# Towards Multimodal Query in Web Service Search

Bin Xu, Sen Luo, Kewu Sun
Department of Computer Science and Technology
Tsinghua University
Beijing, China
{ xubin, luos, kewusun}@keg.cs.tsinghua.edu.cn

***Abstract*—Web service search has been a serious concern for service-oriented software development. The challenge is how to find the right Web services efficiently and effectively for a given development task. There have been many efforts on developing techniques or systems to search services. Though effective in certain scenarios, existing techniques do not form systems for public use; or are based on one query modal--keyword query, which cannot give the matched services accurately. In this paper, we introduce proposed multimodal query search, where users can use keyword and file as query or custom the query. The multimodal query is based on an innovative similarity measure approach, which incorporates both semantic information and structural information of Web services. Our experiments and test cases validate the effectiveness of the approach. Compared with the alternative system Seekda, it is able to obtain much higher search accuracy with keyword query (with a match rate of 2-4 times higher than that of Seekda). The custom search can achieve 100% top-3 match rate, while Seekda fails in most cases using keywords.**

***Keywords-Web service search, multimodal query, similarity, semantic***

## I. INTRODUCTION

Benefiting from the notion of interoperability and reusability, Web service and service-oriented architecture (SOA) has attracted millions of developers to use services for software development. 70% of the most popular Facebook apps leverage one or more Amazon AWS [32] services. One critical challenge of SOA is to search existing online Web services which can fulfill requirements, before developing any new software.

To support easy locating of right web services, Universal Description, Discovery and Integration (UDDI) standard was proposed for registering and searching Web services. However, UDDI has not been widely adopted in the way its designers had hoped. IBM, Microsoft, and SAP announced they closed their public UDDI nodes in 2006. Other than UDDI, extensive researches have been done in web service search; however, most of them fail to form system for public use.

Nowadays, the situation becomes more and more critical, as Web services are flourishing on the Web, but without an effective mechanism to search them. Conventional search engines such as Google and Bing can be used for document search, but do not provide specialized support for Web service search. To the best of our knowledge, Seekda [12] is the most comprehensive search engine for Web Service nowadays. However, Seekda only provides keyword search, which makes its search quality far from satisfactory. For example, assume that a developer wants to search a Web service with the function of sending email. If he types "send email" in Seekda, the first matched Web service is a Short Message Service (SMS). If he inputs "email" in Seekda, the first Web service is for email validation.

Studies have shown that keyword-based queries significantly limit the expressiveness of users and, therefore, degrade the effectiveness of search [33]. In the case of searching Web services, using only keywords to express the function of a Web service is problematic. As a consequence, it may take developers a considerable amount of time and effort to discover the right set of keywords through a trial-and-error process. The keyword search is insufficient for locating right Web services. Meanwhile, there is multimodal query in other search cases. Such as in Google image search, users can upload an image file as query to search images. In this case, it is easy for users to express query by using image file.

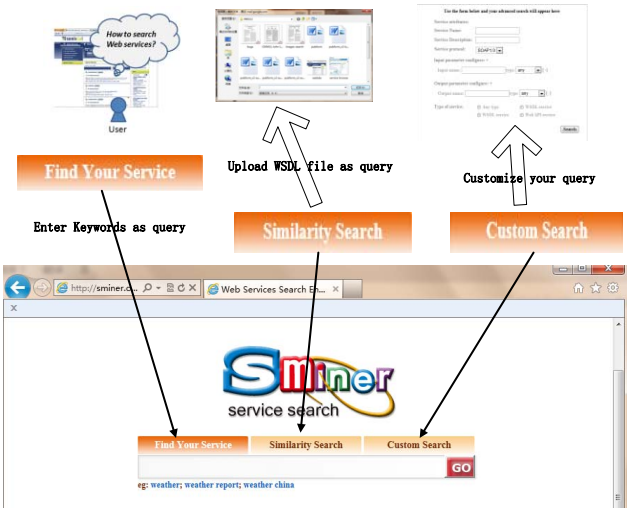


Figure 1. Scenario of Multimodal Query in Web Service Search

In this paper, we present an approach where multimodal query (file and keywords) is advantageous. Figure 1 provides the scenario of how a user can search Web services with multimodal query. Firstly, a user can simply input keywords as query and press the "Go" button. Secondly, if the user finds keywords insufficient to express his query, he can upload a WSDL file or input the URL of a WSDL as

978-0-7695-4752-7/12 $26.00 © 2012 IEEE
DOI 10.1109/ICWS.2012.42

query by pressing the "Similarity Search" button. Thirdly, if the user cannot find an existed WSDL file as query, he can also customize his query. By pressing the "Custom Search" in Figure 1, the user can type in more information such as input and output of operation, which facilitates the expression of service's function in query.

In this paper, we implemented a multimodal-query Web services search system: SMiner (http://sminer.org). The first advantage of multimodal query is its high accuracy. Multimodal query can express the need of service function well, which significantly improves the search accuracy. The second advantage is its ease of use in query. If a user wants to search a service to replace the old one, he can use the old WSDL file as query to get most similar new ones. He needn't type many keywords in the query.

Technically, the major contributions of this paper are:

- An innovative similarity measure approach between Web services is proposed. The approach measures the similarity between Web services both semantically and structurally. Using the approach, SMiner ranks Web services according to similarities.
- SMiner provides multimodal query for Web service search with high accuracy. Multimodal query for Web service includes keyword search, similarity search and custom search. Using these search functions, efficient and accurate Web services discovery is achieved.

The rest of paper is organized as follows. Section 2 presents the overview of the SMiner system. Section 3 introduces the model we used for the similarity measure. Section 4 shows the measure approach between Web services based on the proposed model. Section 5 introduces the adoption of clustering algorithm. In section 6, a set of test cases and experiments are shown to validate the effectiveness of the approach. Section 7 discusses related works. At last section 8 gives conclusion and future works.

## II. SYSTEM OVERVIEW

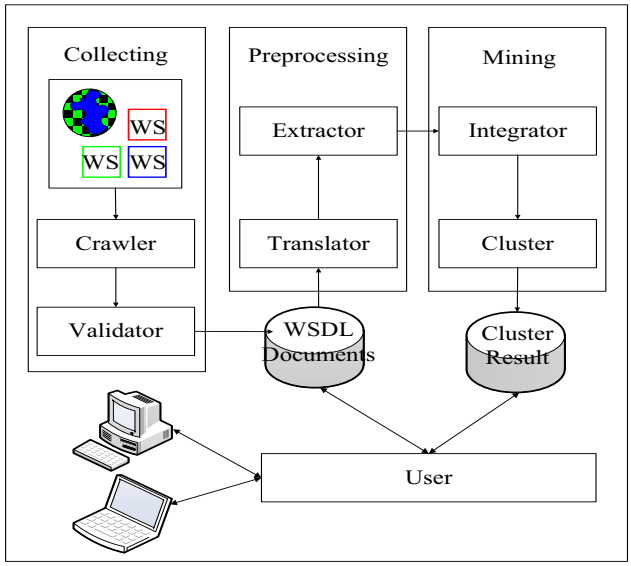


Figure 2. Overview of the Service Searching System

The SMiner system proposed in this paper mainly has three parts, Web services collection, description documents preprocessing and model based mining.

The first part is to collect distributed Web services description from the Internet. A crawler has been implemented for this purpose. Based on our experience, a portion of Web services on the Internet are not usable. A Web services validator is thus needed. We develop a validator to verify the availability of Web services. The validator invokes each crawled Web service to see whether the Web service is available. Unavailable Web services are removed.

When Web services collection is completed, description documents for Web services are stored. However, to use these description documents for similarity measure and clustering, documents preprocessing has to be done first. There is lots of noisy information in the description documents. The information extractor extracts useful information from description documents. As the content of description documents are written in different language, such as Chinese or English. We use Google Translate [13] to unify the language to English.

A model is used to describe the functionalities of services. The effectiveness of similarity measure and cluster lies heavily on whether the model captures the characteristics of services. After careful observation and consideration, we propose a light weighted model for similarity measure of Web services. We will introduce the model and service similarity measure approach in following sections. The integrator is in charge of assembling useful information extracted from description documents into the model. Finally, the cluster divides the Web services on the basis of similarity measure.

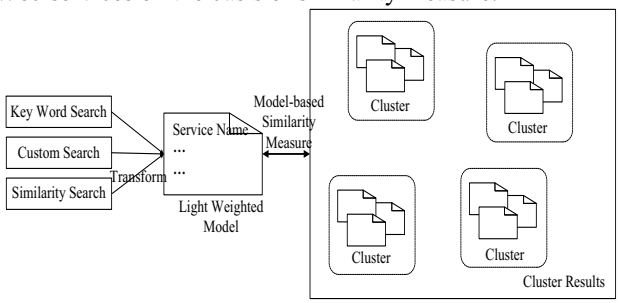


Figure 3. Search Process

As shown in Figure 3, the system provides three search functions for users, including keyword search, similarity search and custom search. When a user submits a search query, the system transforms the query into the lighted weighted model. Then service similarity is calculated on the basis of cluster results. Through similarity measure, the system first finds out the most similar clusters for the query, then get the most similar Web services from the most similar clusters. At last, the system responds the query with the matched Web services.

## III. SMINER MODEL

### A. LIGHT WEIGHTED MODEL

Before measuring the service similarity, it is necessary to define a model to describe the functionalities of each service. Description documents for a Web service has plenty of information about the functionality offered by the Web service. First, it has semantic information which can express the functionality, such as service name and message name. And it also has structural information which can reflect the functionality, such as input/output messages corresponding relationship and structure of types. We propose a light-weighted model (see Figure. 4) which takes in both semantic and structural information.

The model includes elements of service. Generally, a Web service consists of a set of operations. Operation has three parts, Binding, Input Message and Output Message, corresponding to how the operation is called, what parameters it expects and what data structures it returns. Input Message and Output Message are both comprised of several Message Parts. Each element in the model has its own attributes, for example Service has service name and service documentation.

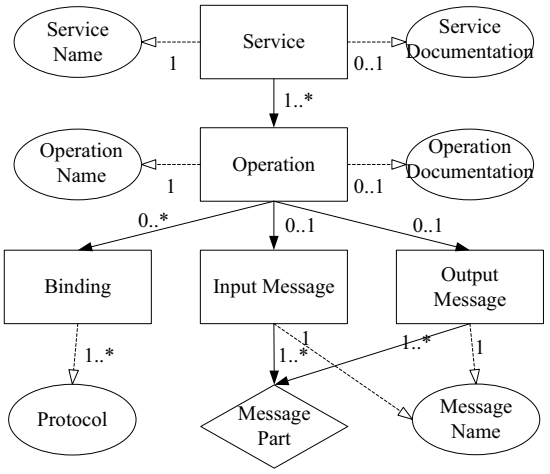


Figure 4. Service Model

In the next subsection, we will show how to extract content from service description to form elements of the model.

### B. ELEMENT EXTRACTION

The extraction has three steps as shown in Figure 5.

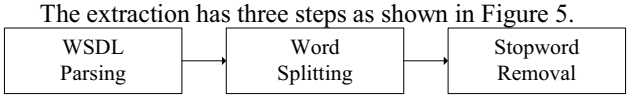


Figure 5. Extraction Process

**Document Parsing**: Description document for Web services is usually XML based. We use DOM [16] to manipulate description document, thus we can extract useful information and do not destroy structural information. In addition, parsing in this way also accomplishes tag removal as DOM does not extract tags out.

**Word Splitting**: Different from text processing, which splitting words by space, we also consider the programming naming specification. Service names, message names, etc. may be composed of several meaningful words, such as QueryService for service name and GetPortalSearchInfoSoapIn for message name. The words are splitted by capital letter and special character (such as "_").

**Stopwords Removal**: Stopwords are words that appear so frequently that they lose their usefulness for search. To remove the stopwords, this paper adopts LUCENE stopwords plus some special stopwords for services. For example, "out" may frequently appears in output message names while has no semantic meaning.

TABLE I. WEB SERVICES STOPWORDS LIST EXAMPLES

| | |
|---|---|
| Web Service stopwords | "out", "soap", "soap12", "http", "get", "post", "request", "response" |

After extraction we get elements in the model. Then we infer relations among these elements through word matching. Using relations among the elements to integrate them, the model is generated from Web services description.

Following code gives out a Web services description example, the operation GetLastTradePrice has input message GetLastTradePriceInput and output message GetLastTradePriceOutput. GetLastTradePriceInput and GetLastTradePriceOutput both have message part TradePrice. And through binding tag, we know that the operation can be called using SOAP.

```
<types>
…
        <element name="TradePriceRequest">
…
        </element>
        <element name="TradePrice">
…
        </element>
…
</types>
…
<message name="GetLastTradePriceOutput">
      <part name="body" element="xsd1:TradePrice"/>
</message>
<message name="GetLastTradePriceOutput">
      <part name="body" element="xsd1:TradePrice"/>
 </message>
…
<portType name="StockQuotePortType">
      <operation name="GetLastTradePrice">
        <input message="tns:GetLastTradePriceInput"/>
        <output message="tns:GetLastTradePriceOutput"/>
      </operation>
 </portType>
…
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
       <soap:binding                                    style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
…
</binding>
```

## IV. SIMILARITY MEASURE APPROACH

### A. OPERATION ORIENTED MEASURE

On the basis of the model, we use an operation oriented approach to measure the similarity between Web services. Suppose a Web service $A$ has n operations, $P_{A1},..,P_{An}$, the

operation similarity degree of *A* to *B* (another Web service, which has m operations $P_{B1},..,P_{Bm}$) is defined in formula (1).

$$SOP(A,B)=\frac{\sum_{i=1}^{n} Max(Sim_{OP}(P_{Ai},P_{Bj})(j=1,..,m)))}{n} \quad (1)$$

Each operation in *A* can find a most similar operation in *B*, namely with which the operation has the maximal similarity degree among operations in *B*. The operation similarity degree of *A* to *B* is equal to the average of these maximal similarity degrees. An example is given out in Table Ⅱ. $P_1$ and $P_2$ are operations of *A*. $P_3$, $P_4$ and $P_5$ are operations of *B*. The operation similarity of *A* to *B* is 0.8.

TABLE II. OPERATION SIMILARITY MEASURE EXAMPLE

| | $P_3$ | $P_4$ | $P_5$ |
|---|---|---|---|
| $P_1$ | 0.1 | 0.5 | 0.9 |
| $P_2$ | 0.7 | 0.3 | 0.1 |

We notice that the operation similarity degree of A to B may be different from the operation similarity degree of B to A. In the example above, the operation similarity degree of B to A is 0.7.

The overall similarity degree of A to B is weighted sum of operation similarity degree and semantic description (service name and service documentation) similarity degree, as shown in formula (2).

$$Sim_S(A,B)=\alpha_1 SOP(A,B)+\alpha_2 SSD(A,B) \quad (2)$$

In the next subsections we will introduce semantic description similarity measure and operation similarity measure.

### *B. SEMANTIC SIMILARITY MEASURE*

Service name, service documentation etc are vectors of words and have useful semantic information. We adopt Normalized Google Distance (NGD) [17] to measure the similarity between word vectors.

Specifically, *NGD* between two words *x* and *y* is defined in formula (3).

$$NGD(x,y)=\frac{\max\{\log f(x),\log f(y)\}-\log f(x,y)}{\log M-\min\{\log f(x),\log f(y)\}} \quad (3)$$

In formula (3), *M* is the total number of web pages searched by Google; *f(x)* and *f(y)* are the number of hits for search words *x* and *y*, respectively; and *f(x, y)* is the number of web pages on which both *x* and *y* occur.

For word vector $V_1$ and $V_2$, the similarity degree between $V_1$ and $V_2$ is defined as following.

$$Sim_{WV}(V_1,V_2)=\frac{\sum_{x\in V_1}\sum_{y\in V_2}(1-NGD(x,y))}{|V_1||V_2|} \quad (4)$$

Semantic description contains service name and service documentation in the model, *SSD* is the average of service name vector similarity degree and service documentation vector similarity degree. Formula (5) gives out definition of *SSD*, continues with notation in former formulas.

$$SSD(A,B)=\beta_1 Sim_{WV}(V_{ASname},V_{BSname})+\beta_2 Sim_{WV}(V_{ASdoc},V_{BSdoc}) \quad (5)$$

### *C. OPERATION SIMILARITY MEASURE*

The similarity degree between operations is composed of five part, operation name similarity degree, operation documentation similarity degree, input message similarity degree, output message similarity degree and binding similarity degree. The definition is given in formula (6).

$$Sim_{OP}(P_1,P_2)=WX^T \quad (6)$$

Where both *X* and *W* are vectors. *X* is the vector of similarity of each part of service $P_1$ and $P_2$, as shown in formula (7). *W* is the weight vector of similarity (Cf. Eq. 8).

$$X=(Sim_{WV}(V_{p_1Name},V_{p_2Name}),Sim_{WV}(V_{p_1Doc},V_{p_2Doc}),\\ Sim_{IN}(In_{p_1},In_{p_2}),Sim_{OUT}(Out_{p_1},Out_{p_2}),Sim_{BD}(B_{p_1},B_{p_2})) \quad (7)$$

$$W=(\varepsilon_1,\varepsilon_2,\varepsilon_3,\varepsilon_4,\varepsilon_5) \quad (8)$$

Operation name similarity degree and operation documentation similarity degree can be calculated using formula (4).

Input messages and output messages are very important parts of operation and they have plenty of useful information, both semantic and structural. Input message and output message similarity degree are measured in the same way, which considers both semantic and structural information, as shown in formula (9) and (10). The similarity degree between input messages is the sum of word vector similarity degree for their names and structural similarity degree.

$$Sim_{IN}(In_{p_1},In_{p_2})=\delta_1 Sim_{WV}(V_{p_1IMsgN},V_{p_2IMsgN})+\delta_2 Sim_{ST}(S_{p_1IMsg},S_{p_2IIMsg}) \quad (9)$$

Similar to formula (9), the similarity between output messages is (10).

$$Sim_{OUT}(Out_{p_1},Out_{p_2})=\eta_1 Sim_{WV}(V_{p_1OMsgN},V_{p_2OMsgN})+\eta_1 Sim_{ST}(S_{p_1OMsg},S_{p_2OIMsg}) \quad (10)$$

The messages (both input messages and output messages) consist of one or more logical message parts. The logical parts are usually defined using XSD. Message parts are elements, and they can be as simple as a simple type or as complex as an array of other elements. Each element has a name attribute and a type attribute. They are both a good candidates for similarity measure.

XSD has defined various simple types; elements are composed from these simple types. For every message in Web services, we extracted all simple types to form a type vector T. The message structural similarity degree is calculated on the type vectors extracted from messages as shown in formula (11).

$$Sim_{ST}(S_{Msg_1},S_{Msg_2})=\frac{Match(T_{Msg_1},T_{Msg_2})}{avg(|T_{Msg_1}|,|T_{Msg_2}|)} \quad (11)$$

The structural similarity degree between messages is type match count divided by average type number. For instance, if type vector for message 1 is {string, float, boolean} and type vector for message 2 is {string, float, float}, the structural similarity degree between messages is 2/3=0.66. And if type vector for message 2 changes to {string, string, string}, the structural similarity degree is 1/3=0.33.

This structural similarity measure between messages makes sense, as there are many seemed different elements or messages are actually have the same structure due to developers' various programming style. For example, following two TradePrice elements have the same structure.

```
<element name="TradePrice">
    <complexType>
```

```
        <all>
          <element name="parameter" type="string"/>
        </all>
      </complexType>
 </element>
<element name="TradePrice" type="string">
```

Moreover in another example, Message1 and Message2 in the following code also have the same structure.

```
<complexType name="Item">
        <all>
          <element name="quantity" type="int"/>
           <element name="product" type="string"/>
        </all>
</complexType>
<message name=”Message1”>
       <part name=”part1” element=”tns:Item”>
</message>
…
<element name="Quantity" type="int">
<element name="Product" type="string">
<message name=”Message2”>
       <part name=”part1” element=”tns:Quantity”>
       <part name=”part2” element=”tns:Product”>
</message>
```

There are multiple protocols and message formats for Web services binding, including SOAP, HTTP GET, HTTP POST and MIME. As shown in formula (12), the similarity degree between bindings equals to protocol match count divided by average protocol number.

$$Sim_{BD}(B_{p_1}, B_{p_2}) = \frac{Match(B_{p_1}, B_{p_2})}{avg(|B_{p_1}|, |B_{p_2}|)} \tag{12}$$

For example, if $P_1$ can be invoked using SOAP and HTTP GET, $P_2$ can be called using SOAP. Binding similarity degree of $P_1$ and $P_2$ is 2/3. If $P_2$ can be called using both SOAP and HTTP POST, Binding similarity degree of $P_1$ and $P_2$ is 0.5.

Operation similarity measure takes into account semantic description (operation name and operation document), how the operation is called (binding), what parameters the operation expects (input message) and what data structures the operation returns (output message), to measure similarity between operations in a comprehensive way (both semantic information and structural information is considered). Similarity measure between Web services makes use of operation similarity measure plus service level semantic similarity measure, thus making similarity measure between Web services more accurate.

As we view every part of the information as equally important, we set the parameter as following: $\alpha_1 = 0.8, \alpha_2 = 0.2, \varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = 0.2, \beta_1 = 0.5,$ $\beta_2 = 0.5, \delta_1 = \delta_2 = 0.5, \eta_1 = \eta_2 = 0.5$. Effectiveness of the setting is validated by test cases and experiments, all the search functions have good accuracy.

In the next section we will introduce the clustering algorithm based on similarity measure.

## V. CLUSTRING

As there are tons of thousand Web services, if we calculate the similarity degree for every pair of Web services, it will be computationally intolerable. Thus we use K-MEANS [18] clustering algorithm to cluster similar Web services based on the similarity measure discussed above, to enhance the efficiency.

In general, K-MEANS aims to partition n objects into k clusters to minimize the within-cluster sum of distance. K-MEANS usually is composed of the following four steps: (1) Place k objects as center objects; (2) Assign each object to the set that has the closest center objects; (3) When all objects have been assigned, recalculate the k center objects; (4) Repeat step (2) and (3) until the center objects no longer move.

The clustering algorithm is carried out as following.

To determine the number of clusters k and the initial center objects, we analyze thousands of Web services on the Internet and identify more than two hundred categories. We relax the category number to three hundred, namely we cluster Web services into 300 sets. We select 300 Web services, which involve considerate manual effort (we endeavor to select Web services from different categories to make the chosen Web services as not similar as possible), as center Web services for 300 sets.

Then assign each Web service into the set to which it is the closest (namely most similar to the center Web service in the set).

Formula (13) is used to find the new center Web services for sets. Center Web service in a set has minimal sum of other Web services' similarity degrees to it.

$$Mean_{Set_i} = S_{j^*} : \sum_{S_o \in Set_i} Sim(S_{j^*}, S_o) \le \sum_{S_o \in Set_i} Sim(S_j, S_o) \quad for \quad all \quad S_j \in Set_i \tag{13}$$

The clustering continues assigning Web services and updating center Web services, until the center Web services stay stable.

## VI. TEST CASE AND EXPERIMENT

Multimodal search, including keyword search, similarity search and custom search, is provided on the basis of the similarity measure algorithm. In this section we validate the effectiveness of these functions through a set of test cases and experiments.

### A. Keyword Search

For keyword search, to find Web services, user just needs to type a word or a phrase. The typed word or phrase is put into the model presented above (in this case, Service Name and Service Documentation are set to be the typed word or phrase, the other parts of the model are empty). Then through the proposed similarity measure proposed, we can get Web services that are most similar to the typed word or phrase.

To validate the keyword search, we conduct comparative experiments with Seekda. For example, we use “send email” as the example keywords. The goal is to find out Web services for email sending. Table III gives out top 20 available Web services of this searching in Seekda.

TABLE III. WEB SERVICES RANKINGS FOR "SEND EMAIL"

| Service Name | WSDL URL | Seekda Ranking | SMiner Ranking | Standard Ranking |
|---|---|---|---|---|
| SendSMSWorld | http://www.webservicex.com/sendsmsworld.asmx?WSDL | 1 | 16 | 17 |
| SendSMS | http://www.webservicex.net/SendSMS.asmx?wsdl | 2 | 18 | 16 |
| Communication | http://www.zeta-uploader.com/webservices/communication?WSDL | 3 | 20 | 11 |
| SendSMS | http://www.webservicex.com/SendSMS.asmx?WSDL | 4 | 17 | 15 |
| fax | http://www.webservicex.com/fax.asmx?wsdl | 5 | 14 | 13 |
| SendEmailService | http://www.abysal.com/soap/AbysalEmail.wsdl | 6 | 2 | 5 |
| fax | http://www.webservicex.net/fax.asmx?wsdl | 7 | 15 | 14 |
| SendMail | http://www.zim.co.il/SendMail.asmx?WSDL | 8 | 8 | 6 |
| webservice | http://smsbug.com/api/webservice.asmx?wsdl | 9 | 19 | 20 |
| Email_x0020_a_x0020_Friend | http://www.discovertheponds.com/ThePonds/ws/EmailAFriend.asmx?WSDL | 10 | 5 | 8 |
| SendEmail | http://www.gottunnel.com/SVOEmailWS/SendEmail.asmx?WSDL | 11 | 1 | 4 |
| WebServices | http://ser02.2sms.com/WebServices/1.0/SMSService.asmx?WSDL | 12 | 13 | 18 |
| EmailService | http://madrid.vitalab.tuwien...2/services/EmailService?wsdl | 13 | 3 | 7 |
| SendMail | http://www.armymkt.com.br/WebServices/SendMail.asmx?WSDL | 14 | 9 | 10 |
| FaxService | http://webservices.venali.net/fax/1.0/faxservice.asmx?WSDL | 15 | 10 | 19 |
| UMSEmailService | https://secure.ums.no/soap/email/1.2/?wsdl | 16 | 11 | 1 |
| timeDateEmail | http://www.vbtrain.net/samplewebprojects/Web%20Interaction%20Sample/timeDateEmail.asmx?wsdl | 17 | 4 | 9 |
| Tiscali_x0020_Email_x0020_Services | http://webservices.tiscali.com/EmailServices.asmx?wsdl | 18 | 12 | 2 |
| EmailSenderService | http://www.mysweetpeace.co.uk/web_services/emailer_ws.wsdl | 19 | 7 | 12 |
| SendToAFriend | http://www.allenandunwin.com/Messaging/SendToAFriend.asmx?WSDL | 20 | 6 | 3 |

Then we invite 10 graduate students (4 PhD candidates and 6 masters) to rank the 20 Web services according to the relevance to the keywords through manual reading and understanding. The most well accepted ranking is also shown in Table Ⅲ. And we use this ranking as standard ranking (the column of Standard Ranking). At last we rank the 20 Web services using keyword search proposed in the paper. Also, the ranking result of the system is provided in Table Ⅲ (the column of SMiner Ranking). To quantitatively measure the result, we use formula (14) to calculate the difference between standard ranking and Seekda Ranking(or SMiner Ranking). *S* represents a Web service. The Set corresponds to the 20 Web services. $Ranking_s$ denotes Web service *S*'s ranking in Seekda or SMiner. $StandardRanking_s$ means Web service *S*'s ranking in standard ranking.

$$D = \sum_{S \in Set} (Ranking_s - \text{Standard}\,Raking_s) \qquad (14)$$

A smaller *D* means smaller difference with standard ranking, which also means better ranking result. Based on Table III, $D_{Seekda}$ is 170 and $D_{SMiner}$ is 80, which means SMiner Ranking is more closer to the standard ranking compared with Seekda Ranking. And the maximal $D_{maximal}$ is the reversed ranking of standard ranking, which is 200. So the match rate of Seekda is (1- $D_{Seekda}$/ $D_{maximal}$)=15%, and the match rate of our system is (1-$D_{SMiner}$/ $D_{maximal}$)=60%. Our system improves match rate by 3 times.

We also carry out other comparative experiments with different keywords in the similar manner; we find that SMiner is able to obtain a much higher search quality (with a match rate of 2-4 times higher than that of Seekda).

### B. Custom Search

Keyword search is still insufficient to find the most accurate Web services (as shown in Table Ⅲ, the top-3 accurate Web services is ranked 11, 12 and 6). This because keyword search sometimes cannot express users' need. It is almost impossible to get the most accurate Web services using such little information.

To overcome this, custom search is also provided by our system. Custom search allows user to type in more information about the Web services he wants, not only keywords description about the Web services.

```
Service Name:"Send Email"
Service Description:"Send email to others"
Inputs parameters:
        name="from" type="s:string"
        name="to" type="s:string"
        name="subject" type="s:string"
        name="body" type="s:string"
        name="attachment" type="s:base64Binary"
        name="from" type="s:string"
        name="cc" type="s:string"
        name="bcc" type="s:string"
Output parameters:
        name="result" type="s:string"
```

Custom search allows user to type in service description, input message names, input message types, output message names and output message types about the Web services he wants. In the same scenario, if we set the information as shown in the above codes, the first Web service found is UMSEmailService, the second Web service is timeDateEmail and the third Web service is SendToAFriend. Namely we find the most accurate Web services through custom search.

Custom search achieve 100% top-3 match precision for all of the experiments (we carry out 5 experiments), while Seekda has 0% top-3match precision.

### C. Similarity Search

In addition to keyword search and custom search, similarity search is also offered by the system. Using similarity search, user needs to input a Web service (URL of the Web service WSDL or a WSDL file), and similarity search returns Web services that are similar to the given

Web service. Then user could select from these similar Web services to get the most appropriate Web services.

The similarity search is implemented through similarity measure and clustering discussed in the above sections. It returns the cluster the given Web service belongs to.

We conduct experiments to shown that the result of clustering is accurate; namely the similarity search is accurate.

We perform a manual classification of the WSDL documents to serve as a comparison point for the clustering algorithm. We distinguish the following clusters: "Excel Service", "Imaging", and "Email". "Excel Service" includes Web services through which we can manipulate an excel spreadsheet remotely, Web services in "Imaging" provide picture management and "Email" is consist of email related Web services .Due to page limit we do not list these clusters.

We use two commonly used criteria in information retrieval to evaluate the effectiveness of our clustering, namely precision and recall. Precision can be seen as a measure of exactness or fidelity, and recall is a measure of completeness.

$$C = \{S_1,..,S_n\}, C' = \{S_1',..,S_m'\} \quad (15)$$

As shown in formula (15), *S* and *S'* both denote Web services. *C* and *C'* are both Web service clusters. *C* is the returned cluster and *C'* is the manual identification cluster (standard cluster).

$$\Pr ecision = \frac{|C \cap C'|}{|C|}, \text{Re } call = \frac{|C \cap C'|}{|C'|} \quad (16)$$

Formula (16) gives out definitions for precision and recall. Precision is the number of correct Web services in *C* divided by the number of Web services in *C*, recall is the number of correct Web services in *C* divided by the number of Web services in *C'*.

The results of clustering are given out in Table Ⅳ.

TABLE IV. CLUSTERING RESULTS

| Cluster | Precision | Recall |
|---|---|---|
| Excel Service | 100% | 91.6% |
| Imaging | 95% | 87.5% |
| Email | 96.4 | 84.4 |

We could not calculate the precision and recall for all the clusters got from clustering, as we could not manually identify all the standard clusters. From the results shown in Table Ⅳ, we can see that our clustering has very high precision and recall, which mean the clusters got from clustering is very close to the standard clusters. Thus we can get that the similarity search discussed in this paper is accurate.

Through these test cases and experiments we can see that all the three search functions could accurately find Web services the user wants. This validates the effectiveness of the system.

## VII. RELATED WORK

**UDDI:** UDDI is part of core Web services standard and current is industry standard for Web services discovery. It is low accuracy as it focuses mainly on keywords matching and is inefficient as there are multiple UDDI registries (user has to search in one UDDI after another).

**Semantic Web Services Discovery:** To overcome the limitation of keywords matching, several semantic descriptions are proposed. These descriptions, which include WSDL-S, OWL-S and WSMO, use ontologies to enhance the service description. A lot of discovery approaches are proposed on the basis of these descriptions. Paolucci [1] regards service match as a logic inference task. Cardoso [2] and Skoutas [3] propose to use the similarity between requested and offered inputs and outputs, which is calculated by comparing classes in associated domain ontology, for service discovery. Further, Bellur [4] calculates similarity between requested and offered inputs and outputs using bipartite graphs. Klusch [5] and Kaufer [6] propose to discovery service in a hybrid way, complementing logic based reasoning with matching based on syntactic similarity computations. In the paper of Skoutas [7], approach to discovery service using multi-criteria dominance relationships, which include keywords, semantic, parameters, etc, is proposed.

However, plenty of Web services do not possess these semantic descriptions, which make aforementioned discovery approaches not practical in reality.

**Web services similarity measure and clustering:** Using basic WSDL, several similarity measure and clustering methods are discussed to facilitate Web services discovery. Dong [8] proposes to calculate inputs/outputs similarity of Web services, then to cluster Web services for service discovery. Wu [9] proposes a service property model, which divides properties of Web services into four categories, and calculating the similarity upon the model. Liu [10] extracts useful information from WSDL, which includes Web services name, Web services abstract and etc. On the basis of the extracted information, text mining methods are used to cluster Web services. Elgazzar [11] improves Liu's work by introducing weight for extracted information and applying the approach to existing Web services.

However, these approaches do not take structural information of Web services into account. As WSDL is a well defined XML schema, description for Web services in WSDL is well structured. The ignorance of structural information will lead to information lost inevitably. Moreover, just like the approaches in semantic Web service discovery, these approaches are not implemented into practical systems for public use.

**Web Services Search Engine:** Although prevalent search engines, for example Google and Bing provide some specific searches, such as Image search and music search, they do not provide specific Web services engine. It is very hard to use them for service discovery.

Web service registries, such as webservicex and xmethods, focus on listing available Web services publicly. Webservicex doesn't provide search function. Xmethods only provides a poor keyword search, which only returns few Web services for query. Using "send email" as keyword it will return an empty list. Seekda [12] is currently the most comprehensive global search engine for Web services. However, Seekda only offers keyword search which leads to low accuracy. Because keyword search could not capture the users' search need well.

## VIII. CONCLUSION AND FUTURE WORKS

In this paper, we study the problem of Web services search. We investigate the available Web services on the Web, and find that using keywords as query is not adequate for service search. Then we propose a comprehensive approach, which incorporates both semantic information and structural information of Web services. Besides keyword search, the system implemented similarity search and custom search. We evaluate the effectiveness of the approach on more than 10000 services. Compared to Seekda with the same searching tasks, SMiner is able to obtain a match rate 2-4 times higher than Seekda. And custom search can achieve 100% top-3 match rate in these searching tasks.

The problem of Web services search is a basic and critical problem of service computing. There are many potential future directions of this work, such as composite service search when a single service cannot satisfy the query and integrating the approach into IDE to facilitate SOA developing in design-time and run-time.

## IX. ACKNOWLEDGEMENTS

This work is supported by China National Science Foundation (No.61170212, and No. 61035004)

## REFERENCES

[1] M. Paolucci, T. Kawamura, T.R. Payne, and K.P. Sycara, "Semantic Matching of Web Services Capabilities," Proc. First Int'l Semantic Web Conf. (ISWC), pp. 333-347, 2002.

[2] J. Cardoso, "Discovering Semantic Web Services with and without a Common Ontology Commitment," Proc. IEEE Services Computing Workshops (SCW), pp. 183-190, 2006.

[3] D. Skoutas, A. Simitsis, and T. Sellis, "A Ranking Mechanism for Semantic Web Service Discovery," Proc. IEEE Services Computing Workshops (SCW), pp. 41-48, 2007.

[4] U. Bellur and R. Kulkarni, "Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching," Proc. IEEE Int'l Conf. Web Services (ICWS), pp. 86-93, 2007.

[5] M. Klusch, B. Fries, and K.P. Sycara, "Automated Semantic Web Service Discovery with OWLS-MX," Proc. Fifth Int'l Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS), pp. 915-922, 2006.

[6] F. Kaufer and M. Klusch, "WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker," Proc. European Conf. Web Services (ECOWS), pp. 161-170, 2006.

[7] Dimitrios Skoutas, Dimitris Sacharidis, Alkis Simitsis, and Timos Sellis. "Ranking and Clustering Web Services Using Multicriteria Dominance Relationships". IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 3, NO. 3, JULY-SEPTEMBER 2010.

[8] Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes, Jun Zhang, Similarity Search for Web Services. In Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.

[9] J. Wu, Z. Wu, Similarity-based web service matchmaking. Proceedings of the IEEE International Conference on Services Computing. Volume 1. 287–294, 2005.

[10] Wei Liu, Wilson Wong, Web service clustering using text mining techniques, International Journal of Agent-Oriented Software Engineering, Vol. 3, No. 1, pp. 6-26, 2009.

[11] Khalid Elgazzar, Ahmed E. Hassan and Patrick Martin, Clustering WSDL Documents to Bootstrap the Discovery of Web Services, Proceedings of the 2010 IEEE International Conference on Web Services (ICWS 2010), Pages: 147-154, 2010.

[12] Seekda. http://webservices.seekda.com/

[13] Web Services Description Language (WSDL) http://www.w3.org/TR/wsdl/

[14] Google Translate. http://translate.google.com/

[15] WSDL4J. http://sourceforge.net/projects/wsdl4j/

[16] Document Object Model (DOM). http://www.w3.org/DOM/

[17] Cilibrasi, Rudi L, Vitnyi, Paul M. B., " The Google similarity distance," IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No. 3, pp. 370-383, March 2007.

[18] K-MEANS. http://en.wikipedia.org/wiki/K-means_clustering/

[19] D. Hand, H. Mannila, and P. Smyth. Principles of Data Mining. The MIT Press, 2001.

[20] Jain AK, Dubes RC, Algorithms for clustering data. Prentice-Hall, Englewood Cliffs, 1988.

[21] E. Sirin, J. Hendler, and B. Parsia. Semi-automatic com-position ofweb services using semantic descriptions. In WSMAI-2003, 2003.

[22] Michael P. Papazoglou, Paolo Traverso, Istituto Ricerca, Scientifica Tecnologica, " Service-Oriented Computing: State of the Art and Research Challenges," Computer, VOL. 40, NO. 11, pp 38-45, Nov. 2007.

[23] Eyhab Al-Masri, Qusay H. Mahmoud, "Investigating web services on the world wide web," International World Wide Web Conference (WWW 2008), pp. 795-804, 2008.

[24] Richi Nayak, " Data mining in Web services discovery and monitoring ," International Journal of Web Services Research, Vol. 5, No. 1, pp. 63-81, January, 2008.

[25] Zibin Zheng, Hao Ma, Michael R. Lyu, and Irwin King. "QoS-Aware Web Service Recommendation by Collaborative Filtering". IEEE Transactions on Services Computing, 2010.

[26] Zibin Zheng, Michael R. Lyu, "Collaborative Reliability Prediction for Service-Oriented Systems", in Proc. 32nd ACM/IEEE International Conference on Software Engineering (ICSE2010), Cape Town, South Africa, May 2-8, 2010, pp. 35-44.

[27] L. Kaufman and P. J. Rousseeuw. "Finding Groups in Data: An Introduction to Cluster Analysis". John Wiley& Sons, New York, 1990.

[28] John Makhoul, Francis Kubala, Richard Schwartz, Ralph Weischedel, "Performance measures for information extraction," DARPA Broadcast News Workshop, Herndon, VA, February 1999.

[29] J. A. Aslam and M. H. Montague, "Models for Metasearch," in SIGIR, 2001, pp. 275–284.

[30] S. Cetintas and L. Si, "Exploration of the Tradeoff Between Effectiveness and Efficiency for Results Merging in Federated Search," in SIGIR, 2007, pp. 707–708.

[31] R. A. Baeza-Yates and B. A. Ribeiro-Neto, Modern Information Retrieval. ACM Press / Addison-Wesley, 1999.

[32] http://en.wikipedia.org/wiki/Amazon_Web_Services#cite_note-1

[33] V. Murdock, D. Kelly, W. B. Croft, N. J. Belkin, X. Yuan, Identifying and Improving Retrieval for Procedural Questions, Information Processing & Management, Volume 43, Issue 1, January 2007, Pages 181–203.