Web Services Searching based on Domain Ontology

Xu Bin, Wang Yan, Zhang Po, Li Juanzi
Department of Computer Science, Tsinghua University, Beijing 100084, P.R.China
xubin@tsinghua.edu.cn,
{wang-yan,zhangpo03}@mails.tsinghua.edu.cn, ljz@keg.cs.tsinghua.edu.cn

Abstract

Searching proper web services is the basic step to composite web services into applications. Current searching in UDDI servers is based on taxonomy and tModel, which is not convenient to find domain related web services. In this paper, we propose a method to search web services based on domain ontology. Firstly the WSDL crawler collects the WSDL files from the Internet resources like Google, Baidu and XMethods as much as possible. Secondly ontology is used to represent the domain, such as travel ontology. Thirdly a Support Vector Machine (SVM) classifier is constructed to select the domain WSDL files from the collected WSDL files; domain vector is built according to the domain ontology, and features are extracted from WSDL files to train the SVM classifier. Finally we evaluate the method through experiment and show that the method is effective.

1. Introduction

Web services are new paradigm to construct distributed applications in the web. Technologies like WSDL, SOAP and UDDI constitute the current standards of web services. Discovery and composition of web services are key steps to build web applications. Nowadays web services are published and searched in central registry of UDDI servers. It is through taxonomy and tModel to encourage the registrants to categorize their businesses, services and service descriptions. Therefore it is also through them for the businesses to find each other and the services meet their needs.

Composition of web services has received much interest to support B2B or enterprise application integration. Applications are to be assembled from a set of appropriate web services and no longer be written manually. Seamless composition of web services has enormous potential in development distributed application.

To composite web services into new application, it is often inside a domain to search related web services. For example, to develop an application about travel, the developer should search web services about booking flight ticket, reserving hotel and car, and weather etc. A domain is a specific area, which is represented by an ontology in this paper, domain. For example, the travel ontology[1] is about concepts of traveling, while the wine ontology[2] is about wine's taste and type. Current web services' searching is based on tModel of UDDI, therefore, it is not convenient to search web services in a domain. The developer have to search through keywords like "travel", "flight", "hotel", "car", "weather" separately. Furthermore, web services which are not registered in UDDI servers can't be discovered through tModel.

At the same time, the operation name and description in WSDL file often give hints about the domain of the web services. Such as in web services of "Global Weather"[3], operation name of "GetWeather" and its description "Get weather report for all major cities around the world" have obvious meaning about weather. Hence, WSDL files contain the information about its domain.

In this paper, we propose a method to search web services according to a domain ontology. Firstly, to get as much as possible WSDL files directly in the Internet, the WSDL crawler collects WSDL files from the web pages in XMethods[1], Google[2] and Baidu[3]. Secondly, to represent a travel domain, the travel ontology is given. Then domain vector is built according to the ontology. Thirdly, features are extracted from selected WSDL files to form the training set of SVM[4] classifier. Experiment shows that the method is effective.

[1] www.xmethods.com
[2] www.google.com
[3] www.baidu.com

Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE'05)
0-7695-2438-9/05 $20.00 © 2005 IEEE

The rest of this paper is structured as follows: in section 2, we present the WSDL crawler and its result. In section 3, we describe the travel ontology. In section 4, we describe the SVM classifier, and in section 5, we give the experiment result and its accuracy. Section 6 demonstrates the application of searching web services based on domain. Section 7 gives the related works. We make concluding remarks in section 7.

2. WSDL Crawler

Each web service has a WSDL file to describe its interface and functions. Besides getting WSDL file through UDDI servers of IBM[4] or Microsoft[5], most WSDL files are directly put in the Internet. So in this paper the WSDL crawler tries to search WSDL files directly in the Internet.

The crawler crawls from three web sources: XMethods, Google and Baidu. XMethods is a website to collect available web services. It lists the web services' publisher, style, service name, description and implementation on web pages. The crawler gets the full list page from XMethods and analyzes its links to each WSDL file. Then the crawler visits all the links to save the WSDL file and the description text file.

The crawler uses the search engine Google to get web pages containing WSDL in their URL. Searching sentences of "inurl:wsdl" or "inurl:asmx" are used in Google, and pages including the link to WSDL or ASMX file are returned. The crawler analyzes the pages and saves the WSDL files.

Baidu is a famous Chinese search engine. Searching sentences of "inurl:wsdl" or "inurl:asmx" are also used in Baidu to get web pages including the link to WSDL or ASMX file. The crawler also analyzes them and gets WSDL files.

The simple algorithm of the crawler is as following:

```
Vector PageList=null;
Vector OnePageURLs=null;

switch(source){
   case GOOGLE:
      clear PageList and OnePageURLs;
      PageList.add("http://www.google
.com/search?hl=zh-
CN&q=allinurl%3Awsdl&lr=");
      PageList.add("http://www.google
.com/search?hl=zh-
CN&q=allinurl%3Aasmx&lr=");
      save PageList to google.log;
      get every OnePageURLs from
google.log;
      get WSDL files from the links
in OnePageURLs and save them in
directory google;
      break;

  case BaiDu:
      clear PageList and OnePageURLs;
      PageList.add("http://www.baidu.
com/s?wd=inurl%3A%28wsdl%29&cl=3");
      PageList.add("http://www.baidu.
com/s?wd=inurl%3A%28asmx%29&cl=3");
      save PageList to baidu.log;
      get every OnePageURLs from
baidu.log;
      get WSDL fils from the links in
OnePageURLs and save them in
directory baidu;
      break;

   case XMethods:
      clear PageList and OnePageURLs;
      pagelist.add("http://www.xmetho
ds.net/ve2/Directory.po;jsessionid=rd
z1afIp9Yxz2bWek5I1D34U(QHyMHiRM)");
      save PageList to xmethods.log
      get every OnePageURLs from
xmethods.log;
      get WSDL files from the links
in OnePageURLs and save them in
directory xmethods;
      break;
   }
```

The web pages returned from Google or Baidu may link to the same WSDL, then the crawler removes the repeated WSDL. For example, after removing the repeated WSDL from the 55679 links in Google, there are only 357 WSDL files left.

Table 1. Quantity of Web Services

Source	XMethods	Google		Baidu	
		wsdl	asmx	wsdl	asmx
Quantity	422	154	357	7	36

Table 1 lists the quantity of web services got from the three sources. There are totally 976 WSDL files collected. Full list of XMethods commonly includes 422 links to WSDL files. Though the links from Google are more than 50000, only 511 WSDL files left because of repeated ones. Baidu is only 43 WSDL files left.

[4] uddi.ibm.com
[5] uddi.microsoft.com

3. Our Travel Ontology

Our travel ontology in this paper is developed through combining concepts from OTA[5] and some other mature travel services. OTA (OpenTravel Alliance) defines a common information representation form for travel domain using XML, to make sure the different travel industry companies can transmit information seamless through WWW. OTA now has over 150 travel industry company members, so it is very valuable for the definition of travel ontology. Now, our travel ontology has 51 classes and 119 properties, in OWL-DL[6] format, including booking air ticket, hotel and car rental sub travel domains. This travel ontology has been published in SchemaWeb[1] or can be got at our website[6].

According to the various categories of ontology information, this Ontology can be separated into two main categories: Domain Ontology and Domain Independent Ontology. The Domain Ontology contains domain related ontology information, while the Domain Independent Ontology contains generic information such as "*Time*", "*Date*", "*Money*", "*Person*", "*Location*" and so on.

The more detailed category in Domain Ontology is: Travel Domain. And the SubDomain contains three uncrossing categories: Car Domain, Hotel Domain and Flight Domain.

Those classes in our Travel Ontology are related by two kinds of relations. One is the hierarchy relation, such as "subClassOf", the other is the reference relation, such as ObjectProperty of a class. When any two classes in our Travel Ontology can be related by some kind of relations, we say the Ontology has connectivity and is complete.

4. SVM Classifier for WSDL

The Support Vector Machine (SVM) is a learning methodology based on Vapnik-Chervonenkis (VC) theory.[4] SVM have received much attention and have been applied to a number of practical problems ranging from text classification to image processing to optical character recognition. We constructed a SVM classifier to select the domain related WSDL files.

We trained the SVM based on our travel ontology. Firstly, we generated a travel domain vector from our travel ontology, which detailed in section 4.1. Secondly, to form the training set for the SVM classifier, we had to manually select positive and negative WSDL files from all the WSDL files listed in table 1. Positive WSDL files are the WSDL files about travel, while negative WSDL files are not.

4.1 Domain Vector

Domain vector is the keywords list extracted from all the classes and properties of domain ontology to represents the characters of the domain. Such as keyword "weather" can be extracted from the class *weather*, and keyword "car" and "rental" can be extracted from the property *hasCarRental*. Stop word list is used during keyword extraction. For example, the domain vector of travel ontology is:

$DomainVector_{travel}$=[Travel, Car, Weather, Hotel, Airport, Airline, Flight, Ticket, Aircraft, Reservation, Room, Restaurant……]

Domain vector has two usages, the first one is to select the possible positive and negative WSDL instances, and the second one is to extract features from the WSDL file.

4.2 Selecting WSDL files about travel

The crawler collects 976 WSDL files. Not all of these files are about travel, but to form the training set of SVM classifier, we must manually select the WSDL files about travel to act as positive instances, and select WSDL files no related to travel to act as negative instances. Firstly, To reduce the manual selecting works, we use the travel domain vector to filter all the WSDL files and get the result set $WS_{filtered}$. Any WSDL files containing one of the keywords in the domain vector could be put to $WS_{filtered}$. Then we look at every WSDL file in $WS_{filtered}$ to decide whether it is about travel, if yes then put it in the positive WSDL instances set $WS_{positive}$. Secondly we select the negative WSDL instances set $WS_{negative}$ from the web services except $WS_{positive}$. Then, we extract features from $WS_{positive}$ and $WS_{negative}$ to form the training set for SVM.

4.3 Extracting Features from WSDL file

Every WSDL file represents the function and interface of a web service. In WSDL file, the tag <*wsdl:operation name="……"*> which describes the interface name has obvious meaning. Such as one fragment of the WSDL file about weather[3]:

```
<?xml version="1.0" encoding="utf-8"?>
  ……
<wsdl:portType name="GlobalWeatherSoap">
<wsdl:operation name="GetWeather">
<documentation xmlns="http://schemas.xmlsoap.org/wsdl/">
```

[6] http://keg.cs.tsinghua.edu.cn/persons/zp/travel_onto_is.owl

```
Get weather report for all major cities around the world.
</documentation>
<wsdl:input
message="tns:GetWeatherSoapIn" />
<wsdl:output
message="tns:GetWeatherSoapOut" />
</wsdl:operation>

<wsdl:operation
name="GetCitiesByCountry">
<documentation xmlns=
"http://schemas.xmlsoap.org/wsdl/">
Get all major cities by country
name(full / part). </documentation>
<wsdl:input message=
"tns:GetCitiesByCountrySoapIn" />
<wsdl:output message=
"tns:GetCitiesByCountrySoapOut" />
</wsdl:operation>
</wsdl:portType>
```

……

In the above WSDL file, the first operation name is "*GetWeather"*, which can be separated into two words "*Get"* and "*Weather"* through the capital letter. And the words in the tag <*documentation*> are "*Get weather report for all major cities around the world*". All these words are information of the web service. So features are selected from these words. The feature of one WSDL file is a word vector. We counted the term frequency[7] from all the words appearing in the tag <*wsdl:operation name="……"*> and <*documentation*> , but every term should be keyword of the travel domain vector. The word vector is as following:

$Vector_{WSDL}=[w1, w2, w3\ldots\ldots w_i]$

$$w_i = \frac{TFi}{\sum_{1}^{i} TFi}$$

Finally, we generated the training set for the SVM classifier. The training set is a data set containing many lines. Each line is a word vector of one web service. And all the web services in the training set are from the positive set $WS_{positive}$ and the negative set $WS_{negative.}$

5. Experiment

To test the accuracy of SVM classifier, we formed the training set and test set from $WS_{positive}$ and $WS_{negative}$. We selected 32 WSDL files from $WS_{filtered}$ to form $WS_{positive}$, and 48 WSDL files to form $WS_{negative}$. WSDL files in $WS_{positive}$ and $WS_{negative}$ were divided into five groups separately, we chose four groups randomly to form the training set for SVM to learn, and the left one group was used for testing. For each WSDL file in the training set, its vector $Vector_{WSDL}$ was calculated and written as one line into *train.dat* file for SVM. In the same way, each WSDL file in the testing set was calculated its vector $Vector_{WSDL}$ and written as one line into *test.dat* file.

Therefore, there was a total of five testing results:

Table 2 Accuracy on test set

Training set	Group 2,3,4,5	Group 1,3,4,5	Group 1,2,4,5	Group 1,2,3,5	Group 1,2,3,4
Testing Set	Group1	Group2	Group3	Group4	Group5
Accuracy	82.35%	87.50%	87.50%	100.0%	94.12%

Table 2 lists the training set, testing set and prediction accuracy of the SVM classifier. Each group contains from 14 to 17 WSDL files. The highest test accuracy is 100%, the lowest accuracy is 82.35%. And the average accuracy of the five tests is 90.29%. This result shows that our method is effective to serach domain related web services.

6. Application of Searching Web Services based on Domain

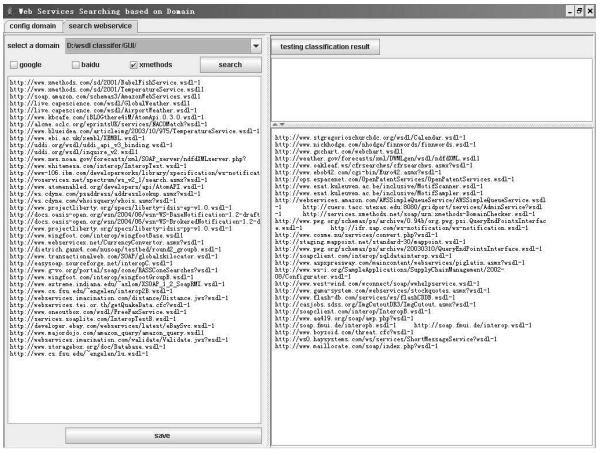

Figure 1 Interface of searching web services

We developed an application for searching web services based on the WSDL crawler, our travel ontology and SVM classifier. There are two interfaces in the application. One is searching web services interface, the other is domain configuration interface.

In the interface of searching web services, user can choose any sources of Google, Baidu and XMethods to collect new WSDL files from the Internet, and choose the domain which he wants to search from the list menu. Then the application starts to crawl all the WSDL files from the selected sources, and compares

with the WSDL files in log, then updates the log with the newest files. Then the SVM classifier decides every WSDL in log whether it is in the selected domain, and lists them in the left window. The right bottom window will show the description of any WSDL files selected from the left window. Because the WSDL files in the web will update from time to time, this interface can get the newest domain related web services.

In domain configuration interface, user can import new ontology, and manually choose the positive instances of WSDL files from all the files in log. The negative instances are automatically selected from the WSDL files. Then the SVM classifier starts to learn the positive instances and negative instances of this domain, and finally gets the precondition rule model of the domain. This interface helps the user to import many kinds of domain ontology, and broaden the searching domains in the first interface.

7. Related Work

WSMO[8] is an ontology for describing various aspects related to semantic web services, which is supported by the SWWS(Semantic Web enabled Web Service) project from European Union. WSMO is based on WSMF(Web Service Modeling Framework). WSMF consists of four different main elements for describing semantic web services: *ontologies*, *goals*, *web services* and *mediators*. The focused crawler for web services discovery[9][10] in WSMO is mainly for searching the semantically annotated web services, in WSDL descriptions or in UDDI/ebXML registries, or finally the semantic web services written in WSML[11]. This crawler is based on keyword searching, is not domain related and has no SVM classifier.

The BINGO! focused crawler[12] is a relatively new, promising approach to improving the recall of expert search on the Web. It typically starts from a user or community-specific tree of topics along with a few training documents for each tree node, and then crawls the Web with focus on these topics of interest. It uses a linear SVM to classify the high-quality documents for expert web search. Though the crawler is towards fully automated portal generation, it is not mainly for WSDL crawling.

8. Conclusion and Future Work

This paper proposes a method to search domain related web services from the Internet. WSDL files often represent the domain information of web services. We use ontology to describe the domain, and generate the domain vector. We construct the SVM classifier to select domain related WSDL files from all the WSDL files collected by the crawler. Positive and negative instances of WSDL files forms training set for SVM classifier. Experiment shows that our method is effective.

There are some works to do in the future. Such as extending the domain scopes from our travel ontology to other ontologies, or increasing the accuracy of SVM classifier by selecting more features from WSDL, or even from semantic descriptions like OWL and WSMO.

References

[1] Po ZHANG. Travel Ontology. http://www.schemaweb.info/schema/SchemaDetails.aspx?id=236, 2005.

[2] http://www.w3.org/TR/owl-guide/wine.rdf

[3] http://www.webservicex.com/globalweather.asmx?WSDL

[4] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1995.

[5] http://www.opentravel.org/

[6] http://www.w3.org/2004/OWL/

[7] Salton, G., and Buckley, C., Term-Weighting Approaches in Automatic Text Retrieval, *Information Processing &Management*, 24(5), pp. 513-523, 1988.

[8] D. Roman, H. Lausen, U. Keller, C. Bussler, D. Fensel, M. Kifer, E. Oren, C. Priest, M. Stollberg. D2v1.1. Web Servcie Modeling Ontology(WSMO), WSMO Working Draft 10 October 2004. http://www.wsmo.org/2004/d2/v1.1/20041010/

[9] David Aiken. D10.1v0.1 Focused Crawler for Web Service Discovery, WSMO Working Draft 13 April 2005 http://www.wsmo.org/TR/d10/d10.1/v0.1/20050413

[10] David Aiken. B2C Web Service Discovery http://sws.deri.ie/members/david/publications/B2CWSD.pdf

[11] J.D Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, D. Fensel, D16.1v0.2 The Web Service Modeling Language WSML, WSML Final Draft 20 March 2005 http://www.wsmo.org/TR/d16/d16.1/v0.2/20050320/

[12] S. Sizov, J. Graupmann, M. Theobald. From Focused Crawling to Expert Information: an Application Framework for Web Exploration and Portal Generation. *VLDB2003: Berlin, Germany* pp1105-1108