
Web Service Classification

Jing Zhang

Department of Computer Science
UC, Irvine
jingz@uci.edu

Dan Pan

Department of Computer Science
UC, Irvine
pand@ics.uci.edu

Abstract

Nowadays Applications based on web services as interaction interface/application programming interface are becoming more and more popular. One of its merits is that it's platform-independent and implementation-independent, which makes it possible to build new web services automatically just by composing existing ones to meet some coming requirements. To ease this work, it is important to classify the available web services into several categories for next step.

1 Introduction

With the development of Web Services, more and more researchers pay attention to this field. Semantic Web Service is an important part of web services research. The most popular way to add semantic to Web Services is as follow, first define terminology by using Ontology, and then add semantic meaning to Web Services by using the terminologies. As we known, terminologies are defined by domain experts and work in a certain domain. So it is necessary to classify Web Services into certain domains.

Besides semantic Web Services, Web Services composition is another research topic in Web Services research. Different from other software, web Services are convenient to compose because of the uniform soap transmission and interface defined. A practical way to do Web Services composition is to match the data type of input and output [1]. However, Web Services with same data type of input and output might have different functions. This situation always happens when the data type of input and output are simple data type. For example, Web Service A is for getting holidays during a period, its input is beginning time and end time of the period. Its output is a list of data. Web service B is for getting date of job affair during certain period, its input is also two date and its output is a list of date. We notice that these two Web Services do not work in the same domain, so classifying web services first and then composing Web Services in certain domain will reduce this kind of miss matching in Web Services composition by matching data type.

Every Web Service has a WSDL file, Web Services Description Language, which definition and description information of this Web Service, like Web Service name, Operations and input/output. There is some research work which focuses on web services classification based on WSDL file. A. Hess and N. Kushmerick use classification to classify web services [2]; they simply map WSDL file to text and use basic text mining to classify Web Services. The group at the University of Washington implements a Web Services searching Engine, which named Woogle [3]. They cluster to classify Web Services that he collected by matching data type of input and output.

In this project, we also focus on Web Services classification. The main difference from others' work is that we pay importance on the structure of WSDL file. Moreover, we introduce several traditional text mining algorithms like tf-idf and try different kernels like polynomial kernel, RBF kernel and string kernel and compare the corresponding classification results. The report is organized as follows. In section 2, we introduce the framework of our project and analyze the problem. Section 3 is about the design and implementation of our project and section 4 about the experimental result and comparison among them. In section 5, we make a conclusion over our work and talk about future work.

2 WSDL Mining Framework

2.1 Problem Analysis

First, let's have a general look into the traditional data mining, text mining and web mining. Compared to the data mining, text mining is more focus on extracting information that is useful to particular purpose, no matter explicitly expressed in the context or not. Web mining takes more characteristics of web structures into analysis, e.g., the contents of other web pages linked to the current one.

Now the self-describing XML file, Extensible Markup Language, offers more straight forward information. WSDL is based on the semi-structured XML and follows the W3C standard, say, it must contain a three-layered structure of web service, operation, input/output argument list, which easier the analysis work. In fact, our basic idea is based on elements contained in the three-layered architecture.

2.2 System Architecture

We build a web service classification system. Every time a new web service comes, we extract its corresponding WSDL file (UDDI file, Universal Description, Discovery and Integration, is ignored in the current work) and parse it into a collection of tokens, the tokenized key words in the description of the web service/operations, the name of the operations and the argument list including both the input and output variable names. Then we map all of them to the feature vectors, which depends on different algorithms and use the classifier to make the decision which group it possibly belongs to and store the result for the future use.

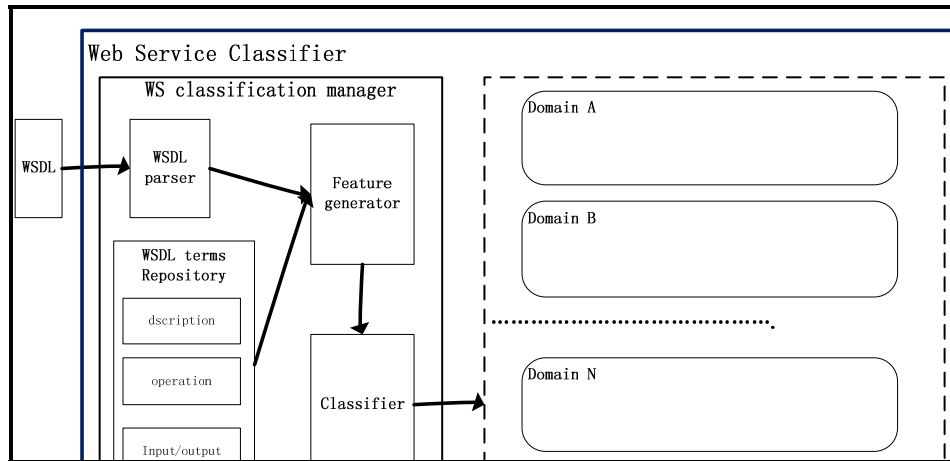


Figure System Architecture

3 Design and Implementation of WSDL Classification

3.1 Algorithm

There is some work on universal XML classification. They usually map the tree-like structure according to the (self-describing) tags in the XML file to the feature vectors. For WSDL, it's easier. The W3C standard imposes a little strict structure architecture. In fact, we could treat a web service as a list of APIs. Then WSDL is the file of the available API list. The textual description on the web services and each operation, the names of each operations and the argument list of each operations are the fundamental elements for every WSDL file for a published web service online. They maintain the basic and most important information about the web service.

Therefore, we build the feature vectors on the three pools of tokenized words. For the description part, we use traditional text classification techniques. For operation name and argument list, we just use the statistical results after they get tokenized since they would never be too complicated. The composed data type is not unfolded in the current work.

3.1.1 Feature attribute value algorithm

Currently we make use of three statistical methods, that is, Binary, Times and tf-idf.

We count and record the key words appearing the training sets and remove the trivial words (e.g., in, of) and tags (<p>,
) and then make it as the space of feature vector.

Binary: if one key word appears in one coming sample, then mark the corresponding bit of its feature vector as 1, else 0.

Times: similar with Binary, but mark the number of the times the words appear in one sample. It's always an integer.

tf-idf: use the tf-idf result instead of the straightforward counting.

As we have mentioned, the three feature spaces are,

Web service description: A web service is described by a name, a textual description in the WSDL.

Operation description: Each operation is described by a name and a textual description.

Input/Output description: It's an argument list. Each input and output of an operation contains a set of arguments. For each parameter, the WSDL describes the name, data type and arity (if the parameter is of array type). Parameters may be organized in a hierarchy by using complex types.

The feature vectors are built respectively for three kinds of features, description, operations and their argument list.

As comparison, we build another set that combines the three feature spaces.

3.2 Implementation

3.2.1 Training set collection

We collect the training data from Internet. They are all real-world web service application published online. Most of the web services are found through a web site of web service publication service, <http://www.xmethods.net>. We implement spider to extract the information about the web services published on it and then download the corresponding WSDL files. The size of the data set is of around 500 entries.

The spider is implemented in C#.

3.2.2 WSDL parsing and feature vector generating

The WSDL parser is implemented in C# with the XML Document support from Microsoft .Net Framework. Hash table is used to improve the efficiency of the statistical computing.

Only the targeted parts are stored in the output file for the classifier and other information is discarded.

The statistical results are the values for the element values for the vector generating. The mapping algorithm is as discussed above.

3.2.3 Classification and verification

We use the Weka 3.4.12 (<http://www.cs.waikato.ac.nz/ml/weka/>) for as the classification tool. Weka is a very powerful and convenient free tool for machine learning experiments and offers open interaction port format to introduce new algorithm as plug-in.

All the work result above is formatted to meet the requirement of the input for Weka.

Several classifiers are used and compared. They are SMO, Naive Bayes, Bagging/DecisionDump, AdaBosstM1/DecisionDump, Random Forest, etc. over the differently preprocessed training data. The result is available in the next chapter.

Cross-validation is used to estimate the accuracy of the algorithms.

4 Experiments

4.1 Experimental results

As we have said in Section 3.2.1, we get about 500 WSDL files for web services and classify these web services into 8 categories manually according to the description on the web pages. The training set consists of Business, Computing, Documenting, Financial, Geographical, Network, News, and Telephone. During extracting WSDL files, we fail to download some entries after timeout. In addition, we abandon some which is even too difficult to classify manually.

We have carried out 2 ways to generate features. The first method is mapping the 3 feature spaces into one. The second one is mapping the 3 feature spaces respectively and takes the three results into consideration for the final classification.

	adaBoostMI /DecisionStump	Bagging /DecisionStump	Naive Bayes	RandomForest	SMO /RBF	SMO /Poly
Binary	38.3%	38.3%	31.5%	36.1%	38.3%	37.2%
Times	38.3%	38.3%	26.7%	35.2%	38.3%	35.5%
tf-idf	39.1%	41%	28.6%	37.9%	39.9%	36%

Table 1-Vector

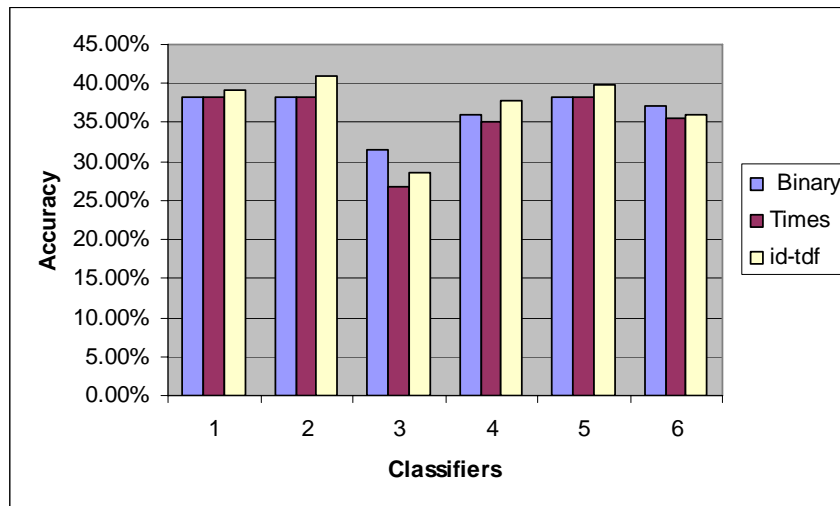


Figure 1-Vector

In Table 1-Vector and Figure 1-Vector, it appears that there's little difference between Binary and Times, while tf-idf always shows some advantage over the other two.

	adaBoostM1 /DecisionStump	Bagging /DecisionStump	Naive Bayes	RandomForest	SMO /RBF	SMO /Poly
Binary	36.5%	42%	30%	40.4%	41.6%	33.2%
Times	37.4%	42.4%	26%	44.2%	37.8%	34.2%
Tf-idf	39.4%	40.5%	29.6%	40.9%	40.5%	39%

Table 3-Vector

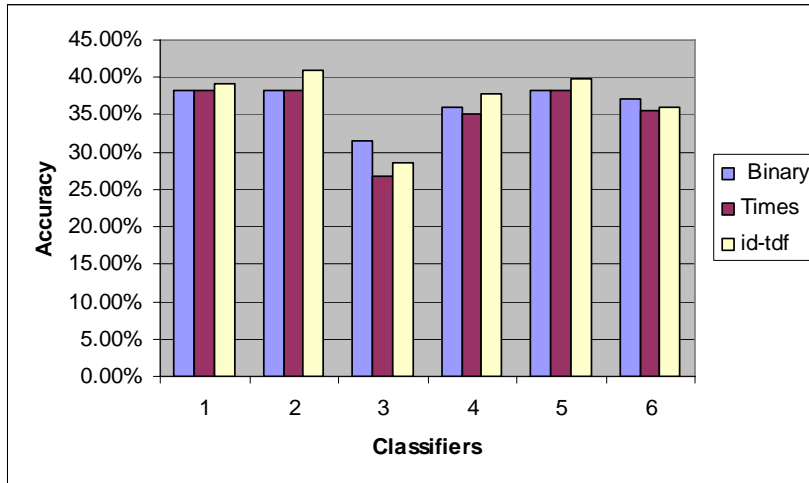
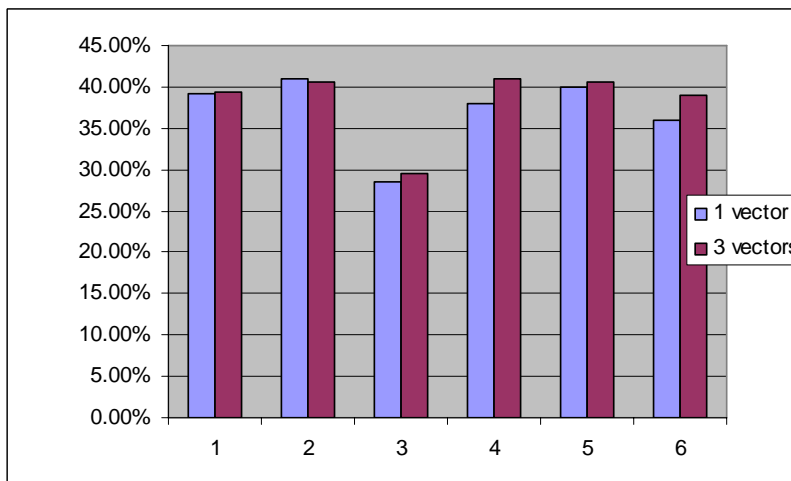


Figure 3-Vector

In Table 1-Vector and Figure 1-Vector, again, it appears that there's little difference between Binary and Times, while tf-idf always shows some advantage over the other two. Now it's safe to conclude that tf-idf represents the information best.



1-Vector vs 3-Vecotr with td-idf

The one with 3-Vector feature spaces outperforms than the other by a little. It proves that the structure attributes in WSDL matters.

4.2 Analysis on the results

The experimental result seems a little disappoint. The accuracy is always between 30% and 40%. Compared to the better results reported by some research groups, we guess the main difference is that they impose the complex semantic analysis on the description of the web services and operations, while we just use plain statistic method. Some group adds extra metadata to the web service/WSDL published to ease the clustering and classification.

What deserves mentioned is that even other experiments' results are not that satisfying compared to the traditional textual classification. Our argument is that it a instinct problem of web service. The web service tends to contain much less info compared to a paragraph for textual classification. We check some WSDL file manually and find it might only contain less then 10 useful key words, which is really bad fact for classification; in that case, classification is little better than random selector.

In addition, we are afraid the manually classification on the training data is hard. It's very common for human that it's really difficult to decide which group one web service should belong to. It's about sample error.

5 Conclusion and Future work

Data mining on web service (e.g., web service classification) is a very promising area for researchers given explosive use of XML Web Service. It's distinguishing from the traditional textual mining (classification), in that its structured nature matters.

The ideal web service classification, based on XML mining, should be a collective consequence of a variety of efforts including not only the data mining, text mining, but also the recent semantic web mining.

Our current work is a plain approach to classify web services with relative straightforward machine learning methods. The experimental result is not very satisfying. Next step is to look for possible better statistical method and more fit classifiers (with kernels).

We did some survey in XML mining kernel. We find some interesting kernels like String kernel and tree kernel which is suppose to improve the text classification accuracy significantly. However, these kernels can not work on numeric features, so we did not implement it in our project. In the future, we can try a string feature with string kernel. Also, we learn Yan from Stanford has computed distance between two words by using Webster dictionary as training set [8]. By using Yan's work, we can build a kernel consider semantic meaning. Unfortunately, Yan's work is not available on web site and he did not reply our mail.

The long term goal is to build up an automatic composing system, which, given a request, would look for desirable existing web services in the repository to build a new web service based on the accurate classification information. For example, some one needs a web service that will list the stock prices of local companies given a zip code; the system will build one by composing a web service on stock price in category Financial and one on addresses of companies in category Geography.

Acknowledgments

Special thanks to our dear Prof. Deva Ramanan and the machine learning group at UC Irvine.

References

- [1] A. Hess and N. Kushmerick, Learning to attach semantic metadata to web services. In ISWC, 2003
- [2] Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes, Jun Zhang, Similarity Search for Web Services. In Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004
- [3] Tao Y., Quality of Service (QoS) in Web services: Model, Architecture and Algorithms, Ph.D. dissertation, 2006
- [4] YU, F., Text Classification Based on a Combination of Ontology with Statistical Method. Machine Learning and Cybernetics, 2006 International
- [5] LODHI, H., Text classification using string kernels. The Journal of Machine Learning Research, 2002
- [6] Swathy Giri, XML Classification, Master Thesis, Computer Science and Engineering, Madras University, Chennai, India, 2002
- [7] JEONG, B., Towards XML Mining: The Role of Kernel Methods. iisl.postech.ac.kr
- [8] JANNINK, J.F., A Word Nexus for Systematic Interoperation of Semantically Heterogeneous Data Sources, 2001