Journal of Mines, Metals and Fuels, 71(5): 645-649; 2023. DOI: 10.18311/jmmf/2023/34164

Print ISSN : 0022-2755

Journal of Mines, Metals and Fuels

Contents available at: www.informaticsjournals.com/index.php/jmmf

Dynamicity of Web Services and Web Scrapping

Sumathi Pawar* **and Manjula GuruRaj***

NMAMIT, Nitte, Karkala, Karnataka, India. E-mail: pawarsumathi@gmail.com / manjulagururaj@nitte.edu.in

Abstract

This research focuses on dynamic Web service link searching using web scraping (WS). WS is method of extracting required contents from search engine results. The proposed system inputs the functional word to search engine and retrieves required Web services from search engine results. WS method retrieves the links of the Web Service Definition Language from which service and operation elements are extracted and processed to check suitability of Web services. Matched Web services will be clustered and invoked. This research is implemented using J2EE and remote Web services are executed according to user requirement. Analysis of results justified that this will be a suitable search method to search for remote Web services and to execute using open source software.

Keywords: *WS, Hash-Map, Web Service, Clusters, WSDL, INVOKE.*

1.0 Introduction

WS is a technique for obtaining the appropriate and necessary data from the Web. Web services are remote services that deliver a range of operations when they are executed remotely and produce results. The Web Service Definition Language, or WSDL, is an XML-based description of Web Services that lists all input and output elements of operation names along with their data types and schemas.

The suggested study makes use of a search engine and the query "functional word? WSDL" to look up the desired functional word as Web services. This search returns a group of Web services with WSDL files that are referred to as "community Web services" and are semantically related to the functional term. For each functional term that is requested, the cluster returns many WSDL links that have semantically related operations. But among these outcomes, some of the functions might not correspond to those that the customer requested. Therefore, in order to locate Web services that are appropriate, each WSDL must be processed. Invoking Web services and processing WSDL are done using Java-based APIs. The methodology shows how to find the right WSDL linkages via web scraping, along with a block diagram.

The structure of this paper is review of the literature carried out in the section 2. WS is concentrated in the section 3. The results and discussion is elaborated in the section 3 and the final conclusion and future enhancement is depicted in the section 4.

2.0 Review of Literature

The extensive literature review discusses the current procedures for choosing and composing Web services.

In earlier Web service discovery research, the emphasis was on dynamic Web service discovery using Web and WSDL processing[1].

In prior studies on the creation of the WSDL weight matrix, the emphasis was on selecting appropriate Web services based on the highest computed weight of several related Web services was proposed by Sumathi et al[2].

Divya Sachan et al[3] proposed the Ontologies and mining are used to select Web services that are semantically similar. The ranking method is used to get the QoS parameters of numerous functionally comparable Web services. In order to rank the Web services, authors get user requirements as

*Author for correspondence

inputs and take into account user comments and QoS criteria..

With the help of a cutting-edge idea called skyline, it is suggested that the best user-desired service can be chosen without user participation is proposed by R.Joseph Manoj et al[4]. Service schema and service model are the two key elements that service model defines. The service graph used to depict composite Web services has a list of steps that must be taken in order to complete the task. The operations are then sorted in this graph using topological sort, which prioritizes dependencies among them. Service For each request, an execution plan is dynamically constructed. The QoS elements of each Web service operation are saved in the repository and indexed by operation IDs because Web service operations are static. According to the QoS considerations, execution plans that are dominated by other plans are filtered out.

[5] The open source UDDI specification is implemented in Java as jUDDI, which is pronounced "Judy." JDK 1.6 or later supports jUDDIv3 services. Numerous preset tModels, such as Quality of Service Metrics, are included in jUDDI. This was tested with Apache Open JPA and Hibernate and is based on JPA-standard interfaces.

The suggested research does not make use of UDDI because there is no UDDI register. Because of this, using the Web services is the only option to test Web service QoS measurements.

[6]WS-Policy lays out the requirements for offering the service. After reading these terms, a customer decides whether or not to utilize this service. The fundamental assertions "wsp:exactlyone>" and "wsp:all>" denote a set of policy claims.

- The binding WSDL components, as well as the input, output, fault, operation, and port elements, can contain WS-Poilcy assertions.
- It may be added as one of their child elements, or it may be referenced using the wsp: PolicyReference > element.

[7] The WSDL analyzer is a free tool that can be used to verify the accuracy of the WSDL and provide details on each of its components. This system accepts a WSDL as input and outputs every element and all of its children. WSDL analyzer verifies and returns the WSDL elements but this system cannot extract these elements from the browser, and doing so would make it difficult to use. In order to use Web services, the suggested research merely collects WSDL elements from the WSDLHelper API.

[8] The language used to describe how the services behave is called XLANG. By adding a behaviour element, XLANG improves the WSDL language for a Web service. The terms "WSDL operations" and "timeout operations" define two different categories of actions. The XLANG contract element is also used to connect many XLANG service descriptions to one another. The context element supports transactions that, in the event of failure, result in cancellation or rollback of the transactions. The standards mentioned provide XLANG data type to BPEL4WS data type conversion.

[9] The focus of an earlier version of this work's published article was on fully implementing WSDL processing and calling Web services using the WSDLHelper class.

WSDL processing and clustering are used in the proposed system to choose appropriate Web services. If the user-requested input parameters are not present in the chosen Web services, Web service composition or integration is performed. Consequently, a thorough assessment of the literature is done on web service composition and selection.

3.0 WS: WEB Scraping

The sole purpose of WS is to extract the necessary data from web search results. When a query is given as Query word?WSDL, search engine retrieves all the links that results in matching of query word either syntactically or synonymically. The number of links retrieved by search engines depend on existing WSDL files in the real time. For example if user enters the query "weather?WSDL", the result returned will be all the links related to "weather" Web services. As given in the partial code snippet of Figure 1, "Jsoup" API is used to retrieve the contents form Web according to the given query. In the retrieved results the result returned by object "dc" is processed for anchor links with the method "dc.getElementsById("a")". The variable element link holds all the hyperlinks which have "anchor" elements. From these hyperlinks only WSDL links are extracted using Jsoup API.

In addition to retrieving WSDL links from the

```
Document dc =Jsoup.connect("http://www.bing.com/search?
                q="+Query_word+"%3Fwsdl").get();
Elements lnk = dc.getElementsByTag("a");
      //get links present in HTML page
      for(Element lnk : links)
{String Hreflink = lnk.attr("href");URL.add(Hreflink);}
```

Figure 1: Partial coding for WSDL Retrieval

aforementioned search query, the Web services' information are also returned. Only WSDL file links are extracted, and the number of WSDL links will be counted. To extract the "service" element, the WSDL files that were retrieved here are processed. This element contains the precise "service" name that the given Web service offers.

Many methods for processing WSDL definition is stored in javax.wsdl.Definition class.

WSDL helper APIs are used to process extracted WSDL

```
Definition def = w.getDefinition(dfile)
```

files. As shown in the code statement "def" object is created by "Definition" class and parameter given to this method is particular WSDL file(which is stored in 'dfile' object.

Above statement uses method "getAllServices()" to

```
sampleMap2 = def.getAllServices();
```

extract service elements from WSDL file and stored in the variable sampleMap2. The hash map structure is where the service element should be kept. As a result, Map "sampleMap2 = new HashMap()" creates the WSDL structure is stored in the hash map.

The hashMap sampleMap2 is where the aforementioned statement keeps all service items. "serviceName=qName.getLocalPart()" then creates an object of the QName class and stores the hashMap's key portion in the serviceName.

Using pattern match with regular expression, it is possible to determine whether a given functional word matches the name of a particular service.

To discover <service> elements that match user-requested functions, the regular expression of pattern matching is used. To determine the search precision or support of the provided search, apply equation 1 in the following step.

$$\text{Support} = \frac{\text{Count of Matched Service Name to functional word}}{\text{Count of Retrieved WSDL}} \quad (1)$$

Case 1: In order to obtain information regarding operation names, input/output parameters, pre-conditions, effect elements, etc., those WSDL entries are processed once more if support is more than zero. Here, it is noted that there are two approaches that can be used to get this information from the WSDL.

The first approach entails placing all necessary WSDL in various files in the local repository.

The second approach involves downloading each WSDL from the internet, overwriting it on a local file, and then processing the resulting WSDL files. To keep all of these WSDL in several local repositories, the first option needs a

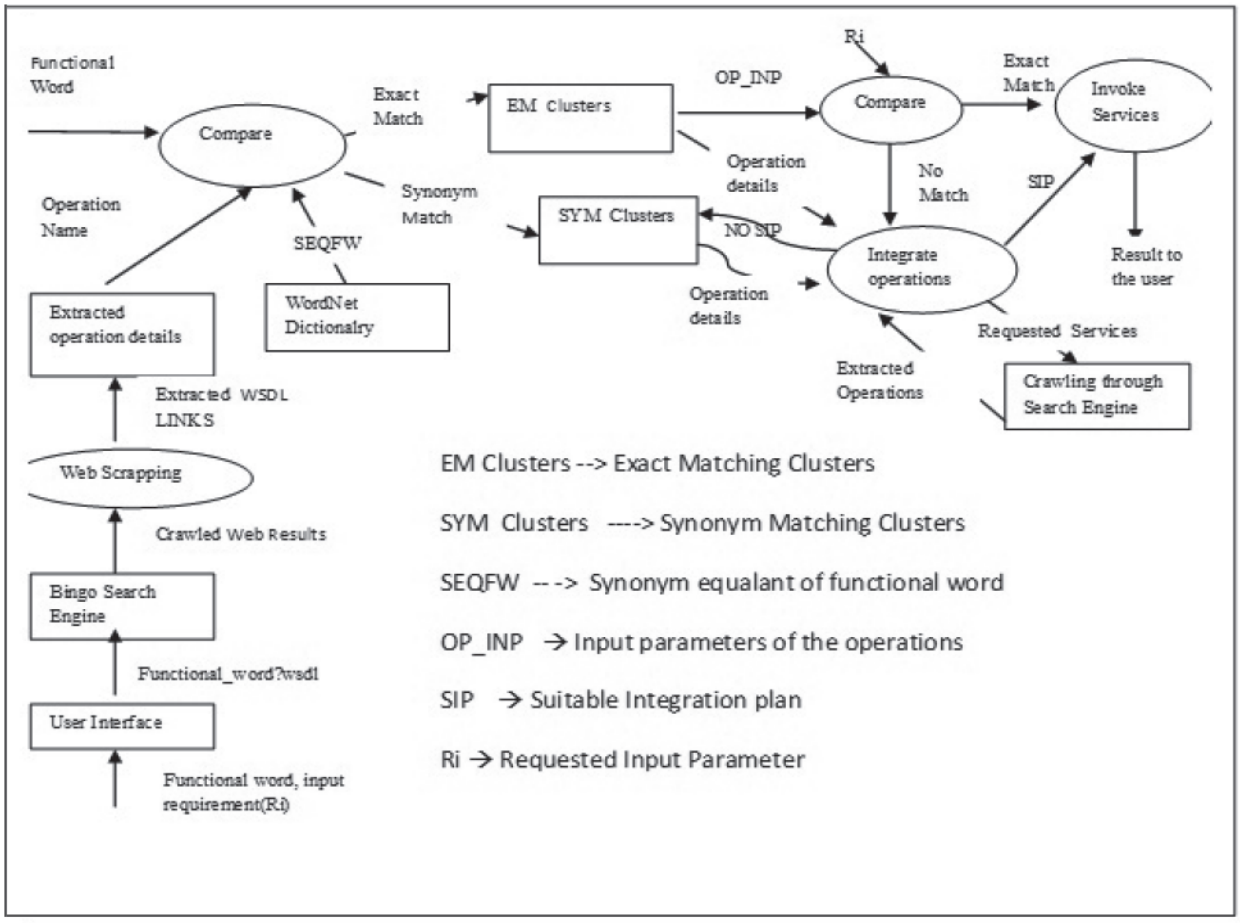


Figure 2: Web Scrapping Flow Diagram

large amount of storage. Internet bandwidth is used by the second technique to download each WSDL for processing. The process is described in Figure 2.

The accuracy of search results is calculated in terms of confidence and support, and functional words that are used in web service operations are discovered based on these values. These current Web services' WSDL documents are extracted and parsed. The appropriate Web services are chosen based on the necessary metrics.

Depending on how closely the input parameters of various web services match the user-provided inputs, the functional term provided by the user may exist as a single web service or as the combined/integrated results of other online web services.

As shown in Figure 2, given a user request, the expression <E = F, Ri> includes the functional word 'F' and the input 'Ri'. The returned WSDL URLs of the available Web services for the user-requested functional word are contained in the set "OW". Let 'n' be a part of the finite set of natural numbers, and let LW=<wsd_1, wsd_2, wsd_3,... wsd_n>. If 'F' appears as the operation name of any of the WSDL in the set LW and the input Ri matches one or more of the input parameters for these operations, the Web service is chosen.

- EM Clusters: Assume that <OPC = p1,p2..pn> refers to operations that exactly match the specified functional word, and that EM represents a cluster of such operations.
- SM Clusters: These clusters are created from operations where the names of the retrieved operations match the specified functional words in a synonym.

A. Embedding the operational process

When these clusters' input parameters <Cin=in_1,...in_n> do not match the user-supplied input Ri, integrating operations are necessary. The input components for these activities must then be retrieved, and the output components of already-existing Web services should be sought. Assume that <IN= inp_1, inp_2....inp_n> refers to the extracted input parameters of the i^{th} operation of the set OPC, where I is an integer between 1 and n, and 'n' is a part of the finite set of natural numbers. The procedure of integrating the Web services now involves looking for these parameters as "operation" / "output" of the currently available online Web services and feeding it as the input of the operation of set OPC.

B. Exploring Search Engines

This is the procedure for using web scraping to find the necessary functionally appropriate Web services and returning results to the web services integration process. Web crawling process is used to deeper search of the extracted links by repeatedly inputting extracted elements of WSDL into search engine until requirements are met.

4.0 Result

Table 1 displays the outcomes of web scraping when various functional phrases are entered into the search engine on the internet. J2EE and WSDL APIs from open source software are used in this system's implementation to process service specifications. The entire system will function as a client, and Apache Axis and Apache Tomcat software can be used to imitate the server system. If the user enters the keyword "Temperature" into this client system, four separate WSDL links are collected throughout the web scraping process. Table 1 of the availability of those services lists the names of the extracted services. Similar results are displayed in the table for other requests.

Precision and recall of different queries are conducted and it is found that according to availability of real time services, Web scraping method scores the result as given in the figure 3. It is found that query 4 has got precision 75% and recall 80% for its search, similarly other queries got different results as given in the Figure 3.

Here query1 is "weater?WSDL", query 2 is "currency?WSDL", query 3 is "GeoIP?WSDL" and query 4 is "StockMarket?WSDL". According to the availability of Web services, queries returns the proper results in the form of WSDL links. The precision and recall of the results are calculated using equation 2 and equation 3.

Table 1 WS Results

Requested Word	Extracted Service Name	WSDL Links after Web Scraping
Temperature	GlobalWeather	http://www.Webservicex.com/globalweather.asmx?WSDL
	TemperatureConversions	http://Webservices.daehosting.com/services/TemperatureConversions.wso?WSDL
	WeatherForecast	http://www.Webservicex.net/WeatherForecast.asmx?WSDL
	CurrencyConvertor	http://www.Webservicex.net/CurrencyConvertor.asmx?WSDL
London Gold Fix	LondonGoldAndSilverFix	http://www.Webservicex.net/LondonGoldFix.asmx?WSDL
GeoIP	GeoIPService	http://www.Webservicex.net/geoipservice.asmx?WSDL

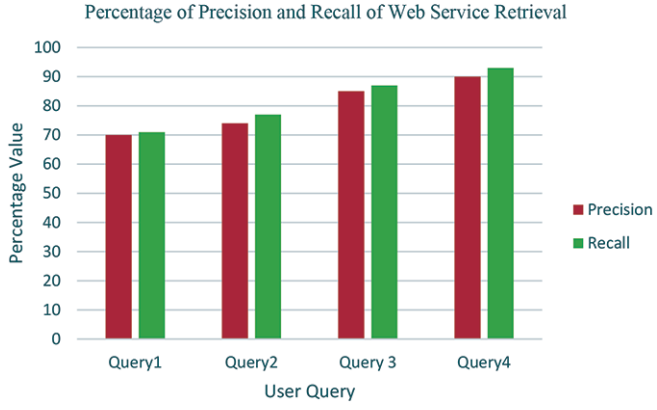


Figure 3: Precision and recall of the Web Scrapping

$$\text{Precision} = \frac{Retrieved \cap relevant}{retrieved} \quad \ldots (2)$$

$$\text{Recall} = \frac{Retrieved \cap relevant}{relevant} \quad \ldots (3)$$

Precision of the query1 is 70% and recall is 75%. Similarly, for query 2, query 3 and query 4, the precision and recall are 78%, 80%, 82%, 84%, 90% and 92%. According to the availability of real time services, precision and recall changes dynamically.

5.0 Analysis

Lower recall and better precision are employed for idea matching and parameter matching in existing papers[10]. When compared to other methods, Woogle search for Web services has the highest precision. WOOGLE's preferred UDDI registries are no longer available, and recommended systems now search for WSDL using search engines.

[11] Authors employed the UDDI-dependent VUSE for WSDL searches. For the search term "weather," VUSE returned 8 WSDL connections. However, none of the other Web services will function without these. VUSE relies on UDDI to retrieve the WSDL linkages even if it retrieves more links than our system. Many of the links between them don't offer genuine Web services or are inoperable. However, our approach makes use of a search engine rather than UDDI. As a result, it makes sense that the proposed system would be superior to the VUSE system. [12] In their Web Service Search System, the authors used OWL-S or semantic annotations for Web services (WESS). The contents of WESS include WSDL, OWL-S parsers, and web crawlers. Additionally, information is exchanged between WSDL and OWL-S.

6.0 Conclusions

Different search engines included in the analysis section demonstrate that utilizing a search engine to dynamically search for Web services is a quicker procedure, and the suggested method also gives information on the status of currently available Web services. The suggested method is quicker than the current system since Eclipse, an open-source program, is used to develop it. This system also has the benefit of being open-source software-implemented and adaptable enough to change in accordance with future requirements. One of the system's shortcomings is that users must express their needs using functional language. Future development will place a greater emphasis on Web service integration.

7.0 References

1. Sumathi, Niranjan N. Chiplunkar, Ashok Kumar A., (2014): Dynamic Discovery of Web Services using WSDL. In *I. J. Information Technology and Computer Science*, 10, 56-62.
2. Sumathi, Niranjan N. Chiplunkar, 'Populating parameters of web services by automatic composition using search precision and WSDL weight matrix', *Int. J. Computational Science and Engineering*, Vol. 17, No.3, pp. 333-342
3. Divya Sachan, Saurabh Kumar Dixit, Sandeep Kumar, (2013): "A System for Web Service Selection Based on QoS", International Conference on Information Systems and Computer Networks, IIT Roorkey, pp 139-144.
4. R. Joseph Manoj, Dr. A. Chandrasekar, (2013): "A Literature Review on Trust Management in Web Services Access Control", *International Journal on Web Service Computing* (IJWSC), Vol.4, No.3, September, pp 1-18.
5. http://juddi.apache.org, The Apache Software Foundation, Licensed under the Apache License, Version 2.0 Copyright © 2014.
6. Web Services Policy 1.5 – Framework http://www.w3.org/TR/ws-policy/, Retrieved on April 2022.
7. Online WSDL Analyzer, https://www.wsdl-analyzer.com/service/service/600982223? version=1, Retrieved on June 2021.
8. Microsoft. Web Services for Business Process Design (XLANG), http://xml.coverpages.org/xlang.html, Retrieved on Feb 2021.
9. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang. J., (2004): "Similarity Search for Web services" , Proceedings of VLDB, pp 372-383.
10. Ourania Hatzi, Georgios Batistatos, Mara Nikolaidou, Dimosthenis Anagnostopoulos, (2012): "A Specialized Search Engine for Web Service Discovery", International conference on Web Services Computing IEEE.
11. Chen, Wuhui, Incheon Paik. (2013) "Improving efficiency of service discovery using Linked data-based service publication." *Information Systems Frontiers* 15.4: 613-625.
12. Pawar, S., Chiplunkar, N. N. (2017): Open source APIs for processing the XML result of web services. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), September, pp. 1848–1854.
13. Sumathi Pawar, V. Nayak, G. Laxmi and N. N. Chiplunkar, (2020): "A Novel Web Service Search System Using WSDL," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, pp. 1252-1257.
14. Sumathi, Dr. Niranjan N Chiplunkar "Dynamic search and selection of Web Services", (2015): International Conference on Applied and Theoretical Computing and Communication Technology, TamilNadu, India IEEE.
15. Sumathi, Dr. Niranjan N Chiplunkar, (2017): "Dynamic Composition of Web Services by Service Invocation Dynamically", IJEME ISSN: 2305-3623 (Print), ISSN: 2305-8463, July, DOI: 10.5815/ijeme.2017.04.05, pp 41-50.
16. Sumathi, Pandith, K., Chiplunkar, N., Shetty, S. (2022): Dynamic Search and Integration of Web Services, Proceedings of 6th International Conference on Information and Communication Technology for Intelligent Systems, published in IOT with Smart Systems. Smart Innovation, Systems and Technologies, vol 312, 613-625.