An Analysis of Web Services Attributes for Discovery Support

Héctor Jimenez Salazar, Christian Sánchez Sánchez,
Carlos Rodríguez Lucatero, and Arturo Wulfrano Luna Ramírez

Universidad Autónoma Metropolitana Unidad Cuajimalpa
División de Ciencias de la Comumicación y Diseño
Departamento de Tecnologías de la Información
Av. Constituyentes 1054, 3er piso, Col. Lomas Altas, Miguel Hidalgo, C.P. 11950,
Mexico D.F., Mexico
{hjimenez, csanchez, crodriguez, wluna} @correo.cua.uam.mx

Abstract. The number of Web Services, available over the Internet, has increased due the Web Services properties that allow them to be reused and to develop loosely coupled systems. Nevertheless, the current number of Services complicates the discovery, that is why it is necessary to improve the current mechanisms, matching and similarity, that support locating them. This approach focuses in analysing the Web Services attributes, which are contained in their descriptions. Our aim is to contribute to identify the role of attributes at the measurement of structural similarity function between web services and, through clustering, to develop resources for classifying and improving discovery. This proposal was tested through experiments over a collection of Service descriptions in WSDL which can be taken as an standard referent for such experiments. We show that our proposal outperforms the baseline taken as the best $F-$measure when the collection is represented by any attribute.

Keywords. Web services, discovery support.

1 Introduction

Currently, the number of Web Services (WS), available over the Internet, has increased due the Web services properties that allow them to be reused and to develop loosely coupled systems, as it can be noticed in available WS registries (or UDDIs) like Xmethods [1] and Seekda [2]. Web services are described by some languages, such as WSDL, OWL-S or WSMO.

In this paper we worked with WSDL, because is more common to find descriptions in this language than in the others, which are semantic languages. A WSDL[4] document defines services as collections of network ports, 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.

Nevertheless, the current number of services complicates the discovery taking into account that it is hard for human beings to read Web Service descriptions in WSDL and to search inside them.

Regarding to software systems, the accessible UDDI's just match keywords, contained in the user request, within the whole service descriptions, based on a business oriented classification. When a provider registers a service in an UDDI he/she chose a category of business, but also the searching through these categories, some times this is not enough because the classification depends on the provider expertise and/or criteria.

The problem of searching by means of keyword, and to syntactically relate them can produce ambiguity and additionally don't take advantage of the internal WSDL structure.

According to the Semantic Web, service matching can be done analyzing the description of what the services do: through its service profile in the OWL-S language [3]. Those profiles have functional and non-functional properties in order to describe the service. Functional properties describe Service functionalities through showing methods and their Inputs, Outputs, Preconditions and Effects (IOPE). On the other hand, non-functional properties provide extra information as quality of service, country, provider info and so on. Unfortunately, the most of the services does not have a semantic description yet; they just have syntactic descriptions in WSDL.

Because of this problem and with the intention of providing semantics to the descriptions, some approaches like [5], [6] and [7] have been focused on relating (semi-automatically) parameter names and concepts in ontologies, despite of there are few semantic descriptions of services and ontologies.

For those reasons it is necessary: a) to exploit the embedded structure (attributes) of the Web Service description, b) to extract the description semantics, c) to identify attributes that help to define a suitable structural similarity function (matching) and d) to use semantics for developing resources in order to support the WS discovery. In this paper we focus the first point with the purpose of comparing attribute-based representation of WSDL and, then, identify the role of attributes on matching for discovery.

The remainder of this paper is organized as follows: Section 2 contains the state of the art, Section 3 shows how was gotten a non-structured document from a WSDL, Section 4 describes the experiment concerned to the influence of the attributes for representing WSDL, and finally in Section 5 concludes this work.

2 Matching and Clustering of WSDL

Concerning the extraction of the WSDL descriptions embedded semantics, using Natural Language Processing and Information Retrieval tools as well as the things regarding structural matching of the Web Services, it has been developed some approaches, as we will show in the following paragraphs.

Stroulia et al.[8] preprocess information from several WSDL descriptions in order to apply Information Retrieval methods to determine the similarity of the service descriptions, on the other hand also take into account the structure of WSDL documents to determine the similarity of two services, starting by

comparing the data types, and then compare the messages and their parameters to finally finish the execution of the operation. They compare these elements because they have the intuition that the names of these attributes usually reflect the semantics embedded service capabilities.

Using Information Retrieval, there is an approach proposed by Hao et al. [9]. Here the similarity function that help to rank the services, mainly takes into account three aspects:

1. The relevance of the service: according to the terms it shares with the request.
2. The importance of service: through the services most used by others.
3. The service connectivity: analyzing the similarity of the XML tree of the description.

However, in these approaches it has not been really proved that these attributes or aspects, going farther than the intuition, are the more important for obtaining the structural similarity function. On the other side, they use WordNet for obtaining the synonyms, hypernyms, hyponyms, nearer concepts and steems but due to the WordNet generality it can deviate to the true of those domains that contribute to find the description similarity.

Also, seeking to identify major similarity between the descriptions and accelerate the discovery, there are some approaches that seek to categorize Web services, as in the case of Bruno et al. [11], that in addition to using Information Retrieval techniques, they sort the services in classes (predetermined by them) using the concept extracted from the descriptions that vectorize, so services can be classified using support vector machines in order to sort them into a lattice, using IS-A relationships. Unfortunately, these approaches have not scaled up on several collections of WSDL.

Liang et al. [12] show another approach where categorization is used. Their approach is focused on finding similar services working with semantic and syntactic descriptions. They categorized WS schemes to match WS that can operate in heterogeneous domain ontologies. Having an upper ontology and using its OnEx-Cat tool, they can determine when a Web Service is a possible replacement by another. They considered the categorization of ontologies as a term categorization search problem, which also it is similar to the task of document classification. As in the previous approaches, in these works it is not done attribute selection for supporting a good classification.

Working under the hypothesis that the automatically generated clusters will be able to suggest similar services, some authors have grouped web services information, obtained from WSDL and records (information registered by the creator in UDDIs). For instance, Fan et al. [13] joined the recording service information, the service documentation and their operations, and started to form groups using the Hierarchical Agglomerative Cluster (HAC) algorithm and the Jaccard similarity measure. They found a very big amount of noise in the formed groups. By this reason they concluded that many descriptions lacked of documentation, and that there were not enough information in the recorded information for distinguish it from others in the clustering.

Similarly, Dong et al. [14] created an algorithm to group in concepts having semantic meaning, the parameter names of the Web Services operations, by means of which, the similarity of the input and output operations of the services can be determined. Their algorithm works as sequence of refinements of the classical agglomerative clustering, that step by step associates groups that share terms that are nearly related and meet the cohesive property. However, when the groups are found by the algorithm they can be contamined by noisy information, and then, for trying to improve the obtained results, this noise is eliminated by using some heuristics. For verifiying similarity between two services, his algorithm mix the information concerning the services, operations, inputs, outputs, documentation and recording, and conclude that the performance improvement is not greatly enhaced by simple combination of the aforementioned information.

The found problems for classifying WS description documents are: there are not enough efforts to categorize them, and the existent descriptions are keep updated by its providers with different level of background and expertise. Web Services are available dynamically and the classes become obsolete rapidly. As is known the providers do not share the same vocabularies and the most of the times use n-grams or compound words.

As we have seen, all tasks related to discovering and/or reusing web services rely on similarity, therefore representation of WS is an important topic. Furthermore, this problem demands support from domain dependent resources and/or to use adequately the attributes of WS. Motivated by this conclusion we want to determine what WSDL attributes enable us to enhance the clustering of the Web Services, making easier the discovering task by giving hints about the structural similarity function of the elements.

3 Getting a Non-structured File from WSDL

Description of a web service by WSDL constitutes a reusable binding. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types**- a container for data type definitions using some type system.
- **Message**- an abstract typed definition of the data being communicated **(parameters).**
- **Operation**- an abstract description of an action supported by the service.
- **Documentation**- description in natural language to describe the service and/or its operations.
- **Port Type**- an abstract set of operations supported by one or more endpoints.
- **Binding**- a concrete protocol and data format specification for a particular port type.
- **Port**- a single endpoint defined as a combination of a binding and a network address.
- **Service**- a collection of related endpoints.

According to Stroulia et al. [8] information like the services' names associated to the methods, parameters and data types is useful, because they reflect the semantics of the underlying capabilities.

Our approach considers a preprocessing step before performing the analysis which is explained in Section 4.

Fig. 1. Preprocessing of a non structured short WSDL document.

The preprocessing consists of extraction of the words contained in the attributes of the descriptions of Web services. The attributes considered for the analysis were the following: Name of the Service, Name of the Operations, Documentation, Name of the Messages, Name of the parameters. Since the documentation is a natural language description, the extraction of words was done by taking all the words contained and separated by spaces.

For the case of other attributes, obtaining words are performed as follows.

– **Step 1**.- Analyzing Names. Identifying the names formed by one known word or composed words, using a dictionary (WordNet), this is done verifying if the name is contained in the dictionary.

– **Step 2**.- Breaking names into single words. For finding a set of terms related to a name, the next alternative procedures are proposed.
  a) Obtaining all the possible substrings from the names: This method can be applied to every composed word name.
    • For each name a number k of substring are gotten, where $k = \sum_{i=0}^{l}(l-i)$ and l is the length of the name.
    • When we have all the substrings then we identify which of them are known words, as it is done in Step 1, with the purpose of relating them to each name.
  b) Obtaining strings which starting prefix is a capital letter. This string treatment is applied to every composed word name that starts with a capital letter and is formed by other capital and small letters.
    • The name is broken into k substrings, where $k = C$ is the number of capital letters contained in the string. The name is cut from each contained capital letter until finding other capital letter or the end of the name.
    • When we have all the substrings then we identify which of them are known words, as it is done in Step 1, with the purpose of relate them to each name.
  c) Obtaining strings that are separated by special characters This string treatment can be applied to every composed word name that contains special characters; where $(\_ - =:,)$ were all the characters found in the used colecction to separate words.
    • The name is broken into k substrings, where $k = S + 1$ if the prefix of the name is not a special character, or $k = S$ if it is. The name is cut from the beginning or where is found the special character $+1$ until finding other special character or the end of the name.
    • When we have all the substrings then we identify which of them are known words, as it is done in Step 1, with the purpose of relate them to each name.
– **Step 3**.- Deleting stop words. The list of all deleted words, *articles, prepositions, pronouns, etc.*, can be found in lextek[1].

4 Analyzing the Description of WS

As we have said the help for classifying and searching WS has an important implication for the information resources at the existing WS. It does not only deal with the resource compilation helping to describe most of the WS, besides, based on the volatility of the web services, it is convenient that service classes and derived resources being dynamic. So, we propose a methodology in order to build up elements of enhancement for WSDL classification.

The analysis, we will show here, gives importance to WSDL components through the clustering supported in some attributes alone or combined. First of

[1] `http://www.lextek.com/manuals/onix/stopwords1.html`.

all, we measure the performance of each attribute used to represent the whole collection of WSDL: which of them obtains the better clustering or, saying, which of them makes the best representation of WSDL. From these results we analyze some attribute combinations and its meaning aiming to improve the clustering.

4.1 Data Collection

The WSDL document collection that we selected, for the sake of proving this approach, and taking into account there are not few of them, can be found at ASSAM[2] [10]. That collection is composed by real Web services description documents, obtained from Salcentral and Xmethods. The WSDL documents are organized into a class hierarchy, that in some cases have subclasses with at most two levels of depth. The collection have 814 WSDL distributed in 26 classes. However, to prove our approach we needed to make two modifications:

- flatten classes (subclasses eliminate from the main classes),
- select WSDL documents which could be extracted at least one word (recognized by WordNet) for each attribute (name of service and methods, documentation, messages and parameters) that is part of the description of the WSDL document.

Therefore we gather the collection, which was reduced to 22 classes with 203 WSDL. The next table shows some data about the gathering collection:

Feature	**Value**
# Classes	22
# WSDL	203
$\|Vocabulary\|$	2,829
WSDL×Class (avg)	9.2
Terms × identifier (avg)	2.7
Terms × documentation (avg)	26.34

4.2 Experiments

The experiments were carried out considering:

Purpose. They focused to know the attribute quality given in a description of web services.

Testing. The analysis of hypothesis on the role of each attribute, given by the set of values that experiments take, is made through a combination of attributes. By instance, the hypothesis: `Name` attribute (of a method in the WS) may be part of a nominal phrase which is more precise when includes the `Parameter` attribute (names of); it is tested by means of the combination of those attributes.

Preprocessing. The preprocessing previous to the clustering:

[2] `http://www.andreas-hess.info/projects/annotator/index.html`.

1. selection of attribute(s) for representation,
2. lowerizing and deleting term repetition, and
3. determining a percentage of terms included in the set of values of the attribute(s) through the ranking given by a term selection technique.

It is important to address that we work with a bag of words representation. This decision is based on the low frecuencies of terms in descriptions which is not proper for the Vector Space Model [19]. So, we use the Jaccard coeficient to measure similarity between instances.

Attributes. Referring to the list of components of WS (see Sec. 3), the attributes taken into account at the experiments were the following:

- `Name` (of operation): terms taken from the identifier.
- `Parameters`: they are given by type names.
- `Documentation`: this is the only attribute regarding as a phrase of natural language.
- `Message`: terms provided by the procedure call.
- `All`: in this case all the above attributes are joined to represent each WS.

WS representation. Decision on instance representation requiered to test some Term Selection Techniques (TST). Since our purpose was the comparison among the attributes we did not be exhaustive on this point. Three TST were used: DF (document frequency) gives greater importance to terms that appear in more instances [15]; TP (transition point), the importance of a term is high when its frequency closes to the frequency which divides the vocabulary into the high and low frequencies [16]; and DEN (density) term importance is high if it contributes to form few classes [17]. From these TST we selected the one that obtained the best clustering performance was the best at development phase of the experiment. In the experiments we take percentages from 10 to 100 to representing each instance. Next step was the clustering of the represented instances.

Clustering. We are using a method of the $K - NN$ family, $k-$star [18], a divisive and non hierarchical method. The resulting clusters are evaluated by a metric supported on precision and recall measures (F).

4.3 Results

A prime measurement was determined applying directly the clustering procedure to the collection without term selection. Next table shows F values of each one of five attributes.

Attribute	F
`Name`	0.23
`Parameter`	0.23
`Documentation`	0.21
`Message`	0.22
`All`	0.23

All the attributes gave a poor performance: $F \in [0.21, 0.23]$. So, we used the DEN-TST in order to put into relevance the strength of attributes. Figure 2 shows the curve for each attribute: at horizontal axis, percentages of terms according to the Den-TST, starting from 10 till 100 percent of terms; vertical axis grades the F value determined by the clustering of the collection represented with the given percentage.

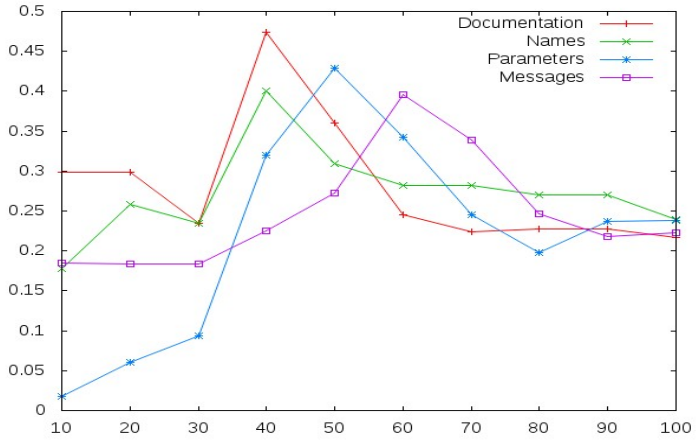

Fig. 2. Clustering performance of WS with four attributes.

`Documentation` attribute obtained the best performance, $F = 0.47$. The result is understood as: it is easier the matching between sets of terms forming sentences than the sets given by terms extracted from heterogeneous structures; i.e. phrases, or structured identifiers (types). However, this result has no great relevance due to that most of WS lacking of documentation. Rather we will consider the F value of `Documentation` as a baseline.

Our approach was to trying to "reformulate" a documentation from the other attributes. Three hypothesis were raised:

1. Attributes `Name` and `Parameter` have a role on documentation; i.e. they might constitute a paraphrase of some kind: *(To do) name (use)* par_1, ... par_n.
2. Since `Message` attribute has information about the procedure call, it might, in combination with `Name`, both would contribute to some paraphrase of documentation, namely: *Name (is used as) message.*
3. As well we might hope the combination `Name`, `Message` and `Parameter` could paraphrase some kind of documentation: *(The) name (operation takes)* par_1, ... par_n *(to proceed as) message.*

Therefore, to test the behavior of such combinations of attributes the clustering of collection was carried out. Figure 3 depicts the F values of the three combinations and they are confronted with F values of `Documentation`.

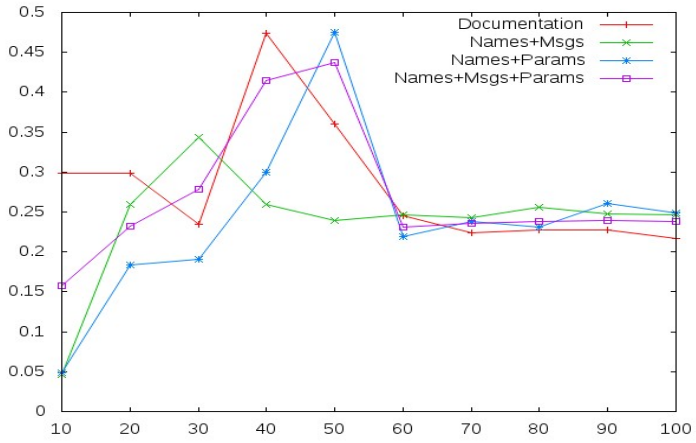

Fig. 3. Clustering performance of WS by attribute combination.

According to this result we claim that, in this collection, `Name-Parameter` ($F = 0.47$) combination may substitute `Documentation`. On the other side, the great difference of performance on the combinations `Name-Message` and `Name-Message- Parameter` is due to the noise provided by the inclusion of non related terms, in the second case.

Then, we can test the "virtual documentation" applying this representation to WS which have not `Documentation`. We used the complete collection, and applied the clustering procedure representing each WS by attributes `Documentation` and `Name-Parameter` combination. The following table summarizes some features of the collection:

Feature	**Value**
# Classes	25
# WSDL (with doc.)	213
# WSDL (without doc.)	226
$\|Vocabulary\|$	3,228
WSDL×Class (avg/stdev)	17.5/20.7

The result of the best clustering for those attributes were $F = 0.46$ for `Documentation` and $F = 0.53$ for `Name-Parameters`, which shows a clear en-

hancement for the WS representation through the proposed attribute combination `Name-Parameters`.

5 Remarks and Conclusions

On the context of comparing WS, we analyzed a better way to represent web services instead of a direct use of their attributes. Even though we can select the attribute which gives the best performance for comparing WS, the experiments carried out show the usefulness of combining attributes. We have found that, in the used collection of WS, `Name-Parameters` attribute combination outperforms `Documentation` attribute; which gives the better representation alone.

This result has relevance due to most of WS lack of documentation. Just 51% of the WS contained in the original collection `Documentation` is missing. This fact is also observed in many of the WS retrieved from repositories. For example, Fan et al. [13] reported that almost half the services do not have any documentation for any of the operations supported. Thus, for WS discovery, the lack of `Documentation` is an important problem to be tackled. In this work we have showed that `Documentation` attribute can be "reformulated" by the combination of other two: `Name` and `Parameters`, obtaining a better performance that the one of `Documentation`; $F = 0.47$ and $F = 0.53$ respectively.

Since the representation of data impacts the searching, classification and clustering of WS, it is important to continue the improvement of WS representation. By instance, it is necessary to incorporate, in some way, the rest of attributes contained in the WSDL description.

We have seen certain behaviour of WSDL attributes on a particular public collection, but it is necessary to support such behaviour using more supervised WSDL collections. In spite of the dynamic of WS on the web, it is imperative to consider standard supervised collections of WSDL aiming to compare diverse approaches on the task of discovery of web services.

Acknowledgments. We wish to thank to Conacyt-México by the given support to project grant Nr. 153315.

References

1. Xmethods, `http://www.xmethods.net`.
2. Seekda, `http://webservices.seekda.com/`.
3. Owl-s, `http://www.w3.org/Submission/OWL-S/`.
4. Wsdl, `http://www.w3.org/TR/wsdl`.
5. Guo, H.; Ivan, A.; Akkiraju, R. & Goodwin, R.: Learning ontologies to improve the quality of automatic web service matching. In: Proceedings of the 16th International Conference on World Wide Web, New York, NY, USA, ACM, pp. 1241–1242 (2007)
6. Qu, C.; Zimmermann, F.; Kumpf, K.; Kamuzinzi, R.; Ledent, V. & Herzog, R.: Semantics-Enabled Service Discovery Framework in the SIMDAT Pharma Grid. IEEE Transactions on Information Technology in Biomedicine. pp. 182–190 (2008)

7. Jaeger, M.C.; Rojec-Goldmann, G.; Liebetruth, C.; Mühl, G. & Geihs, K.: Ranked matching for service descriptions using owl-s. In: KiVS (2005)
8. Stroulia E. & Wang Y.: Structural and semantic matching for assessing web-service similarity. International Journal of Cooperative Information Systems. vol. 14, pp. 407–437 (2005)
9. Hao, Y.; Zhang, Y. & Cao J.: Web services discovery and rank: An information retrieval approach, Future Generation Computer Systems, vol.26 No.8, pp. 1053–1062 (2010)
10. Hess, A.; Johnston, E. & Kushmerick, N.: ASSAM: A Tool for Semi-automatically Annotating Semantic Web Services. Lecture Notes in Computer Science, Vol. 3298, pp. 320–334 (2004)
11. Bruno, M.; Canfora, G.; Di Penta, M. & Scognamiglio, R.: An Approach to support Web Service Classification and Annotation. In: Proceedings of IEEE International Conference on e-Technology, Hong Kong, China, pp. 138–143 (2005)
12. Liang, Qianhui Althea & Lam, Herman: Web Service Matching by Ontology Instance Categorization. scc. vol. 1, In: IEEE International Conference on Services Computing, pp. 202–209 (2008)
13. Fan, J. & Kambhampati: S.: A snapshot of public web services. SIGMOD Records, **34**(1) 24–32 (2005)
14. Dong, X.; Halevy, A.; Madhavan, J.; Nemes, E. & Zhang, J.: Similarity search for web services. In: Proceedings of VLDB, pp. 372–383 (2004)
15. Yang, Y. & Pedersen, P.: A Comparative Study on Feature Selection in Text Categorization. In: Proc. of International Conference on Machine Learning, pp. 412–420 (1997)
16. Jiménez-Salazar, H.; Pinto, D.& Rosso, P.: Uso del punto de transición en la selección de términos índice para agrupamiento de textos cortos. Procesamiento del Lenguaje Natural, **35**, España, pp. 416–421 (2005)
17. Jiménez-Salazar, H.; Sánchez, C.; Rodríguez, C. & Luna, A.: Modelación léxico semántica de descripciones de servicios web, In: 8o. Taller de Tecnologías del Lenguaje Humano, Tonatzintla (2011)
18. Shin, K. & Han, Y.: Fast Clustering Algorithm for Information Organization, Lecture Notes in Computer Science, vol. 2588, pp. 619–622 (2003)
19. van Rijsbergen: Information Retrieval. University of Glasgow (1979)