

**Research Report for Service Computing**

**RECOMMENDATION TECHNIQUES FOR WEB SERVICE  
DISCOVERY**

Xianrong Zheng

(3265676)

[Xianrong.Zheng@gmail.com](mailto:Xianrong.Zheng@gmail.com)

## **1. Introduction**

Service-Oriented Computing and Web services are becoming more popular, enabling organizations to use the Web as a market for selling their own services and consuming existing services from others. Nevertheless, the more services are available, the more difficult it becomes to find the most appropriate service for a specific application <sup>[1]</sup>. In this report, we will discuss how recommendation techniques can be used to solve the *Web service discovery problem*.

## **2. Improving Web services discovery with recommendation techniques**

### **2.1 Web services discovery problem and its solutions**

Existing approaches to Web service discovery tend to address different information processing styles <sup>[1]</sup>. Some approaches develop extensive service description and publication mechanisms <sup>[2]</sup>; others use syntactic, semantic, and structural reviews of Web service specifications <sup>[3]</sup>.

However, Web services have nonfunctional characteristics such as response time, throughput, availability, and reliability that can be difficult to control. So, users who want to discover new Web services often seek help from their business partners, experts and friends who have relevant experience.

Another solution to this problem is to apply recommendation systems to Web services discovery and selection: service requests are connected with observation data from the service invocations and executions that follow such requests. Data collected during observations will be used to identify relevant services for specific a request.

### **2.2 Implicit culture**

Recommendation systems for Web services discovery are based on the *implicit culture*. It defines a relation between a set and a group of agents such that elements of the set will behave similarly. People might lack prior knowledge about available services when they need a service to perform a specific task. But others who have previously faced similar needs might know suitable services. As a result, such implicit knowledge can be used to make practical recommendations. This approach assumes that it is possible to utilize the implicit knowledge by observing the behavior of the involved parties and then encouraging newcomers to behave similarly to more experienced members. Table 1 shows the core concepts of the implicit culture.

**Table 1 Core concepts of the implicit culture**

Concept	Implication
<i>attribute</i>	<i>represents additional information about objects, actions, or agents</i>
<i>agent</i>	<i>a particular type of object that can perform actions</i>
<i>group</i>	<i>contains several agents</i>
<i>action</i>	<i>characterized by its name, a set of related attributes, and a set of related objects</i>
<i>scene</i>	<i>includes the set of possible actions and the set of objects the agents can operate with</i>
<i>observation</i>	<i>includes the performed action and the scene</i>

The recommendation system models users or service-based applications that submit requests for Web-service operations as *agents*. It stores Web services' names and information about their providers as *attributes of operations*, while it models client requests, service invocations, and corresponding responses as *actions*. A *scene* could be a set of actions corresponding to the invocation of various service operations: *invoke (...; getWeather (service = GlobalWeather); ...)*. An example performed action could be *invoke (Peter; getWeather (service = GlobalWeather); April 5, 2008)*, which states that Peter invoked the operation *getWeather* of the Web service *GlobalWeather* on April 5, 2008.

### **2.3 The Web services discovery system**

The Web services discovery system records the history of Web services requests and helps users to select services suitable for their applications. In general, three steps are needed to create a Web service recommendation system:

- Formalize the application domain
- Define the culture theory
- Generate recommendation

#### **2.3.1 Formalize the application domain**

A request contains a textual description of the goal, the desired operation's name, a description of its I/O parameters, a description of a desired Web service, and so on. The system stores the request as a *submit\_request action*. It also collects the feedback via the optional *provide\_feedback action*, which expresses the agent's level of satisfaction with the result, or via the *invoke action*, which marks a service as suitable for the request. If an agent decides to use a service, the system acquires further information. The *get\_response action* marks a service as available, and

the *raise\_exception* action signals that the service is not available. The application will generate a feedback when it receives the response message. For example, the feedback is positive if the application obtains the correct output.

### 2.3.2 Define the cultural theory

A cultural theory for Web service discovery consists of a rule *if submit\_request(...) then invoke (...)*, where the dots refer to specific objects or attributes. The administrator can also use the cultural theory definition language to specify other requirements as well. For example, the following rule

```
if submit_request(request-X(cost="low"))
    then invoke(operation-Y(service-Z), request-X) ∧
    provide_feedback (operation-Y(service-Z), cost = "low")
```

means that if the agent requests a service with a low cost, the system will recommend services that other clients considered cheap.

### 2.3.3 Generate recommendations

The recommendation process consists of three steps:

- Find the cultural theory rule that matches the current observation
- Find the corresponding cultural action
- Find the set of scenes where the cultural action is likely to be performed

Firstly, when an agent performs the *submit\_request* action, the observation corresponding to the action is passed into the system and it is matched with the antecedent part of the theory. Then, the system uses the observation's information about the request to obtain the corresponding cultural action, and selects a set of similar agents based on similarity of their past actions. Finally, the system selects scenes where the cultural action is most likely to occur and recommends Web services from the scenes to the requester.

### 2.3.4 Similarity calculation

It is important to determine the similarity between observed actions such as *submit\_request*, *invoke*, and so on. The request is represented as a sequence of terms  $q = (t_1, t_2, \dots, t_{|q|})$ , where  $|q|$  is the length of the request,  $t_j \in T$  ( $j = 1, 2, |q|$ ) and  $T$  is a vocabulary of all terms from the collection of requests  $Q = \{q_1, \dots, q_n\}$  that the agent submits to the system, where  $|n|$  is the total number of requests. Two different similarity metrics are used to calculate the similarity between requests: one syntactic and one semantic.

### The Vector Space Model (VSM) with Term Frequency-Inverse Document Frequency (TF-IDF) syntactic metric

The TF-IDF weight of the term  $t_j$  in the request  $q_i$  can be determined by the following formula:

$$w_{i,j} = \frac{n_{i,j}}{|q_i|} * \log\left(\frac{n}{n_j}\right)$$

where for each term  $t_j$ ,  $n_{ij}$  denotes the number of occurrences of  $t_j$  in  $q_i$ , and  $n_j$  denotes the number of the requests that contain  $t_j$  at least once.  $|q_i|$  defines the length of the request  $q_i$ .

The similarity between requests  $q_i$  and  $q_k$  is defined using the cosine coefficient:

$$sim(q_i, q_k) = \cos(w_i, w_k) = \frac{w_i^T w_k}{\sqrt{w_i^T w_i} \sqrt{w_k^T w_k}}$$

Here,  $w_i = (w_{i1}, \dots, w_{i|T|})$ ,  $w_k = (w_{k1}, \dots, w_{k|T|})$ , denote vectors of TF-IDF weights corresponding to the requests  $q_i$  and  $q_k$ , and  $|T|$  is the length of the vocabulary  $T$ .

### The WordNet-based semantic similarity metric

$q_i = (t_{i1}, t_{i2}, \dots, t_{i|q_i|})$  and  $q_k = (t_{k1}, t_{k2}, \dots, t_{k|q_k|})$  are two requests. The lexical similarity between any pair of their terms  $t_{ij}, j \in \{1, \dots, |q_i|\}$  and  $t_{kl}, l \in \{1, \dots, |q_k|\}$  can be calculated by the following formula:

$$sim(t_{ij}, t_{kl}) = 1 - \frac{1}{2}(ic_{wn}(t_{ij}) + ic_{wn}(t_{kl}) - 2(\max_{t \in S(t_{ij}, t_{kl})} ic_{wn}(t)))$$

In this expression,  $S(t_{ij}, t_{kl})$  denotes the set of concepts that subsume  $t_{ij}$  and  $t_{kl}$ , and  $ic$  denotes a WordNet concept's information content value, which is defined as

$$ic_{wn}(t) = 1 - \frac{\log(hypo(t) + 1)}{\log(\max_{wn})}$$

Here,  $hypo$  is the number of *hyponyms*, and  $\max_{wn}$  is the maximum number of existing concepts.

The similarity between requests  $q_i$  and  $q_k$  can be formulated as the *Maximum Weight Bipartite Matching problem* in a complete bipartite graph, where terms from the two requests define two partitions of vertices and the lexical similarity values define weights of edges. The goal is to find a matching—that is, a subset of edges  $e_{jl} = (t_{ij}, t_{kl})$  such that no two edges in  $M$  share a vertex—with the maximum total weight. This weight defines the similarity between requests  $q_i$  and  $q_k$ .

$$sim(q_i, q_k) = \max_M \sum_{j=1}^{|q_i|} \sum_{l=1}^{|q_k|} \sigma_{jl} sim(t_{ij}, t_{kl})$$

Here,  $\sigma_{jl}$  is 1 if  $e_{jl} = (t_{ij}, t_{kl}) \in M$ . Otherwise, it is 0.

### 3. Conclusion

It is a challenge problem to quickly find web services satisfying the given request when there are a large number of services available. In my opinion, different techniques such as search techniques, semantic techniques, data mining techniques, graph techniques and recommendation techniques should be combined to address the web service problem. Users can first use the keyword search to find a particular Web service operation. Then semantic, data mining, recommendation techniques can be used to provide more accurate query operations if search techniques do not work well. Graph techniques can be used to address the problem of finding the set of Web services with certain constraints.

In this report, how to use recommendation techniques to address the Web service discovery problem is presented. It is the contribution of this work.

### References

- [1] A. Birukou, et al. Improving Web service Discovery with Usage Data, IEEE Software, 2007.
- [2] <http://www.w3.org/Submission/OWL-S/>
- [3] <http://www.wsmo.org/2004/d5/d5.1/v0.1/20041112/>
- [4] T. Yu, et al, Efficient Algorithms for Web services Selection with End-to-End QoS Constraints, ACM Transactions on the Web, 2007.
- [5] J. Pathak, et al. A Framework for Semantic Web Services Discovery, WIDM'05, 2005.
- [6] H. Song, et al. Web service Discovery Using General- Purpose Search Engines, IEEE international Conference on Web Services, 2007.
- [7] H. Lu, et al. Semantic Web services Discovery and Ranking, IEEE/WIC/ACM International Conference on Web Intelligence, 2005.
- [8] Z. Zhu, et al. Fast Discovery of Interesting Collections of Web services, IEEE/WIC/ACM International Conference on Web Intelligence, 2006.
- [9] R. Nayak, et al. Data Mining in Web services Discovery and Monitoring, International Journal of Web services Research, 2008.
- [10] S. Oh, et al. BF\*: Web services Discovery and Composition as Graph Search Problem.
- [11] D. Kavvadias, An efficient Algorithm for the Transversal Hypergraph Generation, Journal of Graph Algorithms and Applications, 2005.