a match degree of 100% if it has an exact match relationship with a property used in a service request. A match degree of 50% will be given if it has a plug-in relationship with a service request property and the relationship is out of five generations. Similarly, a property used in a service advertisement will be given a match degree of 50% if it has a subsume relationship with a service request property and the relationship is out of three generations.

Each decisive property used for identifying service advertisements has a maximum match degree when matching all the properties used in a service request. \( S(R, s) \) can be calculated using formula (5).

\[
S(R, s) = \frac{\sum_{j=1}^{L_u} \sum_{i=1}^{M} \max(m(p_{Rj}, p_{Aj}))}{L_D}
\]  

(5)

Using the formula (5), ROSSE calculates a matching degree for each service advertisement related to a service request. The similarity degrees are used to produce a lower and an upper approximation set of discovered services.
In this section, we present a use case of ROSSE to discover vehicle services. Figure 4 shows the ontologies used in this scenario defining the classifications of vehicles, objects, exhausts, locations, configurations, brands respectively. Two ontologies are used to classify configurations of vehicles represented respectively by e1-e5 and g1-g4. Relevant vehicle services are registered with ROSSE. In the following sections, we describe how services are matched in terms of the following query to search for car services that sell red BMW mini coopers that have an exhaust of 1.0, and are configured with ABS, manufactured in the UK. Price information is also provided by the car services.

Building a Decision Table

A service decision table is used to compute dependent properties among services. As the number of services registered with ROSSE can be tremendous, the decision table is constructed by sampling registered services. For a specific query, ROSSE randomly selects a certain number of services records. A service record is selected as long as one of its properties has a valid relationship with a property used in a service query. The relationship can be exact, plug-in or subsume as defined in algorithm 1 which is described in the Reducing Irrelevant Properties section.

Table 2 shows a segment of the decision table with 13 service records for discovery of car services. As can be seen from Table 2, properties of advertised services that are relevant

ROSSE CASE STUDY

Algorithm 3. The rules for calculating match degrees between properties used in service requests and service advertisements respectively

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>for each property ( p_{Ai} ) ( \in \ P^A_D ), ( v(p_{Ai}) \neq NULL )</td>
</tr>
<tr>
<td>2</td>
<td>for each property ( p_{Ri} ) ( \in \ P^R )</td>
</tr>
<tr>
<td>3</td>
<td>if ( p_{Ai} ) is an exact match with ( p_{Ri} )</td>
</tr>
<tr>
<td>4</td>
<td>then ( m(p_{Ri}, p_{Ai}) = 1 )</td>
</tr>
<tr>
<td>5</td>
<td>else if ( p_{Ai} ) is a plug-in match with ( p_{Ri} )</td>
</tr>
<tr>
<td>6</td>
<td>then if ( p_{Ri} ) is the kth subclass of ( p_{Ai} ) and 2 ( \leq k \leq 5 )</td>
</tr>
<tr>
<td>7</td>
<td>then ( m(p_{Ri}, p_{Ai}) = 1-(k-1) \times 10% )</td>
</tr>
<tr>
<td>8</td>
<td>else if ( p_{Ri} ) is the kth subclass of ( p_{Ai} ) and ( k&gt;5 )</td>
</tr>
<tr>
<td>9</td>
<td>then ( m(p_{Ri}, p_{Ai}) = 0.5 )</td>
</tr>
<tr>
<td>10</td>
<td>end if</td>
</tr>
<tr>
<td>11</td>
<td>else if ( p_{Ai} ) is a subsume match with ( p_{Ri} )</td>
</tr>
<tr>
<td>12</td>
<td>then if ( p_{Ai} ) is the kth subclass of ( p_{Ri} ) and 1 ( \leq k \leq 3 )</td>
</tr>
<tr>
<td>13</td>
<td>then ( m(p_{Ri}, p_{Ai}) = 0.8-(k-1) \times 10% )</td>
</tr>
<tr>
<td>14</td>
<td>else if ( p_{Ai} ) is the kth subclass of ( p_{Ri} ) and ( k&gt;3 )</td>
</tr>
<tr>
<td>15</td>
<td>then ( m(p_{Ri}, p_{Ai}) = 0.5 )</td>
</tr>
<tr>
<td>16</td>
<td>end if</td>
</tr>
<tr>
<td>17</td>
<td>end if</td>
</tr>
<tr>
<td>18</td>
<td>end for</td>
</tr>
<tr>
<td>19</td>
<td>end for</td>
</tr>
<tr>
<td>20</td>
<td>for each property ( p_{Ai} ) ( \in \ P^A_D ), ( v(p_{Ai}) = NULL )</td>
</tr>
<tr>
<td>21</td>
<td>for each property ( p_{Ri} ) ( \in \ P^R )</td>
</tr>
<tr>
<td>22</td>
<td>( m(p_{Ri}, p_{Ai}) = 0.5 )</td>
</tr>
<tr>
<td>23</td>
<td>end for</td>
</tr>
<tr>
<td>24</td>
<td>end for</td>
</tr>
</tbody>
</table>
Figure 4. Ontologies used in the search scenario

Query: Car, mini cooper, ABS, UK, BMW, Exhaust 1.0, Price, Red

to the car service query are \( f_6, g_2, d_3, f_3, c_2, b_2, b_3, d_2, c_1, e_1/g_1, d_1, b_6 \). If a property in a service record is marked with 1, this means that the property is used by the service in its advertisement. For example, the service \( S_1 \) has properties of \( f_6, g_2, d_3, f_3, c_2, d_1, \) and \( b_6 \) in its advertisement. A property marked with 0 in a service record means that the service does not have the corresponding property in its advertisement, for example, properties such as \( b_2, b_2, d_2, c_1, \) and \( e_1/g_1 \) are not used by the service \( S_1 \) for advertisement. However, it should be noted that a property marked with 0 in a service record does not necessarily mean this property is not relevant to the service. Such a property might be an inherent property of the service. ROSSE deals with properties marked with 0 as uncertain properties when matching services.

**Computing Dependent Properties**

Once a service decision table is constructed, the next step is to compute dependent properties. Using the algorithm 2 presented in the Reducing Dependent Properties section, properties
Table 2. A segment of the decision table used for discovery of car services

<table>
<thead>
<tr>
<th>properties</th>
<th>b5</th>
<th>b2</th>
<th>b3</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>e1</th>
<th>g1</th>
<th>d1</th>
<th>b6</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3. Computed dependent properties

<table>
<thead>
<tr>
<th>properties</th>
<th>b5</th>
<th>b2</th>
<th>b3</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>e1</th>
<th>g1</th>
<th>d1</th>
<th>b6</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

g2, d3, f3, and c2 are indecisive properties which are reduced from the decision table in matching services as shown in Table 3. Table 4 shows the segment of the decision table without dependent properties.

Computing Match Degrees

Decisive properties are used for computing the similarities between an advertised service and a service request. For each decisive property used in a service advertisement and a property used in the service query, a maximum matching degree can be computed using ontologies defined in Figure 4. Table 5 shows the matching degrees of the decisive properties used in the exemplified 13 service records. It should be noted that both e1 and g1 refers to the same property Configuration, but they use different ontology definitions as shown in Figure 4. The matching degree of Configuration to the ABS property used in the query is computed in such way that a mean of two matching degrees using the two ontology definitions (i.e., 100% and 90%) is computed which is 95%.

It is worth noting that for an uncertain property which is marked with the number of 0 in a box of Table, a matching degree of 50% is given. Based on the formula (5) presented in the Computing Match Degrees section, the

Table 4. The segment of the decision table without dependent properties

<table>
<thead>
<tr>
<th>properties</th>
<th>b5</th>
<th>b2</th>
<th>b3</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>e1</th>
<th>g1</th>
<th>d1</th>
<th>b6</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5. Computation of matching degrees

<table>
<thead>
<tr>
<th>Match Degrees</th>
<th>0%</th>
<th>100%</th>
<th>90%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>c2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>c3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Copyright © 2009, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
similarity degree between an advertised service and a service query can be computed. In the car service query case, for example, service \( S_1 \) has a similarity degree of 66.25% and service \( S_{13} \) has a similarity degree of 74.375%.

**ROSSE IMPLEMENTATION AND EVALUATION**

ROSSE is implemented with Java on a Pentium III 2.6G with 512M RAM running Red Hat Fedora Linux 3. Figure 5 shows the homepage of ROSSE. It has two registries for service registration, a UDDI registry and an OWL registry. The UDDI registry is used to register services with WSDL interfaces, and the OWL-S registry is used to register services with OWL-S interfaces. The UUID of a WSDL service registered with the UDDI registry is used to uniquely identify semantic annotation records of the registered service. In this way, WSDL services registered with ROSSE can be matched with semantic inferences instead of using keywords only. jUDDI (http://ws.apache.org/juddi) and mySQL (http://www.mysql.com) are used to build the UDDI registry and UDDI4J (http://uddi4j.sourceforge.net/) is used to query the registry.

OWL-S API (http://www.mindswap.org/2004/owl-s/api) is used to parse OWL-S documents to register services with OWL-S interfaces with the OWL-S registry in ROSSE.

ROSSE provides graphical user interfaces to register services. Figure 6 shows a page to register a vehicle service that has a WSDL Interface, and Figure 7 shows the four steps used to semantically annotate the vehicle service. Figure 8 shows the registration of a zip code finding service with an OWL-S interface in ROSSE.

For a service request, ROSSE computes a matching degree for each service advertisement in terms of its functional input and output properties using formula (5). As shown in Figure 5, ROSSE can discover services with WSDL interfaces or OWL-S interfaces. It can also discover the best service from service advertisements which has the highest matching degree related to a service request.

In this section, we evaluate the accuracy and efficiency of ROSSE in service discovery. We compare ROSSE with UDDI and OWL-S respectively. RACER (Haarslev & Möller, 2001) was used by OWL-S to infer the relationships between properties used in service queries and service advertisements. We implemented a light weighted reasoning component in ROSSE to
overcome a high overhead incurred by RACER. The component uses the Protégé OWL API (http://protege.stanford.edu/plugins/owl/api/) to parse OWL documents.

We designed Pizza services for the tests using the Pizza ontology defined by http://www.co-ode.org/ontologies/pizza/pizza_20041007. owl. Figure 9 shows the Pizza ontology structure. The approach adopted here can be applied to other domains—where a specific ontology can be specified. The use of service properties needs to be related to a particular application-specific ontology.
Figure 8. Registering OWL-S services with ROSSE

Figure 9. Pizza ontology structure

ROSSE Accuracy in Service Discovery

Precision and recall are standard measures that have been used in information retrieval for measuring the accuracy of a search method or a search engine (Rijsbergen, 1979). We performed 4 groups of tests to evaluate the precision and recall of ROSSE in service discovery using 10 service records in each group. Each service had 5 properties of which 2 properties were dependent properties. For a service query, each group had 3 relevant services. The 10 services in group 1 did not have uncertain properties, but group 2
had 3 services with uncertain properties, group 3 had 5 services with uncertain properties and group 4 had 7 services with uncertain properties. Properties such as Size, Price, Nuttopping, Vegetariantopping, and Fruittopping were used by the advertised services. Table 6 shows the evaluation results.

In the tests conducted for group 1, both OWL-S and ROSSE have a precision of 100%. This is because all service advertisements in this group do not have uncertain properties (i.e., properties with empty values). UDDI discovered 4 services, but only 2 services were relevant to the service query with a precision of 50%, and a recall of 66.7%. In the tests of the last 3 groups where advertised services have uncertain properties, OWL-S cannot discover any services producing a precision of 0 and a recall of 0. Although UDDI can still discover some services in these tests, the precision of each group is low. For example, in the tests of group 3 and group 4 where the service property certainty rates are 50% and 30% respectively, UDDI cannot discover any relevant services. ROSSE is more effective than both UDDI and OWL-S in dealing with uncertain properties when matching services. For example, ROSSE is still able to produce a precision of 100% in the tests of the last 3 groups albeit with a low recall which is 33.3%.

**Table 6. ROSSE accuracy in service discovery**

<table>
<thead>
<tr>
<th>Service Property Certainty Rate</th>
<th>UDDI</th>
<th>OWL-S</th>
<th>ROSSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
</tr>
<tr>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>70%</td>
<td>33.3%</td>
<td>33.3%</td>
<td>0</td>
</tr>
<tr>
<td>50%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ROSSE Efficiency in Service Discovery

We have registered 10,000 Pizza service records with ROSSE for testing its efficiency in service discovery. Service discovery involves two processes, one is service matchmaking and the other is service accessing (i.e., accessing matched services). We compared the efficiency of ROSSE in matching services with that of UDDI and OWL-S respectively, and the evaluation results are plotted in Figure 10. We also compared their efficiency in accessing matched services, and the results are plotted in Figure 11.

From Figure 10 we can see that UDDI has the least overhead in matching services. This is because UDDI only supports keyword based exact matching. UDDI does not support the inference of the relationships between requested service properties and advertised service properties which is a time consuming process. We also observe that ROSSE has a better performance in service matchmaking than OWL-S when the number of advertised services is less than 5500. This is because ROSSE used a simpler reasoning component than RACER which was used by OWL-S for matching services. However, the overhead of ROSSE in service matchmaking increases when the number of services gets larger. This is due to the overhead caused by a reduction of dependent properties. The major overhead of OWL-S in
matching services is caused by RACER which is sensitive to the number of service properties instead of the number of services.

From Figure 11 we can see that the ROSSE matchmaking algorithm is most efficient in accessing matched services due to its reduction of dependent properties. The OWL-S has a similar performance to UDDI in this process.

**RELATED WORK**

Service matchmaking is becoming an issue of vital importance in service-oriented systems. UDDI has been proposed to support service publication and discovery. However, the search mechanism supported by UDDI is limited to keyword matches and does not support any inference based on the taxonomies referred to by the tModels. Various extensions (Miles, Papay, Dialani, Luck, Decker, Payne et al., 2003; Powles & Krishnaswamy, 2005; ShaikhAli, Rana, AI-Ali, & Walker, 2003) have been proposed to complement UDDI with rich descriptions and powerful match mechanisms in support of service discovery.

Among the extensions, the UDDI-M approach (Miles et al., 2003) is flexible in attaching metadata to various entities associated with a service, but this approach assumes the properties used in service advertisements and in service requests are consistent. Semantic Web service technologies such as OWL-S and WSMO have been proposed to enhance service discovery with semantic annotations. However, the classical OWL-S matching algorithm (Paolucci et al., 2002) cannot deal with uncertainty in service properties when matching service advertisements with service requests. This work has been extended in various ways in applying Semantic Web services for service discovery. For example, Jaeger, Rojec-Goldmann, Mühl, Liebetruh, and Geihs (2005) introduce “contravariance” in matching inputs and outputs between service advertisements and service requests using OWL-S. Li & Horrocks (2004) introduce a “intersection” relationship between a service advertisement and a service request. Majithia, Ali, Rana, and Walker (2004) introduce reputation metrics in matching services. However, these OWL-S based methods still cannot deal with missing (uncertain) properties.

WSMO introduces mediators trying to support distinct ontologies employed by service requests and service advertisements. However, the discovery mechanism (Keller, Lara, Polleres, Toma, Kifer, & Fensel, 2004)
proposed in WSMO requires that properties used by both the goals and services should be consistent.

Compared with the work mentioned above, ROSSE matchmaking can deal with uncertain properties in matching services. It takes all service advertisements belonging to one service category into one search space to dynamically identify and reduce irrelevant and dependent properties which may be uncertain properties related to a service request.

CONCLUSION AND FUTURE WORK

In this article we have presented ROSSE for service discovery. ROSSE is novel in its capability to deal with uncertainty of service properties for high accuracy in service discovery. The preliminary experimental results achieved so far are encouraging. However, the following issues need to be considered for ROSSE enhancement:

- It has been shown that finding a minimal reduct in Rough set is a problem of NP-hard when the number of attributes gets large (Skowron & Rauszer, 1992). Heuristic methods need to be investigated to speed up the process in service property reduction.
- Services registered with ROSSE could be tremendous. Scalability is one the issues that need to be tackled. UDDI Version 3 (http://uddi.org/pubs/uddi_v3.htm) provides larger support for multiple registries, but the specification does not specify how these registries should be structured for enhanced scalability in service registration. Distributed Hash Table (DHT) based Peer-to-Peer (P2P) systems such as Chord (Stoica, Morris, Liben-Nowell, Karger, Kaashoek, Dabek et al., 2003) and Pastry (Rowstron & Druschel, 2001) have shown their efficiency and scalability in content lookup. Scalability in ROSSE can be improved with DHT structured P2P systems.
- Advertised services may be further described in terms of their non-functional properties related to QoS such as reliability and cost. One challenge is how to model such QoS data so that functionally matched services can be evaluated in terms of their QoS properties.
- Currently ROSSE only supports keyword-based queries. It is expected that complex queries to be supported in ROSSE, for
example, queries with a range or fuzzy queries.

REFERENCES


Maozhen Li is a lecturer in School of Engineering and Design at Brunel University, UK. He received the PhD from Institute of Software, Chinese Academy of Sciences in 1997. He has published over 50 research papers in international journals and conferences. His research interests are in the areas of web services, service discovery and composition, semantic web, intelligent systems, grid computing. He co-authored a book on grid computing entitled The Grid: Core Technologies which was published by Wiley in April 2005. He has been serving as a TPC member for a number of conferences in this area, e.g. IEEE CCGrid’05, CCGrid’06, CCGrid’07, CCGrid’08, IEEE SKG ’05, SKG’06, SKG’07, IEEE CSE’08. He is on the editorial boards of Encyclopedia of Grid Computing Technologies and Applications, and the International Journal of Grid and High Performance Computing.

Bin Yu received the PhD from school of engineering and design at Brunel University in April 2007. He is currently a system analyst in Levele Limited in Edinburgh. His research interests are in the areas of service oriented computing, grid computing and applications, service discovery and composition optimization.

Vijay Sahota is currently a PhD student in school of engineering and design at Brunel University, UK. His research interests are in the areas of Web services modeling and monitoring, service discovery, scalable peer-to-peer networks, and grid computing.

Man Qi is a lecturer in Department of Computing at Canterbury Christ Church University, UK. She was a research fellow in Dept. of Computer Science at University of Bath, UK from Jan. 2001 to Oct. 2003. Her research interests are in the areas of computer graphics, multimedia and Web services applications.