Effective Web-Service Discovery Using K-Means Clustering

A. Santhana Vijayan[1] and S.R. Balasundaram[2]

[1] Department of Computer Science and Engineering,
National Institute of Technology, Tiruchirappalli – 620 015 Tamilnadu, India
vijayana@nitt.edu
[2] Department of Computer Applications,
National Institute of Technology, Tiruchirappalli – 620 015 Tamilnadu, India
blsundar@nitt.edu

Abstract. Web Services are proving to be a convenient way to integrate distributed software applications. As service-oriented architecture is getting popular, vast numbers of web services have been developed all over the world. But it is a challenging task to find the relevant or similar web services using web services registry such as UDDI. Current UDDI search uses keywords from web service and company information in its registry to retrieve web services. This information cannot fully capture user's needs and may miss out on potential matches. Underlying functionality and semantics of web services need to be considered. In this study, we explore the resemblance among web services using WSDL document features such as WSDL Content and Web Services name. We compute the similarity of web services and use this data to generate clusters using K-means clustering algorithm. This approach has really yielded good results and can be efficiently used by any web service search engine to retrieve similar or related web services.

Keywords: Web Service, WSDL document features, K-means Clustering, WV Tool.

1 Introduction

A Web service is a method of communication between two electronic devices over the web (internet). It is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language, known by the acronym WSDL). The term Web-service describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone [7]. XML is used to tag the data, SOAP is used to transfer the data (bind), WSDL is used for describing the services available (publish) and UDDI is used for listing what services are available (find). Fig.1. shows the web services triad that includes a broker, a service provider and a service requestor. Used primarily as a means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall.

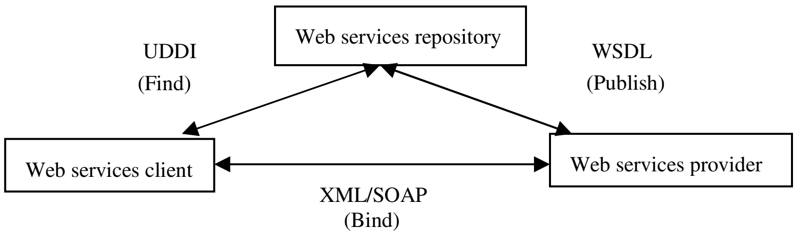

Fig. 1. The Web Services triad

1.1 Classes of Web Services

There are mainly two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations.

1.2 Problems Encountered in Retrieval of Non-semantic Web Services

There are so many problems encountered during the search for non-semantic web services. There is a difficulty in the discovery of non-semantic Web services through search engines as these engines do not recognize the Web service functionalities summarized in the WSDL file. Based on the Web service name, location and business defined in the WSDL file, the search engines partly relate the search terms entered by the user in order to retrieve the results back. Web service name is essential as a part of the search query in order to retrieve the exact service required. Hence, the user must take care in using the accurate keywords so that appropriate services can be obtained.

There is a possibility for the user to ignore services because of using alternate meanings for the keywords. For example, a service that includes ”motorbike” in its name may not be retrieved from a query looking for ”two wheeler”. In order to effectively improve the service discovery process by tumbling the search space, clustering techniques can be used to group similar services based on their functionality which improves the search engine retrieval.

The rest of our paper is organized as follows. Section 2 gives a brief introduction on existing methods and related work. The structure of WSDL documents is described in Section 3. Our proposed clustering approach is introduced in Section 4. The feature extraction from WSDL documents is explained in Section 5. Section 6 describes the feature integration process in order to set up the relation between Web services. Section 7 includes the experiments and results. Finally, Section 8 concludes our paper and summarizes future research activities.

2 Existing Methods and Related Work

Nowadays, service discovery has become a recent research issue because of increasing use of web services by most of the web application developers. WSDL documents are used to describe the non-semantic Web services where as Web ontology languages (OWL-S) [2] or Web Service Modeling Ontology (WSMO) [3] are used to describe the semantic web services. Non-semantic Web services are becoming more popular because of the support obtained from both the industry and development tools. Based on the various Web services description techniques, the process of service discovery is somewhat different. By using various web service description methods, non-semantic web services can be discovered whereas semantic Web services can be discovered using web ontologies such as OWL-S [2] and WSMO [3]. In our approach, we focus on the discovery of non-semantic Web services. According to the approach proposed by Nayak [5], the discovery of web services can be improved using the Jaccard coefficient that determines the similarity between Web services. With respect to other users' experiences on similar queries, Nayak [5] give the users with associated search terms. In our approach, the search space is reduced by clustering the Web services based on their functionality. We extract two features such as WSDL content and web service name to compute the similarity between Web services. We modify the approach used by Kahlid Elgazzar et. al [1].

3 WSDL Document Structure

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings[7]. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types, which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitutes a reusable binding.

A WSDL document uses the following elements in the definition of network services:

<Types> – a container for data type definitions using some type system (XSD).

<Message> – an abstract, typed definition of the data being communicated.

<Operation> – an abstract description of an action supported by the service.

<Port Type> –an abstract set of operations supported by one or more endpoints.

<Binding> – a concrete protocol and data format specification for particular port type.

<Port> – a single endpoint defined as a combination of binding and network address.

<Service> – a collection of related endpoints.

WSDL does not introduce a new type definition language. WSDL recognizes the need for rich type systems for describing message formats, and supports the XML Schemas specification (XSD) as its canonical type system [7].

4 Proposed Clustering Method

Our proposed method is based on the information available in WSDL documents. We find the WSDL documents in order to extract two features such as WSDL content and web service name which describe the semantic and behaviour of the Web service. The functionality of a Web service can be revealed from these features [5]. These features are then integrated together such that the web services can be clustered to form similar groups based on their functionally by using K-means clustering algorithm. A service search engine can use this step as a precursor in categorizing the Web services with users' requests.

5 Feature Extraction from WSDL Document

This section describes how the two proposed features such as WSDL Content and Web Service Name can be extracted from WSDL documents. Fig. 2. illustrates the steps involved in feature extraction process.

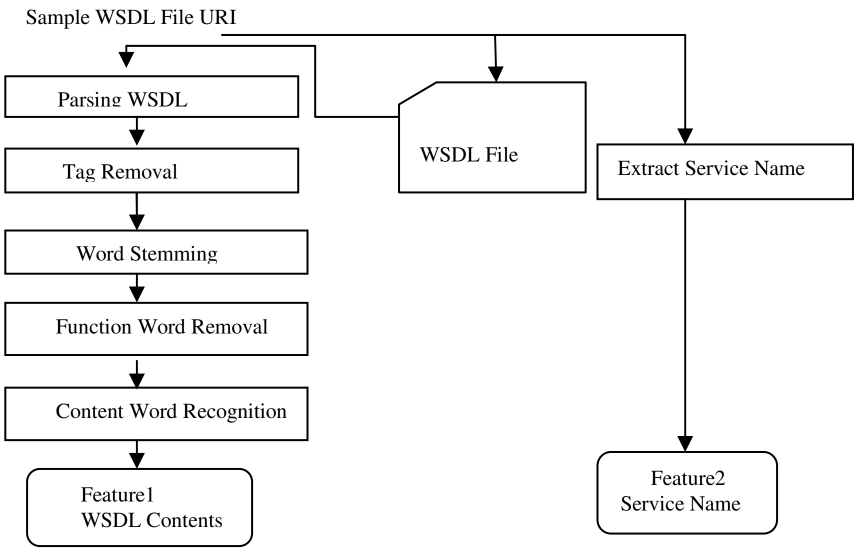

Fig. 2. Block diagram of feature extraction process

Feature 1: WSDL Content

The WSDL URI can be used to read the WSDL document contents. Let each f_i denote a WSDL document which describes a Web service s_i. A vector of meaningful content words for the given Web service s_i can be extracted by means of processing the WSDL document contents. In our approach, the vector can be constructed by using the following five steps.

1) Parsing WSDL: A vector of tokens T_i can be produced by parsing the given WSDL document contents with respect to white spaces.
2) Tag removal: In order to obtain a vector consisting only of valid content words, all tokens from T_i that are part of a XML tag are removed. As all XML tags specified in a given WSDL document are predefined, the process of removing XML tags from the tokenized vector is simple.
3) Word stemming: With the help of Porter stemmer algorithm [11], only the relevant words in T_i are reduced to their base words. Tokens among a common stem will generally have the similar meaning, for example, 'establish', 'established', 'establishing', and 'establishment' all have the same stem 'establish'. With respect to word deviations in the semantic of a Web service, using one or all of the tokens will not make a distinction. But, the words that appear frequently are more important when compared to others. The number of occurrences will be considered in the following steps.
4) Function word removal: Function words are said to be autonomous with respect to one another. With the help of Poisson distribution to model word occurrence in documents [6], function words can be differentiated from content words. Using this step all function words from the service word vector can be removed. By calculating the overestimation factor for all words in the word vector, we can decide which word is a function word as follows:

$$V_{ij} = \frac{f_{ij}}{f_{dj}} \tag{1}$$

f_{ij} is the number of occurrences of term 'i' in document j.
f_{dj} is the number of terms occurring in document j.

The overestimation factor [6] for all words in T_i and the average $avg[\Lambda]$ of all overestimation factors can be calculated as follows. An overestimation factor threshold (Λ^{thre}) is formulated as follows [6].

$$\Lambda^{thre} = \begin{cases} avg[\Lambda] \text{ if } avg[\Lambda] > 1 \\ 1 \quad \text{otherwise} \end{cases} \tag{2}$$

Any word is said to be a content word if it has an overestimation factor above the Λ^{thre}. Otherwise the word is termed as a function word that should be removed from the vector T_i. By using this step, all function words from the service word vector can be removed.

5) Content word recognition: Certain general computing content words such as 'data','web','port', etc. are typically present in WSDL documents. We cannot distinguish between Web services based on these words as they appear in most WSDL files. The goal of this step is to eliminate words which do not correspond to the specific semantics of the Web service. The k-means clustering algorithm [8] with k = 2 on T_i is then applied to cluster the remaining words into two groups such that one group corresponds to the meaning of the Web service where as the other group is meant for general computing words. As the number of clusters is known earlier, K-means algorithm is used as it is simple, fast, and efficient. Normalized Google Distance (NGD) [9] is used as a featureless distance measure between two words x and y as follows.

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

where M is the total number of web pages searched by Google; f(x) and f(y) are the number of hits for search terms x and y, respectively; and f(x, y) is the number of web pages on which both x and y occur. If the two search terms x and y never occur together on the same web page, but do occur separately, the normalized Google distance between them is infinite. If both terms always occur together, their NGD is zero, or equivalent to the coefficient between x squared and y squared.

Based on the similarity factor calculated using Equation (5), similar web services can be grouped for efficient service discovery.

Feature 2 : Web Service Name

The composite name such as 'ComputerArchitecture' can be split up into multiple names based on the assumption that a capital letter indicates the start of a new word. The similarity between services names can be then found using NGD as follows:

$$\mathrm{sim}(\mathrm{sname}_i, \mathrm{sname}_j) = 1 - \mathrm{NGD}(\mathrm{sname}_i, \mathrm{sname}_j) \tag{4}$$

where sname_i and sname_j are the names of the Web services s_i and s_j respectively.

6 Feature Integration

K- means clustering algorithm is used to cluster similar Web services based on the two similarity features presented above as it is computationally inexpensive. The similarity factor $\Theta(s_i, s_j)$ between two Web services s_i and s_j can be measured as follows:

$$\Theta(S_i, S_j) = 0.5S(T_i, T_j) + 0.5\mathrm{sim}(\mathrm{sname}_i, \mathrm{sname}_j) \tag{5}$$

$\Theta(s_i, s_j)$ is equal to "1" if the two services are identical and

$\Theta(s_i, s_j)$ is equal to "0" if they are completely different.

We normalize $\Theta(s_i, s_j)$ by assigning weights of 0.5 to each of the two similarity features. We determined experimentally that these weights give reasonable results. In Equation (5), T_i and T_j are the content word vectors of services s_i, s_j respectively.

$S(T_i, T_j)$ is the average similarity between the content word vectors T_i, and T_j and is calculated with

$$S(T_i, T_j) = \frac{\Sigma a \, \varepsilon \, T_i \, \Sigma b \, \varepsilon \, T_j \mathrm{sim}(a, b)}{|T_i||T_j|} \tag{6}$$

where sim(a, b) is the featureless similarity factor computed between words a and b using NGD based on the word coexistence in Web pages. sim(a, b) is calculated using

$$\mathrm{sim}(a, b) = 1 - \mathrm{NGD}(a, b) \tag{7}$$

where a and b are the two most important content vector words belong to T_i and T_j respectively.

7 Experiments and Results

We use two criteria to evaluate the performance of our approach, namely Precision and Recall [10]. Precision and Recall have been often used to evaluate information retrieval schemes [4].We extend the use of these two measures to evaluate our approach as follows:

$$\mathrm{Precision} = \frac{\Sigma_{i \in C} P_{c_i}}{\mathrm{length}(C)}, \; P_{c_i} = \frac{\mathrm{succ}(c_i)}{\mathrm{succ}(c_i) + \mathrm{mispl}(c_i)} \tag{8}$$

$$\mathrm{recall} = \frac{\Sigma_{i \in C} R_{c_i}}{\mathrm{length}(C)}, \; R_{c_i} = \frac{\mathrm{succ}(c_i)}{\mathrm{succ}(c_i) + \mathrm{missed}(c_i)} \tag{9}$$

where c_i is the cluster i, P_{c_i} and R_{c_i} are precision and recall for cluster c_i respectively, $\mathrm{succ}(c_i)$ is the number of Web services successfully placed in the proper cluster c_i, $\mathrm{mispl}(c_i)$ is the number of Web services that are incorrectly clustered into c_i, $\mathrm{missed}(c_i)$ is the number of Web services that should be clustered into c_i but are incorrectly placed in other clusters, and length(C) is the number of clusters.

Our experiments are based on the WSDL files obtained from online Web service providers and brokers. The contents of the WSDL file can be directly obtained from the corresponding URI. Using K-means clustering algorithm, the WSDL documents are grouped together to form the following two categories such as "Computer Architecture" and "Weather".

The contents of the WSDL documents are parsed in order to produce the content word vector T_i. The next step is to obtain a vector consisting only of valid content words without the XML tags by using the java library word vector tool.

Using the Porter stemmer method, the vectors obtained in the previous step are reduced to their roots. Then the function words and content words can be differentiated by calculating the overestimation factor for all the words in each vector. The Term Frequency can be calculated using the WV Tool. The content words for the Web services can be then discovered by clustering each word vector into two groups using the k-means clustering algorithm, in which NGD is used as a featureless similarity measure between words.

We use Python as a scripting language for calculating the NGD and similarity factor. Feature extraction and integration is implemented using Java.

The word vectors and term frequencies generated as a result of feature extraction process for the sample web service categories such as 'weather' and 'computer architecture' are shown in Table. 1.

Table 1. Block Word vectors and Term frequencies generated for sample files

Term Frequency	Word Vector
0.760286	Computer
0.304114	technology
0.304115	year
0.076029	gener
0.152057	electron
0.076029	creat
0.076029	person
0.228086	perform
0.076029	memor
0.076029	stor
0.076029	rapid
0.076029	innov
0.152057	improvement
0.076029	emerg
0.076029	highes
0.152057	cost
0.076029	bus
0.228077	microprocessor

The clusters formed with respect to the sample service category files such as 'weather' and 'computer architecture' using K-means clustering algorithm with k=2 from content word recognition step are shown in Fig. 3.

The NGD and Similarity values obtained from feature integration process are tabulated in Table. 2.

The performance of our approach using recall and precision was compared with [1]'s approach and the results are tabulated in Table 3 and Table 4.

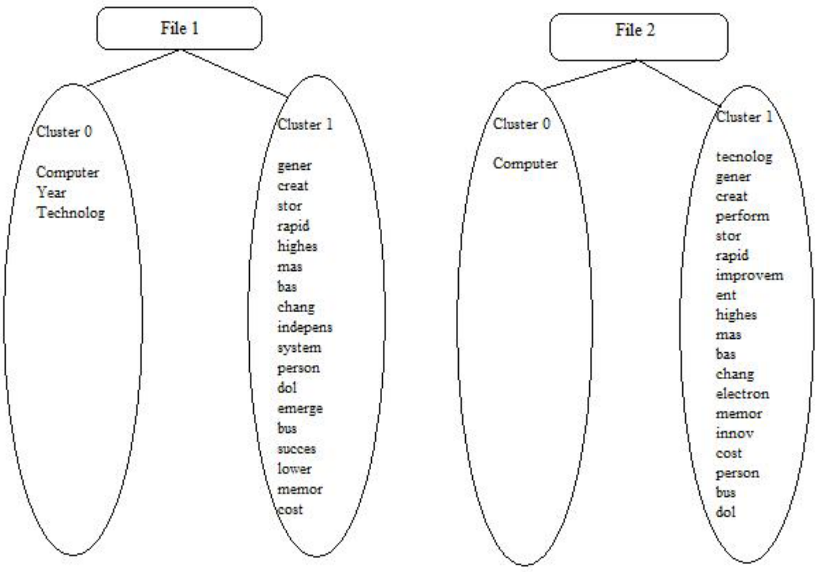

Fig. 3. Clusters generated from Content word recognition step by K-means clustering

Table 2. NGD and Similarity values generated by feature integration process

S.No	Word1	Word2	NGD	Similarity
1	computer	computer	0	1
2	computer	weather	0.52845	0.47154
3	computer	earth	0.39224	0.60775
4	computer	system	0.71737	0.28262
5	technology	computer	0.42513	0.57486
6	technology	weather	0.22959	0.77040
7	technology	earth	0.24365	0.75364
8	technology	system	0.26721	0.73278
9	weather	weather	0	1
10	weather	earth	0.34830	0.65169
11	computers	computer-architecture	0.21067	0.78932
12	computers	earth-weather	0.12563	0.87436

Table 3. Performance measurement of our approach

Cluster	Precision	Recall
Computer Architecture	100%	75.2%
Weather	75.1%	100%

Table 4. Performance measurement of [1]'s approach

Cluster	Precision	Recall
Computer Architecture	80.2%	100%
Weather	94.1%	100%

8 Conclusion and Future Work

Our approach provides better performance in experimental results in terms of time complexity as we consider only the relevant words for word stemming step during the extraction of first feature, when compared to [1]'s approach. Our proposed approach can be used as a prior step into search engines for efficient web service discovery based on the relevant user request. In future, we decided to extend this work, by including additional features such as context to support context-aware pervasive web services and cloud based web services.

References

1. Elgazzar, K., Hassan, A.E., Martin, P.: Clustering WSDL Documents to Bootstrap the Discovery of Web Services. In: 2010 IEEE International Conference on Web Services, vol. 1, pp. 287–294 (2010)
2. Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K., Martin, D.: OWL-S: Semantic Markup for Web Services. W3C Member Submission (2004)
3. Lausen, H., Polleres, A.: Web Service Modeling Ontology (WSMO). In: W3C Member Submission (2005)
4. Deng, S., Wu, Z., Wu, J., Li, Y., Yin, J.: An Efficient Service Discovery Method and its Application. International Journal of Web Services Research 6(4), 94–117 (2009)
5. Nayak, R.: Data mining in Web services discovery and monitoring. International Journal of Web Services Research 5(1), 63–81 (2008)
6. Liu, W., Wong, W.: Web service clustering using text mining techniques. International Journal of Agent Oriented Software Engineering 3(1), 6–26 (2009)
7. Coyle, F.P.: XML, Web Services and the Data Revolution. Pearson Education, South Asia (2002)
8. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Englewood Cliffs (1988)
9. Cilibrasi, R.L., Vitnyi, P.M.B.: The Google similarity distance. IEEE Transactions on Knowledge and Data Engineering 19(3), 370–383 (2007)
10. Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R.: Performance measures for information extraction. In: DARPA Broadcast News Workshop, Herdon VA (February 1999)
11. Porter, M.F.: An Algorithm for Suffix Stripping. Program 14(3), 130–137 (1980)