Chapter 92
A Web Service Discovery Method Based on Data Segmentation and WordNet

Tingna Liu and Ling Jiang

Abstract The Web service infrastructure has been an important software component on the Internet. A critical step in Web service applications is the service discovery method. The structured information in the web services description language (WSDL) of a Web service is unsuitable when using the traditional information retrieval (IR) method. We propose a Web service discovery method that is based on extracting information from WSDL documents. The method employs data segmentation to analyze the information to a meaningful structure where the traditional IR techniques can be applied. Furthermore, WordNet has been exploited in this method to carry out a synonym search. The experiments show that our method represents a significant improvement in the performance of Web service discovery.

Keywords Web service discovery • Search engine • Lucene • WordNet

92.1 Introduction

A Web service is a self-containing and self-describing modular application that can be published, located, and invoked across the Web. The effective discovery technology in a Web service is of vital importance and is a critical issue in the development of Web services. Existing research in Web service discovery focuses on three main fields: text-based matching [1], semantic-based matching [2], and similarity cluster matching [3].

This paper presents a Web service discovery method that makes use of both extracting information from the web services description language (WSDL) by the web services description language for Java toolkit (WSDL4J) and establishing indexes by Lucene. To improve the accuracy of Web service searches, we employ a kind of data segment to parse the structured information of WSDL into human

T. Liu • L. Jiang (✉)
School of Resources and Environment, University of Electronic Science and Technology of China, 611731 Chengdu, China
e-mail: jiangl_sre@uestc.edu.cn

© Springer International Publishing Switzerland 2015

W.E. Wong (ed.), *Proceedings of the 4th International Conference on Computer Engineering and Networks*, Lecture Notes in Electrical Engineering 355,
DOI 10.1007/978-3-319-11104-9_92

natural language. In addition to segment processing, WordNet is also used to achieve a functional similarity to Web service discovery.

The paper is organized as follows. In Sect. 92.2, we present a method to extract information from Web services based on WSDL documents and build indexes by Lucene. In Sect. 92.2.4, we discuss how to incorporate the WordNet synonym thesaurus into Web service discovery. In Sect. 92.3, we describe a prototype system of Web service discovery and discuss the experimental results. The last section provides a conclusion.

92.2 Building Indexes for Web Services

92.2.1 Extracting Information from Web Services

Currently, the majority of Web services are described by WSDL1.1, which can be parsed by WSDL4J [4], which allows for the creation, representation, and manipulation of WSDL documents. In this paper, we extract the service address, service name, service operations, and I/O parameters from WSDL documents using WSDL4J.

92.2.2 Segmentation Processing

When using a Web services search engine, a service requester submits meaningful words, for example the word "weather," as a keyword. WSDL is an XML-based language that is processed by a computer, in which most of the information is not described in a natural language. For example, take a service operation name—getWeatherForecast()—as an example; the service structured information is almost described as a compound word form in WSDL documents. Traditional information retrieval techniques will treat *getWeatherForecast* as a unit, leading to mismatch errors with the keyword "weather" in a Web service search. Apparently, some useful Web services could not be found because of this limitation.

In analyzing WSDL documents, we find that most Web service names accord with CamelCase notation. The CamelCase-format names of Web services can be separated easily into meaningful words, which is helpful to services requester for discovering potential Web services. Let us use again the Web service operation name *getWeatherForecast*() as an example. The CamelCase word *getWeatherForecast* will be segmented into three words—*get*, *weather*, and *forecast*. It seems like we are violating the Web service operation structure, but it actually makes sense in practice [5].

The use of segmentation processing may lead to a lower recall rate. Take the previous example of *getWeatherForecast*(); it is likely to result in redundant search

Table 92.1 Fields in an index document

Field	Store	Analysis
serviceName	Yes	Yes
operationName	Yes	Yes
ioType	Yes	Yes
Documentation	Yes	Yes
url	Yes	No

results because of the separation of weather and the forecast, Web services such as *financial forecast* or *sales forecast* may also be discovered in addition to *weather forecast* when *forecast* is submitted as a keyword. Selective keywords are needed to find a desired Web service in this case.

92.2.3 Building Indexes

Apache Lucene is an open source information retrieval software library, originally created in Java by Doug Cutting [6]. In this paper we build indexes for Web service information by Lucene. As shown in Table 92.1, five fields are created in an index document.

Lucene has a very complicated scoring mechanism. Lucene's conceptual scoring formula is as follows [6]:

$$\begin{aligned}\text{score}(q,d) = \text{coord}(q,d) \times \text{queryNorm}(q) \\ \times \sum_{t \text{ in } q} \left(tf(t \text{ in } d) \times idf(t)^2 \times t.\text{getBoost}() \times \text{norm}(t,d) \right).\end{aligned} \tag{92.1}$$

The factor coord(q, d), which depends on how many of the query terms are found in a specified document, is affected by the segmentation processing we used in the proposed method. norm(t, d) encapsulates a few (indexing time) boost and length factors as follows:

$$\begin{aligned}\text{norm}(t,d) = d.\text{getBoost}() \times \text{lengthNorm}(\text{field}) \\ \times \prod_{\text{field } f \text{ in } d} f.\text{getBoost}()\end{aligned} . \tag{92.2}$$

The lengthNorm(f) factor is of vital importance for our experiments; its formula is as follows:

$$\text{lengthNorm}(f) = \frac{1}{\sqrt{\text{num of terms in field } f}}. \tag{92.3}$$

The service name reflects the Web service's function to some extent, which

should thus be given much more weight while matching Web service with keywords given by users. A relatively short service name usually contributes less to the score based on the Lucene scoring mechanism. The service operation is the same as the service name. We increase the service and operation names' weights as appropriate to highlight their importance when building an index document for Web service information. The default weights are 1.0 in Lucene, and we increased the weights of the fields *serviceName* and *operationName* by 2.0 and 1.5, respectively.

92.2.4 Introducing WordNet

As discussed in Sect. 92.2.2, the desired Web service is probably missed if a keyword mismatches with WSDL information in a Web services discovery. A refined keyword might be a solution in that case, whereas the Web service discovery has lost its universal significance. Moreover, the same semantic service could be described using synonymous words [7], creating more difficulties in syntactic rule-based Web service matching. For example, say a service name contains the word *car*, which can be matched to a keyword, but the service name contains the word *vehicle*, which cannot be matched with the same keyword [8].

WordNet is an English dictionary developed by researchers at Princeton University that differs from a regular dictionary in that it contains semantic information [9, 10]. In recent years, many researchers have incorporated WordNet into the discovery of Web services and obtained some notable results [11]. We can effectively solve the aforementioned problems, which caused by using synonymous words in Web service descriptions, by incorporating WordNet's synsets of synonyms into Web service discovery. There are 203,147 synonym records in the WordNet2.0 ws_n.pl file. We use Syns2Index.java to build a Lucene index of synonyms in order to improve search speed [12]. The search accuracy is improved in two ways with respect to retrieval processing—one for synonym retrieval and another for Web service matching.

92.3 Prototype System and Results Analysis

Data used in this paper are from Dr. Yilei Zhang's research results [13], including 3,738 WSDL files [14]. We use the JAVA language to extract Web service features in addition to using WSDL4J and Princeton WordNet.

To verify the performance of the method proposed in this paper, in our experiments, five words that are commonly used in Web services are selected as keywords: *email*, *weather*, *map*, *news*, *fund*.

Table 92.2 Four experiments designed based on whether segmentation or WordNet is used

	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Segmentation	Yes	Yes	No	No
WordNet	Yes	No	Yes	No

92.3.1 Recall

The recall ratio of a Web search is reflected by the total number of search results (we use *hits* instead). We perform four groups of experiments based on whether word segmentation processing or WordNet synsets synonyms are used, as shown in Table 92.2.

The results are shown in Fig. 92.1a–c, where the vertical axis represents the total hits of the search results and the horizontal axis represents the five keywords we selected.

A comparison of the results of Experiments 1 and 2 is shown in Fig. 92.1a, from which we see the total hits of the keywords *map*, *news*, and *fund*, which increased significantly under WordNet. A comparison of Experiments 1 and 3 is shown in Fig. 92.1b, which illustrates that the total hits of all five keywords significantly increased under segmentation processing. A comparison of Experiments 1 and 4 is shown in Fig. 92.1c, which demonstrates a huge difference between incorporating or not incorporating segmentation processing and WordNet into Web service discovery.

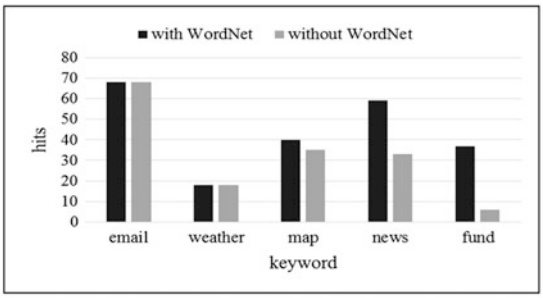
Note that the hits of the keywords *email* and *weather* did not increase even when using WordNet. Actually, WordNet contains a limited collection of synonyms. To increase the accuracy and recall ratio, it is necessary to establish professional WordNet synonym libraries.

92.3.2 Precision

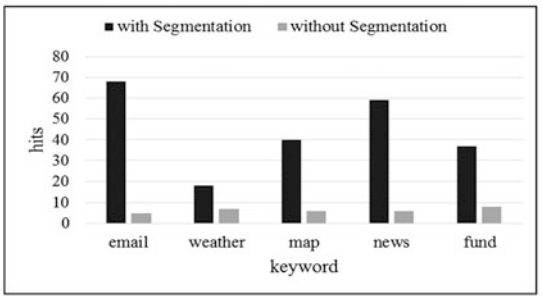
Precision is another feature of Web service discovery. In this subsection, we perform four sets of experiments depending on whether segmentation processing or WordNet is used, just as in Sect. 92.3.1.

We choose the same five keywords as previously. The experimental results are shown in Fig. 92.2a–d, in which the vertical axis represents the precision reflected by scores of the results and the horizontal axis represents the top five results that best match the keywords.

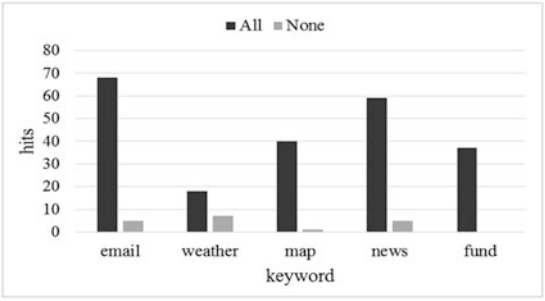
A comparison of Fig. 92.2b, d shows that segmentation processing brings significantly improved performance with respect to the top five matches. A comparison of Fig. 92.2a, b shows that the scores from matching the keywords *news* and *map* decline with the use of WordNet synsets synonyms. This is because WordNet synsets synonyms turn keywords into a type of synonym set. The accuracy of the



(a) The comparison of hits in results with and without WordNet



(b) The comparison of hits in results with and without segmentation

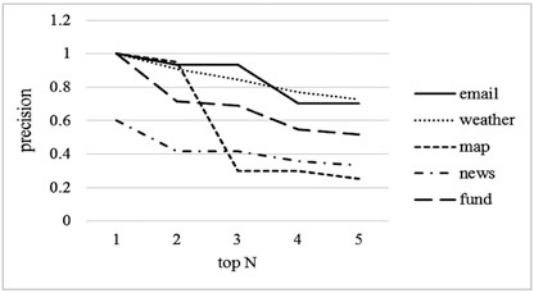


(c) The comparison of hitsin results with and without both WordNet and segmentation

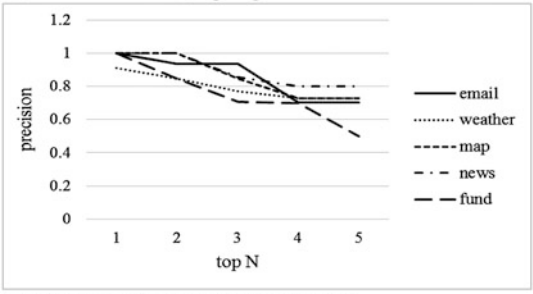
Fig. 92.1 (**a**) Comparison of hits in results with and without WordNet. (**b**) Comparison of hits in results with and without segmentation. (**c**) Comparison of hits in results with and without both WordNet and segmentation

results may be decreased to some extent by using WordNet, but it is worth doing because of the significant increase in the recall ratio [8].

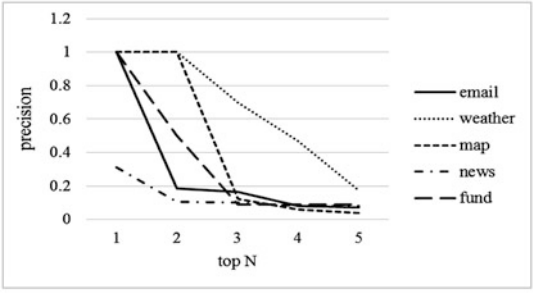
In theory, we know that the score will decline when document d matches few keywords based on the scoring formula factor coord(q, d). Only one expression of the synonym set will be used in a Web service WSDL document, which means the score will decline when the synonym set is used for keywords. However, the score of search results such as *weather* does not decline, which indicates that what we did at the end of Sect. 92.2.3 has come into effect.



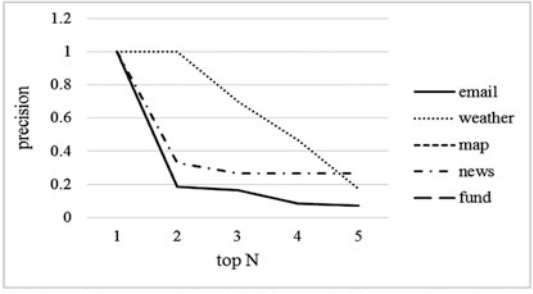
(a) Precision of using Segmentation and WordNet



(b) Precision of using Segmentation only



(c) Precision of using WordNet only



(d) Precision without using Segmentation nor WordNet

Fig. 92.2 (**a**) Precision using segmentation and WordNet. (**b**) Precision using segmentation only. (**c**) Precision using WordNet only. (**d**) Precision without using Segmentation or WordNet

Conclusion
By analyzing the WSDL, we built an index document of Web services using data segmentation processing and a WordNet synonym thesaurus in the discovery method. Retrieval was carried out twice in this method to realize discovery for Web services. Experimental results showed that the precision and recall ratio of Web service discovery improved to some extent. Meanwhile, there are also problems with the method, such as no guarantee of the validity of search results; additionally, a specialized thesaurus for WordNet synset synonyms should be created. In future work, we will devote attention to the quality of service and try to establish a professional WordNet thesaurus.

Acknowledgments The work conducted by Ling Jiang is partially supported by the National Natural Science Foundation of China (Grant 41201388), the Open Research Fund of the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, and the Fundamental Research Funds for the Central Universities (Grant ZYGX2012J154).

References

1. Stroulia E, Wang Y. Structural and semantic matching for assessing web-service similarity. Int J Coop Inf Syst. 2005;14(04):407–37.
2. Naveen Kumar S, Pabitha P, Mansoor Ahamed AK. Web service discovery based on semantic description. In: Proceedings of the IEEE International Conference on Cloud & Ubiquitous Computing & Emerging Technologies (CUBE 2013). Washington, DC: IEEE Computer Society; 2013. p. 199–203.
3. Mecella M, Pernici B, Craca P. Compatibility of e-services in a cooperative multi-platform environment [M]. In: Casati F, Georgakopoulos D, Shan MC, editors. Technologies for E-services. Berlin: Springer; 2001. p. 44–57.
4. WSDL4J [EB/OL]. http://wsdl4j.sourceforge.net/ (2014).
5. Liu F, Shi Y, Yu J, Wang T, Wu J. Measuring similarity of web services based on wsdl. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2010). Washington, DC: IEEE Computer Society; 2010. p. 155–62.
6. Lucene [EB/OL]. http://lucene.apache.org/ (2014).
7. Wang M, Chen R, Shi R. Web services discovery based on service description. In: Du W, editor. Informatics and management science III. London: Springer; 2013. p. 243–52.
8. Elgazzar K, Hassan AE, Martin P. Clustering wsdl documents to bootstrap the discovery of web services. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2010). Washington, DC: IEEE Computer Society; 2010. p. 147–54.
9. Miller GA. WordNet: a lexical database for English. Commun ACM. 1995;38(11):39–41.
10. Fellbaum C. WordNet: an electronic lexical database. Cambridge: MIT; 1998.
11. Birukou A, Blanzieri E, D'Andrea V, et al. Improving web service discovery with usage data. IEEE Softw. 2007;24(6):47–54.
12. Syns2Index [EB/OL]. http://www.chencer.com/techno/java/lucene/wordnet.html (2014).
13. Zhang Y, Zheng Z, Lyu MR. Wsexpress: a qos-aware search engine for web services. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2010). Washington, DC: IEEE Computer Society; 2010. p. 91–8.
14. WS-DREAM [EB/OL]. http://www.wsdream.net/dataset.html (2014).