An Automatic Service Classification Approach

Haiyan Zhao and Qingkui Chen

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai,200093, P.R. China
zhaohaiyan1992@sina.com.cn, chenqingk@sohu.com

Abstract. With the development of the service technology, more and more organizations are publishing their business function as service through Internet. Service classification is a key approach for service management. With the quick increase of service number, the cost of classifying these services through manual work is becoming more and more expensive. A service automatic classification approach based on WordNet by combining text mining, semantic technology and machine learning technology was given in the paper. The method only relies on text description of services so that it can classify different type services, such as WSDL Web Service, RESTfulWeb Service and traditional network based software component service. Though text mining and applying word sense disambiguation models, a service can be described as a sense vector with no ambiguous. Then a K-means algorithm is used to classify these services. Experimental evaluations show that our classification method has good precision and recall.

Keywords: Web Service, Text mining, WSD, Similarity Calculation.

1 Introduction

With the development of the service technology, more and more organizations are publishing their business function as service through Internet. Assigning a proper category to a service can be a tedious and error prone task due to the large number of services. Techniques for automatically classifying services will help to address this problem effortlessly. This paper describes an automatic classification approach based on text mining, machine learning techniques and semantic techniques.

During the past few years some efforts and research have been placed on assisting the developer to classify Web services. As a result, some semiautomatic and automatic methods have been proposed. These approaches are based on text mining and semantic annotations matching techniques. AWSC [1], METEROR-S [2] and Assam [3] are systems to classify web services based on text mining techniques. These systems generally consist of two parts, text mining module and web service classification module. Text mining module is responsible for extracting a feature vector from Web service descriptions. In this context, a feature vector is a term collection. Web service classification module is responsible for generating a feature vector for each classification

Y. Wang and T. Li (Eds.): Knowledge Engineering and Management, AISC 123, pp. 531–540.
springerlink.com

using machine learning techniques (such as Naive Bayes and Rocchio) and a training data-set. Then the similarity of each tested Web services and each classification is computed and the service is classified to the category with which it has the largest similarity. These methods have some limitations. The main limitation is that they need a group of classified Web services as the training data-set so that these methods are suitable for classifying newly added Web service after an initial classification framework has been set up. Secondly, most of these methods do not consider the semantic similarity and semantic conflict. For example, when a web service is descripted as <*advertising*> and another web service is descripted as <*campaign*>, these two services can probably be classified to different categories if without considering their semantic similarity.

An approach to add semantic annotations to Web services and classify these Web services by computing semantic similarity was proposed in [4]. The shortcoming of this approach is it relies on the domain ontology library. Although many domain experts are developing their domain ontology libraries, only ontology libraries of biology domain are widely accepted (such as TMO and Gene Ontology). Ontology libraries of other domains, such as finance, are far from complete. Therefore the classification approach based on semantic annotation and matching is difficult to implement.

In addition, current classifying methods mainly support web services classifying in terms of the WSDL documents. With the diversification of service types, more and more Web services adopt the RESTful style, such as Google AJAX Search API and Amazon Simple Storage Service (Amazon S3). Although currently the number of RESTful Web services is less than the number of WSDL Web services, this style of Web services has extensive application prospect. At the same time, traditional software components are also used in practical application development. But the research on how to classify RESTful web services and traditional software components is relatively less.

In order to overcome these limitations, a service automatic classification approach based on WordNet, which combines text mining, semantic technology and machine learning technology was given in the paper. The approach only relies on text description of services so that it can classify different type services, such as WSDL web service, RESTful web service and traditional software component service. This method does not need a training data set and is suitable for classifying large quantity services. This paper is organized as follows. Section 1 was an introduction. The framework of the system was given in Section 2. Section 3, 4, 5 and 6 discussed important steps of this approach respectively. Experiments and evaluations were given in Section 7. Section 8 concluded the whole paper.

2 The Framework for Service Automatic Classification

The framework of our system for service automatic classification consists of four parts, i.e., a text mining module, a word sense disambiguation module, an automatic web service classifier and a similarity calculation module.

The text mining module is used to extract interesting and trivial information and knowledge from description document of each service. Its outputs are a term vector.

Word semantic in the term vector is clarified using the word sense disambiguation module so that a matrix is obtained, where row represents each service and column represents a sense vector of the associated service. The automatic web service classifier based on K-Means algorithm is used to classify these services. The word sense disambiguation module and the automatic web service classifier both adopt a sense similarity algorithm based on WordNet to calculate the similarity of two senses.

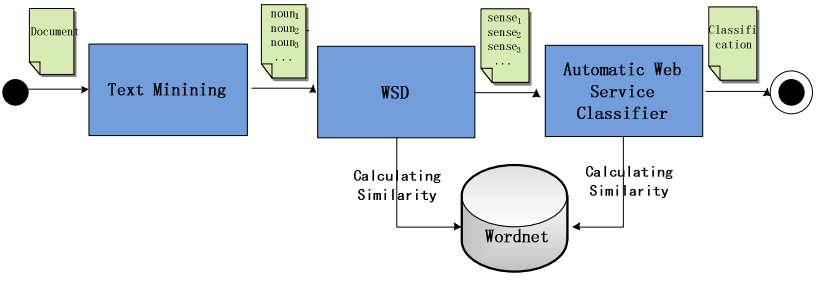

Fig. 1. System Framework for Service Automatic Classification

3 Text Mining

The text mining module is responsible for extracting interesting and trivial information from description document of each service. Its output is a term vector that is the input of the subsequent word sense disambiguation module. The quality of the output has direct impacts on the quality of service classification.

Current service classification methods extract trivial information from WSDL document of each service. WSDL is a well-structured and standard XML document format for describing Web services. Most methods rely on a parser based on WSDL4J, which can parses a WSDL document to extract out the service name, operations and arguments as the term vector of the service. This parser only deals with WSDL document and cannot process RESTful Web Service [5] and traditional software components. Our text mining module includes a more general parser which can extract trivial information from all kinds of description documents for each service by using text mining technology.

The traditional text mining approach often adopts TF/IDF algorithm. But in the description documents of a service the frequency of each term is very low. Therefore TF/IDF algorithm is no longer suitable. No matter what kind of services it is, it can be described as a tuple <*action, target, way*> formally, where an action is composed of verbs, target and way are composed of nouns. From this point of view, verbs and nouns are the trivial information of a service description. Our text mining module focuses on the verbs and nouns and ignores the else.

Fig.2 depicts the steps of text mining process and they are splitting word combinations, Pos tagging, reducing terms to their stems, removing noise and translating verbs to nouns.

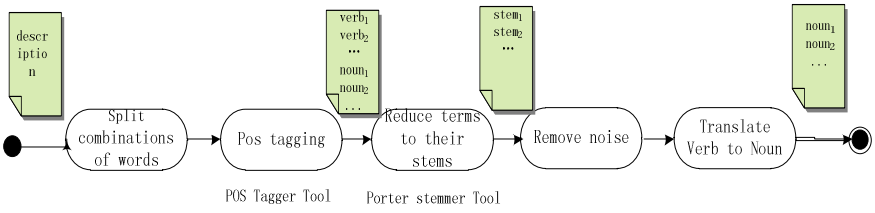

Fig. 2. Text Mining Process

(1)Splitting word combinations

In the textual description of each service, some words are often combined, such as GetWeather or CityName. Our tool searches word combinations and splits them into separate verbs and nouns according to naming conventions of software programming. After splitting word combinations, we can obtain a term vector for each service.

(2)Pos tagging

Verbs and nouns are the trivial information of a service. We will do pos tagging for each term of the term vector. We used Stanford POS Tagger [7] developed by the Stanford NLP (Natural Language Processing) Group to select verbs and nouns.

(3)Reducing terms to their stems

Considering the commoner morphological and inflectional ending, we also employed the Porter stemmer [6] to reduce words to their stems.

(4)Removing noise

Noise in service description will impact the quality of service classification. We removed noise from two aspects: one is common English stops words, the other is stops words which are related to services, such as request, response, soap, http, Result, platform, program, system, site, engine, database and so on.

(5)Translating verb to noun

Word sense disambiguation module and automatic service classifier adopt sense similarity algorithm based on WordNet to compute the similarity of two senses. These algorithms were given in [8][9][10]. The similarity of two senses will be zero when two senses are different parts of speech according to the algorithms. For example, although the first sense of *market* (as a verb) is similar to the second sense of *promotion*, the result of Lin's algorithm is zero because these two senses are different parts of speech. So we must translate all verbs to nouns such as *market* to *marketing* (as a noun). Then the similarity of *marketing* and *promotion* is 0.25 according to Lin's algorithm.

Table 1. Noun Vector for Service

Service Name	Noun Vector
Allegro	{auction marketplace}
AvantLink	{affiliate marketing}
123 Shop Pro	{shopping cart}
Alloga-rage	{marketplace car dealership review recommendation}
Cafe Press	{retail product}
…	…

We can obtain a noun vector for each service by text mining module. Table 1 depicts the noun vector for some services.

4 Word Sense Disambiguation

In WordNet, a word usually has multiple senses, for example, bus has four senses. When bus appears with station, its sense is automobile. But when bus appears with network, its sense is computer bus. Word sense disambiguation is used to find the correct meaning of a polysemous word in a given context.

As non-supervised WSD allows disambiguation of words without user interference, we use a variant of the SSI algorithm [11] to get the senses out of a set of words (Formula (1)). The algorithm disambiguates a word based on a previously disambiguated set of words and their related senses. For each sense of a word (s_j), a similarity with the senses from the context (sc_i) is calculated and the sense with the highest similarity is chosen. After that, the word and its chosen sense will be added to the context (I) and iteration will be done. This process continues until there are no ambiguous words left.

In Formula (1), senseSim (s_j, sc_i) is the similarity of two senses and it is shown in Formula (6).

$$selectedSense(word) = \arg \max_{s_j \in senses(word)} \sum_{sc_i \in I} senseSim(s_j, sc_i) \quad (1)$$

Table 1 will be changed into Table 2 after applying word sense disambiguation. For example, *auction#n#2*, *#n* means noun, *#2* means the 2th sense.

Table 2. Noun Sense Description of Service after Applying WSD

Service	Description	Sense Vector after WSD
Allegro	Online auction marketplace	{auction#n#2 marketplace#n#1}
AvantLink	Affiliate marketing network	{affiliate#n#2 marketing#n#2}
123 Shop Pro	Online shopping cart software	{shopping#n#1 cart#n#2}
Allogarage	marketplace car dealership reviews and recommendations	{marketplace#n#1car#n#1 dealership#n#1 review#n#1 recommendation#n#1}
Cafe Press	Customized retail product service	{retail#n#1 product#n#1}
…	…	…

5 Automatic Service Classifier

After applying word sense disambiguation, we can obtain a sense vector for each service that is unambiguous. The next step is to classify by applying improved K-means clustering algorithm.

K-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. This results into a partitioning of the data space into Voronoi cells.

In traditional K-means clustering algorithm, each observation is a d-dimensional real vector. But in our application, each observation is a d-dimensional sense vector

and we need resolve two problems. One is how to calculate the new centroid of the observations in some cluster. The other is how to calculate the distance of the each observation to the cluster.

For the first problem, calculating the new centroid of the observations in some cluster can be treated as a problem of calculating the centroid of a group of sense vector. Since a cluster can be described by high frequency senses in this cluster, we retreat a group sense vector of all services belonging to some cluster as an input. Then TF-IDF algorithm is used to select the first m frequent senses {$sense_1$, $sense_2$, …, $sense_m$} as the centroid of this cluster. The Fig.3 depicts the detail.

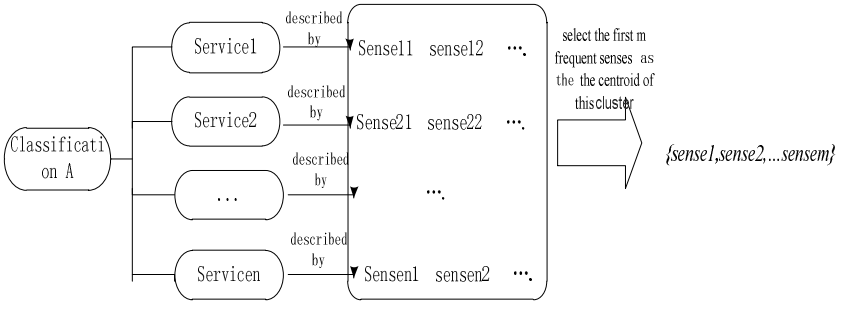

Fig. 3. Computing the Center of a Category

Formally, for each term t_i of a document d, $tf_{idfi}=tf_i*idf_i$, with:

$$tf_i = \frac{n_i}{\sum_{j=1}^{T_d} n_j} \tag{2}$$

Where the numerator (n_i) is the number of occurrences within d of the term being considered, and the denominator is the number of occurrences of all terms within d (T_d), and:

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|} \tag{3}$$

Where $|D|$ is the total number of documents in the corpus and $|\{d: t_i \in d\}|$is the number of documents where the term ti appears.

The first problem also involves another problem of determining initial centroids. By experiments, we find the clustering quality is often poor if we randomly choose k observations as the initial centroids. For example, if we cluster all services in shopping catalogue and advertising catalogue in ProgrammableWeb site. If these services are classified into 2 sets and the initial two centroids which are randomly choose are both in shopping catalogues, the clustering quality is poor. To overcome this problem, we choose first k frequent sense as initial centroids by TF/IDF algorithm (k is the number of cluster). Although further experiment showed the clustering quality was improved, it is still not satisfied. The root cause is some senses often appear simultaneously. For above-mentioned example, shopping often appears with cart. In fact, these two senses stand for the same centroid. But by TF/IDF algorithm, shopping and

cart should be treated as two initial centroids. In order to resolve this problem, the TF/IDF algorithm and association rules are combined in our approach. Firstly, TF/IDF algorithm is used to order each sense. Secondly association rules algorithm is applied to find senses sequence, which often appears simultaneously. Lastly, we choose first *k* frequent sense sequences as initial centroids

For the second problem, the distance of the each observation to the cluster can be calculated by computing the similarity of two sense vectors. Formula (4) shows how the similarity between sense vectors ssw and ss_u is computed. The average of similarity between any sense (s_u) of the sense vector (ss_u) and the sense vector (ss_w)is computed. The average of the similarity between any sense (s_w) of the sense vector (ss_w) and the sense vector (ss_u) is calculated. They are summed up to provide a symmetric match.

$$vectorVect\ \ orSim\ (ss_u, ss_w) = \sum_{s_u \in ss_u} \frac{senseVecto\ \ rSim\ (s_u, ss_w)}{|ss_u| + |ss_w|} + \sum_{s_w \in ssw} \frac{senseVecto\ \ rSim\ (s_w, ss_u)}{|ss_u| + |ss_w|} \tag{4}$$

In Formula (4), senseVectorSim(s_u, ss_w) or senseVectorSim (s_w, ss_u) means similarity of a sense and a sense vector. The similarity between a sense (s_a) and a sense vector (ss_b) is done by searching for the maximum similarity between the sense and each sense (s_b) of ss_b. Formula (5) shows how this is done.

$$senseVectorSim(s_a, ss_b) = \max_{s_b \in ss_b} \quad senseSim(s_a, s_b) \tag{5}$$

In Formula (5), senseSim (s_a, s_b) is the similarity of two senses and its formula is shown in Formula (6).

6 Similarity Calculation

Many similarity measures have been proposed. Lexical comparison measure and semantic similarity measure based on WordNet are widely used. Semantic similarity refers to similarity between two concepts in taxonomy such as WordNet. Our word sense disambiguation module and automatic web service classifier both consider semantic similarity so that it is adopted to calculate the similarity of two concepts.

We adopted Lin's measure and used an open source java libraries:WordNet::Similarity to implement it.

Lin's measure is shown in Formula (6).

$$senseSim(C_1, C_2) = \frac{2\log^{-1} P(C_0)}{\log^{-1} P(C_1) + \log^{-1} P(C_2)} \tag{6}$$

C_1 and C_2 are two concepts of the taxonomy tree. C_0 is the most specific class that subsumes both C_1 and C_2. *P(C)* is the probabilities that a randomly selected object belongs to C. In order to calculate *P(C),* there must be a training corpus to get the probability of each concept. Here Semcor corpus is adopted which is developed by Princeton University.

In our experiment, shopping catalogue and travel catalogue in ProgrammableWeb site are used as test data sets. In these two catalogues, occurrence frequencies of

shopping and travel are high. Let's consider the problem of determining the similarity between two concepts shopping and travel. Fig.4 depicts the fragment of the WordNet. The number attached to a node indicates its *IC(C)* (here, $IC(C)=\log^{-1}P(C)$). From the Fig. 4 we can find that action is the concept that subsumes both shopping and travel. The similarity of shopping and travel is calculated as following:

$$senseSim(Shopping, Travel) = \frac{2\times IC(Action)}{IC(Shopping)+IC(Travel)} = \frac{2\times 3.7}{9.23+6.52} = 0.47$$

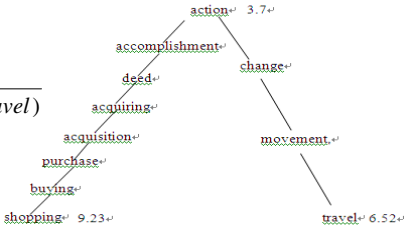

Fig. 4. A Fragment of WordNet

7 Experiment and Evaluation

We did the experiments on a test data set collected from Programmable Web Site. We extracted key information from service description documents and classified these services using our approach. The classification results were compared with the results from other approaches.

The precision and recall values are computed according to Formula (7), where n is the classification number of the testing data set, p_i is the precision of the *i-th* classification and r_i is the recall of the *i-th* classification.

$$p = \sum_{i=1}^{n} p_i \Big/ n \qquad r = \sum_{i=1}^{n} r_i \Big/ n \tag{7}$$

The *precision*(p_i) and *recall*(r_i) are calculated as follows: where m is the number of services which are classified to the *i*-th classification and n is the number of services which belong to the *i-th* classification in the test data set. If a service is classified correctly, then y_i is set 1, otherwise y_i is set 0.

$$p_i = \sum_{j=1}^{m} y_j \Big/ m \qquad r_i = \sum_{j=1}^{m} y_j \Big/ n \tag{8}$$

We did experiment six times and computed the precision and recall for each experiment. Fig.5 depicts the results.

From Fig.5, the average precision of our method is 74% and the average recall of our method is 71%. The result proves our method has satisfying quality of classification.

Comparing with Test 3, the precision and recall of Test 4 decrease greatly. In Test 4, we

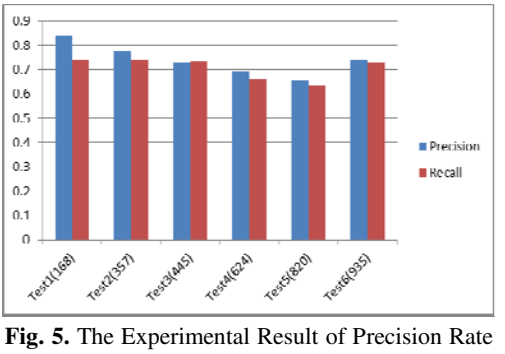

Fig. 5. The Experimental Result of Precision Rate and Recall Rate

added 179 APIs from shopping catalogue and advertising catalogue into the test data set. There are two reasons that lead to the quality decrease. Firstly, we find APIs from shopping catalogue and APIs from advertising catalogue are very similar. For example, their description documents both include *marketing* term and *advertising* term. Secondly, in Lin's method, the similarity is calculated in terms of the probability of each concept. In order to calculate the probability, we adopted Semcor corpus for training. Though Semcor corpus is the largest English corpus, in order to construct a semantic system with high accuracy, it is still necessary to expand this corpus. For example, in our experiment, some APIs are about retailer and should be classified to shopping catalogue. But by Lin's method, the similarity of (*retailer, shopping*) is 0.06934 and the similarity of (retailer, advertising) is 0.1192. Therefore these APIs are classified to advertising catalogue. In the subsequent research, we are going to expand Semcor corpus by adding the vocabularies coming from Programmable Web site.

Comparing with test 4, the precision and recall of test 5 decrease greatly. In test5, we added 196 APIs from travel catalogue and financial catalogue. The APIs from these two catalogues are very similar, which make the quality decrease.

Moreover, we found the imbalance of APIs number of different catalogue also had impact on classification accuracy. For example, in test 5, the APIs number of sports catalogue and weather catalogue is less than other catalogues'. Therefore in test 6, we replaced these APIs from sports catalogue and weather catalogue by the APIs from messaging catalogue and telephony catalogue whose numbers of APIs are proportional to other tested catalogues. Hence, the precision and recall of test 6 was improved greatly.

We compared our method with K-NN, Naive Bayes and Rocchiod method based on TF-IDF. The results are shown in Table 3. It shown that our method achieved better results than K-NN. But our method achieves worse results than Naive Bayes and Rocchiod method. The main reason was Naive Bayes and Rocchiod method based on TF-IDF were supervised machine learning algorithm, which need a training data set. Our method adopted K-means and it was a n unsupervised machine learning algorithm which don't need a training data set. We believe combing the supervised machine learning algorithm and unsupervised machine learning algorithm can achieve better result: firstly, using our method to classify the original set of services. Then using supervised machine learning algorithm to classify newly added services after having an initial classification framework.

Table 3. Comparison between Different Classifiers

Classifer	Average Precision
K-NN	39.59%
Naive Bayes	79.38%
Rocchiod method based on TF-IDF	85.08%
Our Approach	74%

8 Conclusions

A service automatic classification approach based on WordNet by combining text mining, semantic technology and machine learning technology was given in the paper. The method only relies on textual description of services so that it can classify

different type services, such as WSDL web Service, RESTful web Service and traditional network based software component service. Through applying text mining and word sense disambiguation models, a service can be described as a sense vector with no ambiguous. Then a K-means algorithm is used to classify these services. Experimental evaluations show that our classification method has good precision and recall.

Our automatic classification apporach can be extended in several ways. For example, it could be extended with the ability of hierarchy clustering. Furthermore, distributed computing framework, such as Hadoop, can be applied to speed up the classification.

Acknowledgments. This work was supported by China NSF under Grant No. 60970012, 60873230 and 61073021, Shanghai Key Science and Technology Project in Information Technology Field (No.09511501000), Shanghai Key Science and Technology Project (No.09220502800) and Shanghai leading academic discipline project No.S30501.

References

1. Clifton, C., Leavens, G.T., Chambers, C., Millstein, T.: MultiJava: modular open classes and symmetric multiple dispatch for Java. ACM SIGPLAN Notices 35(10), 130–145 (2000)
2. Wegner, P., Zdonik, S.: Inheritance as an Incremental Modification Mechanism or What Like is and Isn't Like. In: Gjessing, S., Chepoi, V. (eds.) ECOOP 1988. LNCS, vol. 322, pp. 55–77. Springer, Heidelberg (1988)
3. Waxman, B.: Routing of multipoint connections. IEEE Journal on Selected Areas in Communications 6(9), 1617–1622 (1988)
4. Yonezawa, A.: ABCL: An Object-Oriented Concurrent System. MIT Press, Cambridge (1990)
5. Matsuoka, S., Yonezawa, A.: Analysis of inheritance anomaly in object-oriented concurrent programming languages. In: Agha, G., Wegner, P., Yonezawa, A. (eds.) Research Directions in Concurrent Object-Oriented Programming, pp. 107–150. MIT Press, Cambridge (1993)
6. Hemige, V.: Object-Oriented design of the groupware layer for the ecosystem information system. University of Montana (1995)
7. Rose, A., Perez, M., Clements, P.: Modechart toolset user's guide[R]. Technical Report, NML/MRL/5540-94-7427, University of Texas at Austin, Austin (1994)
8. Keene, S.: A Programmer's Guide to Object-Oriented Programming in Common LISP. Addison-Wesley Longman Publishing Co., Inc., Boston (1988)
9. Guo, L., Tang, Z.: Specification and verification of the triple-modular redundancy fault-tolerant system. Journal of Software 14(1), 28–35 (2003)
10. Schutze, H.: Dimensions of meaning. In: Whitelock, P. (ed.) Proc. of the Supercomputing, Los Alamitos, vol. 796, pp. 787–796 (1992), `ftp://parcftp.parc.xerox.com/pub/qca/papers/`
11. Sangers, J.: A Linguistic Approach for Semantic Web Service Discovery, Bachelor Thesis, Economics and Informatics.Erasmus University Rotterdam (July 2009)