Use of Classification Algorithms for Semantic Web Services Discovery

Martha Varguez-Moo, Francisco Moo-Mena and Victor Uc-Cetina
Facultad de Matematicas, Universidad Autonoma de Yucatan, Merida, Mexico
Email: msvarmoo@gmail.com, {mmena, uccetina}@uady.mx

Abstract—**Web services (WS) are used in different environments like enterprises, government and industry, providing tools for implementing complex distributed systems. Web service discovery allows a system to find services that meet the requirements of the users. One way to improve this type of discovery would consider not only the functional aspects of the required service, but also relevant aspects such as performance or availability to perform its functions. Moreover, this approach would allow a more efficient discovery process to obtain results closer to the user needs. In this paper we present an approach for Web service discovery through the use of machine learning algorithms for classification of Web services. For Web services matching our proposal takes into account quality of service (QoS) parameters, which include semantic information about each Web service. For semantic Web service specification we use the SAWSDL standard. Whereas the proposed UDDI standard was discontinued, among other reasons, due to limited syntactic Web service discovery, our approach brings important elements to the consolidation of semantic Web service discovery.**

Index Terms—**Web services discovery, semantic Web services, non-functional properties, naïve bayes, support vector machines, adaboost**

I. Introduction

Web Services (WS) are large distributed applications used under different environments such as enterprises, industry and government, in order to implement complex systems. Like any distributed system, Web services are susceptible to failures, for instance the failure of the server where a service resides or a failure in the channel of communication between a service provider and the requester. Another important problem taking place within Web services systems is that of finding the most appropriate service according to the needs of the customers [1].

The Universal Description Discovery and Integration (UDDI) registry was proposed twelve years ago for the publication of services. WS clients can access this registry in order to find the best service that meets their needs. Since UDDI uses syntactic information and does not use semantic information, the search for the most appropriate service is limited in the sense that clients cannot make requests asking for specific desirable properties such as quality of service parameters related to reliability, performance, security, response time, etc. These are some of the reasons that caused UDDI to be discontinued since 2006 [2].

Manuscript received October 20, 2012; revised November 15, 2012; accepted November 25, 2012

With the disuse of UDDI, several proposals have emerged such as URBE [3]. URBE analyzes the structure and terms of the Web Service Description Language (WSDL) file that defines the Web service. This descriptor file is now employed to replace a failed WS.

The combination of the theory of Semantic Web and Web services gives rise to what is known as Semantic Web Services. There are several approaches to add semantic information to services such as: OWL-S [4], WSDL-S [5] and WSMO [6].

Although it is possible to model ontologies to incorporate quality of service information, there are no commonly accepted techniques for finding and comparing the degree of similarity between Web services. In this paper we propose the use of machine learning algorithms as a better alternative to classify and match WS. Machine Learning has been used in the phase of composition of Web services using algorithms such as: ant colony and genetic algorithms [7][8]. Other approaches have used the Naive Bayes algorithm for discovering Web services together with WSMO [9][10]. After mentioning an overview of the service discovery problem, the Section II describes the related works. Section III gives details of the methodology here proposed. Finally, we present the experiments and conclusion.

II. Related Work

Web service discovery consists in retrieving Web services from a registry, which meets the requirements of the request [11]. The approach in [11] specifies a classification of Web services discovery in which the authors define two categories: syntactic and semantic discovery. In the first category, they present techniques of Information Retrieval, based on schemes and links. In the second category semantic discovery is implemented by means of concepts and ontologies. The discovery based on concepts retrieves the semantic information from WSDL files to measure the similarity of parameters. The discovery based on ontologies make a model of semantic information using some approaches like OWL-S and WSMO.

In [2] the authors proposed an empirical approach for Semantic Web Service discovery where a service provider deploys a Web service. Then a collector service collects all the WSDL from services, and finally this information is forwarded to a processor, which parses the files in order to retrieve relevant information. This information is used to generate a document of terms which is transferable to a Vector Space Model (VSM) , indexed using the LSA indexer.

In [12] measurements such as recall and precision are used to determine the degree of match between a user's request and a service response. Given the large number of partial results produced by this approach, it requires a ranking mechanism with the purpose of preparing the process of semantic Web services discovery.

In [13] the authors also propose, as a first stage, to calculate the degree of match between WSDL descriptions using the vector space model, which is an algebraic model to represent text documents as a vector of identifiers. In a second stage, the services are paired through the use of a hierarchical ontology. Finally, a framework for semantic description based on quality of service (QoS) is employed. This approach needs the specification of ontologies in order to add semantic information about quality of service parameters. There are also works that make use of machine learning algorithms such as [9], where a Bayesian classifier is employed. In this approach, four non-functional properties are utilized: location, price, trust and QoS. To model these non-functional properties the Web Service Modeling Ontology (WSMO) is used together with the Naive Bayesian classifier. The training set used by the authors was taken from www.xmethods.net, and it consisted of the values of four non-functional parameters of different Web services. There are proposals that combine different approaches, that is the case in [14], which combines the URBE architecture (UDDI Registry By Example) and DIRE (Distributed Registry). To perform the semantic matching they used URBE, because it has an engine of similarity. This engine is responsible for comparing the request sent from the client with the WSDL files stored in the registry. To verify the similarity between services, the similarity engine makes comparisons between input and output operations proposed by the client, with the file stored in the registry. As a result, it returns a list of similar services.

III. METHODOLOGY

Figure 1 shows the steps of our methodology. The first step consists of adding to each Web service, semantic information with non-functional properties. Such non-functional properties are: reliability, performance, availability, integrity, security and price. Each property could take five possible values: very high quality, high quality, regular quality, low quality, and very low quality.

Among the several existing approaches for adding semantic information to Web services, we have chosen SAWSDL. This approach adds semantic information through annotations in the file that describes the Web services. We employed the FUSION registry [15] to publish semantic Web services with annotation SAWSDL.

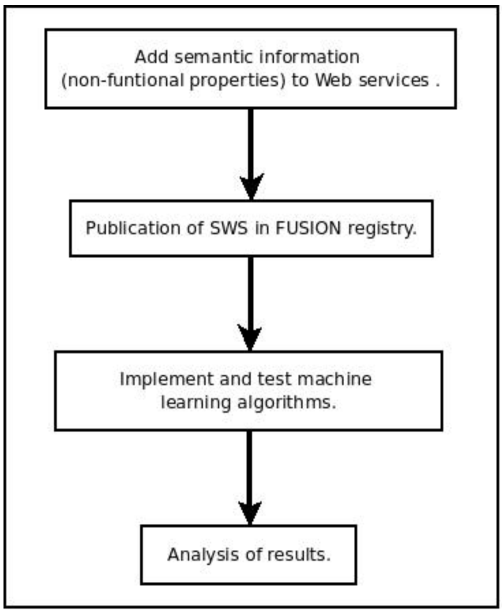
Add semantic information (non-funtional properties) to Web services .
Publication of SWS in FUSION registry.
Implement and test machine learning algorithms.
Analysis of results.

Figure 1. Steps of the proposed methodology.

Once that every Web service has been published, we used Machine Learning algorithms to classify the SWS. We focused our experimental work on three methods: Naïve Bayes, Support Vector Machine (SVM) and AdaBoost.

A. Naïve Bayes

We let x be a feature vector of length equal to the number of non-functional parameters and y the target output. We need to model $p(x|y)$. We will assume that all x_i are conditionally independent given y . Then, according to naïve Bayes algorithm, we have [16]:

$$p(x_1, x_2, \ldots, x_n \mid y) = \prod_{i=1}^{n} p(x_i \mid y) \tag{1}$$

where $p(x_1, x_2, \ldots, x_n \mid y)$ is the joint probability of $x_1, \ldots, x_n$ given y.

B. Support vector machine

From a set of training data $\{x_i, y_i\}_i^n \in R^m \times \{\pm 1\}$, the goal of the algorithm is to find the optimal hyperplane that divides the two kinds of data. The corresponding hyperplane may be defined as [17]:

$$w^T x + b = 0 \tag{2}$$

Let ρ the separation margin, which is defined.

$$\rho = 2\mathrm{r} = \frac{2}{\| w \|} \tag{3}$$

Thus, to ensure that we will find the optimal hyperplane, we minimize ρ with respect to x and b:

$$\min \frac{1}{2} ||w||^2 \tag{4}$$
$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1, i = 1 ..., n$$

When the data is not linearly separable we can use two techniques to solve it: soft margin optimization or the kernel function trick [18].

If you use a kernel, you get the optimal classifier

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \tag{5}$$

where α is the optimal Lagrange multiplier $K(x_i, x)$ is called kernel function. Some of the most common kernel functions are:

Polynomial:
$$K(x_i, x) = (x_i^T x + c)^d$$

Radial basis:

$$K(x_i, x) = \exp(-\gamma \|x_i - x\|^2), \gamma > 0$$

Sigmoidal:

$$K(x_i, x) = \tanh(x_i^T x + c)$$

C. AdaBoost

The algorithm has as input vector, variables going from x_1 to x_n and as targets, variables from t_1 to t_n where $t_i=\{1,-1\}$. Initially, all training examples have the same weight.

For each iteration the current classifier classifies the examples that may have been incorrectly classified by the previous classifier. After the training, the classifier forms a committee with different weights given to each weak classifier [17].

The algorithm initializes the weight coefficients in the data as $w_n = 1/N$ for $n = 1, \ldots, N$ (where N is the number of input vectors) for $m = 1, \ldots, M$ (where M is the number of classifiers). We minimize the weight of the error function assigned to a classifier $y_m(x)$ in the training set.

$$D_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{6}$$

When $y_m(x_n) \neq t_n, I(y_m(x_n) \neq t_n)$ it equals 1, otherwise it equals 0. Then we evaluate

$$\varepsilon_m = \frac{\sum w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum w_n^{(m)}} \tag{7}$$

and

$$\beta_m = \ln\left\{\frac{1-\varepsilon_m}{\varepsilon_m}\right\} \tag{8}$$

Updating the weight coefficients with

$$w_n^{m+1} = w_n^m \exp\{\beta_m I(y_m(x_n) \neq t_n)\} \tag{9}$$

In the end, the following model is used to make predictions.

$$Y_M(x) = sign(\sum_{m=1}^{M} \beta_m y_m(x)) \tag{10}$$

To measure the performance of the algorithms, we used the statistics known as recall and precision. These are defined according to the percentages of true positives and false positives. Assume that you have a binary problem, where data has two possible values: true or false. The true positives (TP) are those examples that being positives, the algorithms classify them correctly as positives. The false positives are negative examples that are classified as positives, therefore they are false positives. Equation (11) defines the recall statistics, which is basically the percentage of true positives. The precision is calculated with Equation (12) as the number of true positives over the sum true positives and false positives (FP).

$$\text{Recall=TP} \tag{11}$$

$$\text{Precision=TP/(TP+FP)} \tag{12}$$

IV. Experiments and Analysis

We distinguished two possible experimental scenarios, namely Case 1 and Case 2. These scenarios usually take place in the context of Web services discovery. In Case 1, we considered the customers' needs; that is, the Web services have values of very high quality and high quality, in their non-functional properties, so they have a high probability of being selected. However, it is not always possible to get high quality services; Case 2 characterizes this situation. For each value of non-functional properties we have a probability. In the following two sections we present the description of both cases.

A. Case 1

This case considered the needs of customers. So, the Web services that have higher quality are more probable to be selected. Our training data is composed of 1200 examples, each one represented by six non-functional properties. Those properties take five possible discrete values: very high, high, regular, low and very low quality.

Before executing the Support Vector Machine (SVM) algorithm, we have to do an analysis to find the best values for parameters C and γ. In Fig. 2 you can see the accuracy rate for the SVM with a 10-fold cross-validation. The accuracy obtained is 98.82%, the values of C and γ can be chosen from any place up to the curve on top, which bounds the area for the best parameters.

TABLE I.
RECALL AND PRECISION RATE FOR ALGORITHMS NAÏVE BAYES, SVM (c = 1:0 and γ= 2:0) AND ADABOOST FOR CASE 1.

Service type	Naïve Bayes		SVM		AdaBoost	
	Recall	Precision	Recall	Prec.	Recall	Prec.
Very high quality	0.995	0.985	0.99	1.0	0.99	0.9480
High quality	0.99	0.986	0.986	0.981	0.981	0.985
Regular quality	0.976	1.0	0.981	0.953	0.971	1.0
Low quality	1.0	0.995	0.99	1.0	0.967	0.995
Very low quality	1.0	0.9950	0.9800	0.9950	0.9950	1.0
μ	0.9722	0.9922	0.9853	0.9840	0.9950	0.9856

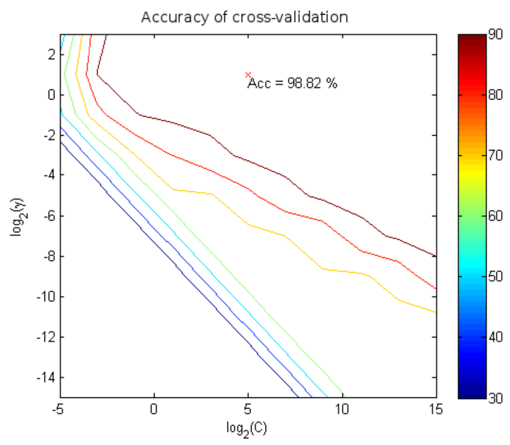

Figure 2. Accuracy rate with 10-fold cross-validation for case 1.

We chose the pair of values C = 1.0 and γ= 2.0 for the experiments. In Table I we provide the recall and precision rate for each algorithm. Observe that the Naïve Bayes gets 1.0% of recall for the low and very low quality services. Meanwhile, SVM gets high percentages of precision for very high quality and low quality services. With AdaBoost, we get a high percentage of precision, for regular and low quality services.

Overall, the three algorithms provide good rates of precision and recall. Naïve Bayes was the algorithm with the best mean percentage of recall, meanwhile the AdaBoost algorithm had the best mean percentage of precision. All algorithms were tested using a 10-fold cross-validation procedure.

B. Case 2

When we consider the common situation under which Web services must work, it is easy to notice that non-functional properties do not have the same probability to be chosen. The training data for this second case is composed of 1000 examples. We considered six non-functional properties taking five possible discrete values.

TABLE II.
RECALL AND PRECISION RATE FOR ALGORITHMS NAÏVE BAYES, SVM (c = 64 and γ = 1.5) AND ADABOOST FOR CASE 2.

Service type	Naïve Bayes		SVM		AdaBoost	
	Recall	Prec.	Recall	Prec.	Recall	Prec.
Very high quality	1.0	0.9850	0.995	0.99	0.995	0.966
High quality	0.986	0.995	0.985	0.976	0.971	0.99
Regular quality	0.985	1.0	0.966	0.99	0.985	1.0
Low quality	0.985	1.0	1.0	1.0	0.985	0.9900
Very low quality	1.0	0.974	0.995	0.985	0.984	0.974
μ	0.9912	0.9908	0.9882	0.9882	0.984	0.984

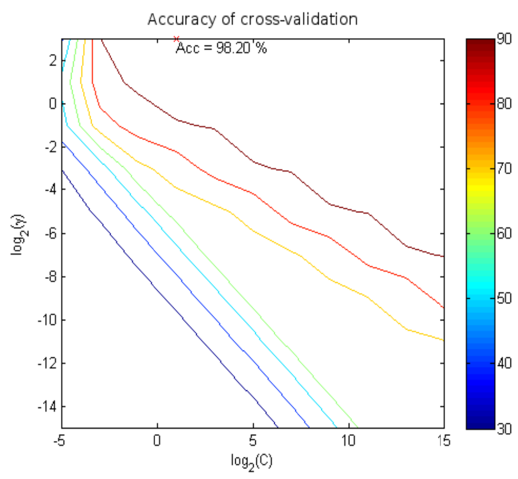

Figure 3. Accuracy rate with 10-fold cross-validation for case 2.

Similarly to the previous case, the first step consisted in finding the best values for C and γ. In Fig. 3 we illustrate the best combinations of parameters for the SVM. We obtained 98.20% of accuracy with a 10-fold cross validation. The Table II shows the percentages of recall and precision we got in the Case 2. The Naive Bayes algorithm reaches 1.0% of recall in both categories, very high and very low quality Web services. The best percentage is in the categories regular and low quality Web services. With respect to SVM algorithm, we got the same percentage in recall and precision, in categories of low quality; hence, this algorithm classified well the low quality Web services.

V. Conclusion

All three algorithms had good average rates of recall and precision. However, the Naive Bayes method provided the best results.

In this article we have proposed an alternative method based on Machine Learning for the classification of semantic Web services. With our method, Web services are automatically classified, which results in a discovery process more agile than other traditional methods. It allows retrieving a Web service that not only meets the requirements of a client, but it also satisfies some non-functional properties.

With the results obtained, we can conclude that the presented approach is feasible and could be implemented on architectures that require the replacement of a Web service that has failed or is not available. Given that the services are classified accordingly to their semantic information, the algorithm retrieves a service similar, in terms of their properties, to the one that needs to be replaced.

References

[1] J. Cardoso and A. Sheth, "Semantic e-workflow composition," *Journal Intelligent Information Systems*, vol. 21, no. 3, pp. 191–225, 2003.

[2] C. Wu, E. Chang, and A. Aitken, "An empirical approach for semantic web services discovery," in *19th Australian Conference Software Engineering*, 2008.

[3] P. Plebani and B. Pernici, "Urbe: Web service retrieval based on similarity evaluation," *IEEE Transactions on Knowledge And Data Engineering*, vol. 21, no. 11, pp. 1629–1642, November 2009.

[4] D. Martin, "Owl-s: Semantic markup for web services," W3C Member, Tech. Rep., 2004.

[5] R. Akkiraju, J. Farrel, J. Miller, M. Nagarajan, M.-T. Schmidt, S. A., and K. Verma, "Web service semantics-wsdl-s," Tech. Rep., November 2005.

[6] D. Roman, U. Keller, H. Lausen, J. de Brujin, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, "Wsmo-web service modeling ontology." in DERI Working Draft, vol. 1, Digital Enterprise Research Institute (DERI). IOS Press, 2005, pp. 77–106.

[7] M. Tang and L. Ai, "A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition," in Proceeding of the *2010 World Congress on Computational Intelligence*, Barcelona, 2010.

[8] D. Ling and X. Zhou, "A service-oriented component composition approach based on ant colony algorithm," in the *IEEE International Symposium on Intelligent Ubiquitous Computing and Education*, 2009.

[9] O. Shafiq, R. Alhajj, J. Rokne, and I. Toma, "Light-weight semantics and bayesian classification: A hybrid technique for dynamic web service discovery," in *Information Reuse and Integration, IEEE International Conference on.* Las Vegas Nevada, 2010.

[10] S. Chitra, K. Vidhya, and G. Aghila, "Web service selection based on ranking of qos using naive bayes through ontology mapping," in *IEEE International Conference & and Communication and Technology*, 2010.

[11] K. Vaneet, Sharma; Mukesh, "Web service discovery research: A study of existing approaches," *Int. Journal on Recent Trends in Engineering & Technology*, vol. 5, no. 1, March 2011.

[12] D. Skoutas, A. Simitsis, and T. Sellis, "A ranking mechanism for semantic web service discovery," in *IEEE Congress on Service*, 2007.

[13] L. Zhou, "An approach of semantic web service discovery," in 2010 *International Conference on Communications and Mobile Computing, IEEE*, 2010.

[14] L. Baresi, M. Miraz, and P. Plebani, "A flexible and semantic aware publication infrastructure for web services," in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, Z. Bellahsne and M. Lonard, Eds. Springer Berlin / Heidelberg, 2008, vol. 5074, pp. 435–449.

[15] D. Kourtesis and I. Paraskakis. "Combining sawsdl, owl-dl and uddi for semantically enhanced web service discovery". In The Semantic Web: Research and Applications, vol. 502, LNCS, pp. 614–628. Springer, 2008.

[16] A. Ng, "Lecture notes. parte iv. Generative learning algorithms," course notes of Machine Learning.

[17] Q. C. S. Xue, H. Yang, "The Top Ten Algorithm in Data Mining", Chapman & Hall/CRC, 2009, ch. Support Vector Machine.

[18] C. M. Bishop, Pattern recognition and Machine Learning. Springer, 2009.

Martha Varguez-Moo received her BS degree in computer science at Universidad Autónoma de Yucatán, México. She is currently pursuing her M.Sc. in computer science at the same university. Her current research interests include Web services and machine learning.

Francisco Moo-Mena is a Professor in Computer Sciences at Universidad Autónoma de Yucatán, in Mérida, Mexico. From the Institute National Polytéchnique de Toulouse, in France, he received a Master Degree in Computer Science and a PhD, in 2003 and 2007, respectively. He also received another Master Degree in Distributed Systems from the Instituto Tecnológico y de Estudios Superiores de Monterrey, Mexico, in 1997. He received a BS in Computer Systems Engineering from the Instituto Tecnológico de Mérida, Mexico, in 1995. His research interests include Self-healing systems, Web services architectures, and Semantic Web services.

Victor Uc-Cetina received the PhD degree (Dr. rer. nat.) in Computer Science from Humboldt-Universität zu Berlin, Germany, in 2009; his M.Sc in Intelligent Systems from ITESM in 2002 and his BS in Computer Systems Engineering from ITM in 1999. He is currently working as an assistant professor in the Facultad de Matemáticas at Universidad Autónoma de Yucatán, México. His research interests include artificial intelligence, machine learning, Bayesian reasoning and vision learning.