# A New Unsupervised Web Services Classification based on Conceptual Graphs

Eiman Boujarwah
CS Department, Kuwait University
POB 5969 Safat, Kuwait, 13060
ebujarwa@sci.kuniv.edu.kw

Hamdi Yahyaoui
CS Department, Kuwait University
POB 5969 Safat, Kuwait, 13060
hamdi@sci.kuniv.edu.kw

Mohammed A. Almulla
CS Department, Kuwait University
POB 5969 Safat, Kuwait, 13060
almulla@sci.kuniv.edu.kw

***Abstract*— With the drastic growth in number of deployed Web services, the discovery of a desired Web service is becoming a challenging research problem. In this paper, we develop a new unsupervised classification technique of Web services using conceptual graphs. A conceptual graph helps in building functional domains and classifying Web services into these domains. Such classification would speed up the discovery of a Web service and save the time of searching the whole Web service registry. The proposed algorithm is shown to have better performance than the Inductive Reasoning (IR) technique based on OWLS-TC benchmark.**

***Keywords-Web services; classification; Conceptual graphs***

## I. INTRODUCTION

Web services are a predominant information technology for the development of loosely-coupled and cross-enterprise business applications. Hence, Web Services are now considered as a trendy research topic. Several research initiatives investigated Web Services classification for discovery purpose. A Web Service interface is specified using Web Service Description language (WSDL) [13], which is a XML-based description language that describes the functionality of a service. It also provides the parameters that a Web service expects to be input to it along with the output parameters it returns.

The objective of this work is to develop an intelligent technique that leverages conceptual graphs in order to build functional domains and to classify Web services into these domains. Conceptual graphs are considered as a system for knowledge representation based on semantic networks. Conceptual graphs include concepts and relations. This structure should help to bootstrap and speed up the discovery of Web services.

The remaining part of the paper is organized as follows: Section II shows the related work, exploring different methods to classify Web services. Parsing and Stemming the WSDL file is introduced in Section III. Next, Section IV presents the proposed classification of Web Services method using conceptual graphs with the experimental results. Finally, the conclusion and future work are discussed in Section V.

## II. RELATED WORK

Classifying Web services is currently an important issue for the Web services community. Several research initiatives tackled this issue from different perspectives. Some of them mainly deal with the description of the Web Service, whereas others focus on the semantic perspective of the interface of a Web service, which is described using WSDL.

Wang et al. [1] proposed a method to manage service classification within a medium or big category. They used Support Vector Machine (SVM) text classification algorithm to classify Web services and they used United Nations Standard Products and Service Code (UNSPSC) as the classification criteria for these Web services.

Meditskos et al. [2] applied several machine learning algorithms to automatically classify Web services into their functional domains based on OWL-S advertisements. They combined the textural description of the Web service and its semantic description. Finally, they compared the accuracy of their algorithm with respect to other algorithms and they found that the semantic signature algorithm achieved better accuracy than the other algorithms.

Segev and Toch [3] provided an analysis of two methods for context-based matching and ranking of Web services for composition purposes. First, they analyzed two common methods for text processing: TF/IDF and context analysis, and two methods of service description namely free text and WSDL. Second, they presented a method for evaluating the proximity of services for possible compositions. Each Web service WSDL context descriptor is evaluated according to its proximity to other services' free text context descriptors. The proposed methods were tested on a large repository of real-world Web services. The experimental results concluded that context analysis is more useful than TF/IDF. Furthermore, the method of evaluating the proximity of the WSDL description to the textual description of other services provides high recall and precision results.

Elgazzar et al. [4] attempted to cluster Web services based on functional similarities. Their proposed technique leveraged the quality threshold of clustering algorithms to cluster similar Web services based on five elements found in the WSDL file namely: WSDL contents, types, messages, ports and service name.

Lately, Kiefer and Briensten [11] came up with a collection of inductive methods to perform classification. Their collection includes Rational Probability Trees (RPT) and Rational Bayes Classifier (RBC). The main idea was to explore links between objects to improve the classification process. Their proposed approach outperformed the kernel methods used in SVM.

Most of the related research initiatives adopt supervised classification methods to classify web services into functional domains. Contrarily, we propose a new unsupervised classification technique based on conceptual graphs. We advocate the use of conceptual graphs to achieve a high level of classification accuracy.

## III. WSDL PARSING AND STEMMING

As mentioned in the introduction, the WSDL is a XML-based language, which describes the functionality of the Web service and how to access a particular Web service. A WSDL document consists of four major elements beside the name of the Web service, these elements are: port type, messages, binding and types.

Port type is the most important element in a WSDL document; it describes the operations performed by the Web service. The messages element defines the data elements of an operation. Binding defines the data format and protocol for each operation. Finally, the types element is used as a container for data type definitions in the Web service. Figure 1 shows an example of a WSDL file.

```
<wsdl:portType name="CarPricequalitySoap">
<wsdl:operation name="get_PRICE_QUALITY">
<wsdl:input
message="tns:get_PRICE_QUALITYRequest">
 </wsdl:input>
<wsdl:output
message="tns:get_PRICE_QUALITYResponse">
</wsdl:output>
</wsdl:operation>
</wsdl:portType>
```

Figure 1. An example of a WSDL file

A Web service interface is described in a WSDL document. Our process of parsing a WSDL file [5] starts with extracting the service name, port type, operation names and input/output parameter names from the WSDL file.

After completing the aforementioned parsing, we perform a tokenization step to produce the set of terms for the Web service by filtering the giving names into terms according to several rules such as case changing, use of underscore and hyphenation, and use of numbers. Table I shows a list of examples of the tokenization rules and how it is applied.

TABLE I: TOKENIZATION RULES EXAMPLES

| Rule | Original | Tokenized |
|---|---|---|
| Case changing | SendEmail | Send, Email |
| Case changing | getListOfServices | Get, List, of, Services |
| Underscore | Computer_science | Computer, science |
| Numbers | Exchange1 | Exchange |

After that, terms are stemmed into their stemmed version. The stemming process is a well-known process and we use Porter stemmer algorithm [6] to deal with it. Table II shows a list of examples of the stemming process.

TABLE II: STEMMING EXAMPLES

| Original | Stemmed |
|---|---|
| Sending | Send |
| Cleaned | Clean |
| Taken | Take |
| Swimming | Swim |
| Easier | Easy |

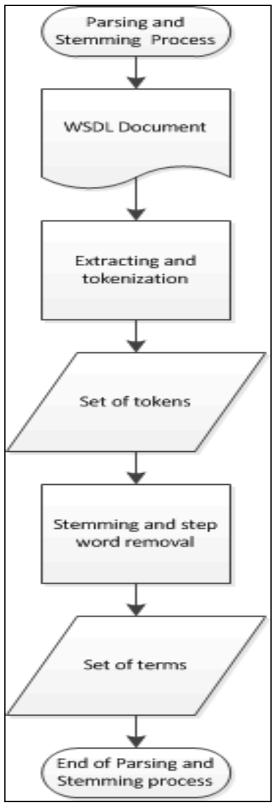
Parsing and Stemming Process
WSDL Document
Extracting and tokenization
Set of tokens
Stemming and step word removal
Set of terms
End of Parsing and Stemming process

Figure 2. WSDL file parsing process

Finally, we remove the stop-words; these are the words that are most commonly used in any English paragraph, short function words, such as: the, is, at, which and on. These words may cause a problem, so we removed them. The result of this process is a set of terms for each Web service. Figure 2 illustrates the process that we have just described above.

## IV. WEB SERVICES CLASSIFICATION

In this research, we consider the problem of how to classify various Web Services into domains according to their functionalities. Domains that will be used in this work are: communication, education, food, travel and weapon. We used the WordNet [7, 8, 9] as a lexical English database, which consists of nouns and verbs; these are grouped into sets of synsets. WordNet is the main conceptual graph and we build the conceptual graph for each domain from it.

We generate a conceptual graph for each functional domain as follows: the terms inside the WSDL are considered as the concepts nodes and the relations are considered as the same semantic relations as the one in WordNet; which are: is-a relations and kind-of relations. A conceptual graph is represented as a hypergraph. A hypergraph is a graph such as each edge can connect to any number of vertices. It is a powerful knowledge representation with higher order relationships between the graph nodes. Figure 3 shows an example of a hypergraph, where *V* is the set of vertices and *E* is the set of edges.

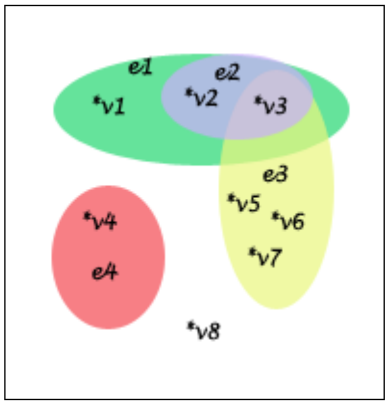

Figure 3. An example of a hypergraph, with $V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8\}$

$E=\{e_1,e_2,e_3,e_4\}=\{\{v_1,v_2,v_3\},\{v_2,v_3\},\{v_3,v_5,v_6,v_7\},\{v_4\}\}$

WordNet has a hypergraph that covers all the nouns and verbs (called terms) in the English language along with the semantic relations that connect these terms using the IS-A relation and the KIND-OF relation. We use this hypergraph to be the main conceptual graph from which we generate the hypergraph for each functional domain.

### *A. HGCA*

To start the classification process, we first create a hypergraph for each domain by extracting its contents from the WordNet hypergraph, which are related to a specific domain. We take the name of the domain and start a Breadth First Search in the WordNet hypergraph to create the domain hypergraph. This extraction process will take care of all the connected semantic relations in the new domain's hypergraph. Algorithm 1 shows the steps of generating a domain hypergraph. The implementation was done in the Java programming language and we used the hypergraph API to create, query and search through hypergraphs [10].

After having each domain hypergraph ready, we start the classification process. The hypergraph classification algorithm (HGCA) starts with the file name of the WSDL file for the Web service. First, we open the WSDL file and we perform parsing and stemming on the content of the WSDL file. Then, we compute the classification score for the Web service for each domain. We take the terms of the result from the parsing and stemming step and try to find if this term is matching any of the terms in the domain's hypergraph then we increase the score by one. We repeat these steps for the all terms of a Web Service. After we complete finding the classification score for each domain we compare them all together. The higher the score, the more this Web service belongs to the domain.

Algorithm 1. Generating Domain Hypergraph Algorithm

```
Algorithm 1: Generating Domain Hypergraph Algorithm
Input: WordNet Hypergraph, Domain Name and K-Level
Output: Domain Hypergraph
begin
    Find domain name in WordNet Hypergraph
    Start Breadth-First-Search from the domain name until we reach
    the end of this graph or the K-level.
end
Return
Domain Hypergraph
```

Our classification algorithm is based on number of nodes matched. We take the Web service and check the number of nodes matched in the domain hypergraph for each domain. For now, we will consider this number as the score for each domain and we will classify the Web service based on the maximum score. Algorithm 2 explains the procedure of the algorithm.

Algorithm 2. HGCA- Hypergraph Classification Algorithm

```
Algorithm 2:HGCA- Hypergraph Classification Algorithm
Input: WSDL document
Output: Score of the classified functional domain
begin
    Filter the WSDL document by parsing and stemming method.
    For each Domain
            Find matching terms inside the domain hypergraph.
            Compute classification score for this domain.
end
Return
The highest score and consider it as the classified domain
```

### *B. Experimental Results*

To explore the practicality of the proposed technique, a dataset is needed for testing. There are several benchmarks available online among which we chose the (OWLS-TC3) benchmark [12]. This data set consists of more than 700 Web services, covering seven domains namely: communication, economy, education, food, medicine, travel and weapon. Figure 4 shows the domain norm statistics of the benchmark dataset.

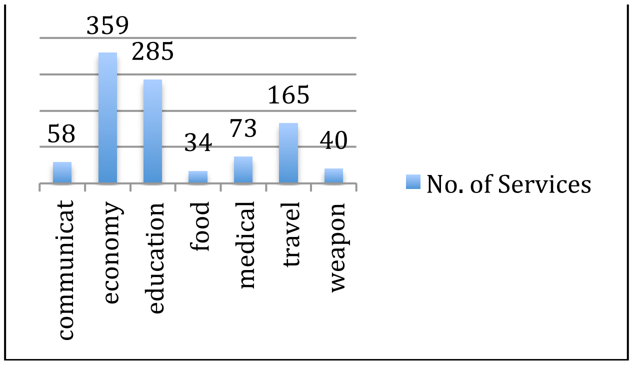

Figure 4. Norm Statistics of OWLS-TC3 benchmark

In our experiments, we randomly selected 200 Web services from the benchmark. It turned out that these Web services belong to five functional domains: communication, education, food, travel and weapon. We got 140 Web services that were correctly classified, 44 Web services not correctly classified and 16 Web services classified not to their functional domains but to other possible domains. For example, a Web service with the name FoodPrice.wsdl should be classified to the economy domain but it appeared in the food domain, which makes sense.

To evaluate the effectiveness of our algorithm, we had to compute the accuracy of the algorithm along with precision and recall. Precision measures the correctness of a classifier and recall measures the completeness of a classifier. When the precision is high, this means that the algorithm has returned more relevant results and when the recall is high it means that the algorithm returned most of the relevant results.

The accuracy of HGCA is almost 80% when we combine the correctly classified Web Service with the Web services that are classified to other possible domains. The precision and recall values for each domain are listed in Table III.

TABLE III: HGCA EFFECTIVNESS

| Domain | Comm. | Edu. | Food | Travel | Weapon | Avg. |
|---|---|---|---|---|---|---|
| Precision | 0.35 | 0.61 | 1 | 0.95 | 0.33 | 0.65 |
| Recall | 0.92 | 0.63 | 0.94 | 0.69 | 0.33 | 0.70 |

Finally, we compared our work with the IR method [11]. Table IV shows the average of the precision and recall for our algorithm compared to IR-based algorithm for five functional domains: communication, education, food, travel and weapon.

TABLE IV: PRECISION AND RECALL COMPARISONS

| Method | Avg. Precision | Avg. Recall |
|---|---|---|
| HGCA | 0.65 | 0.70 |
| IR | 0.61 | 0.42 |

As can be seen from the table above, the HGCA method outperforms the Inductive Reasoning method in the recall, which means that HGCA returned most of the relevant results and it is more complete. On average, the precision is almost the same with slightly difference between the two methods.

For the same experiment, we computed the average execution time in milliseconds of HGCA; figure 5 shows the scalability of the average execution time in milliseconds of HGCA.

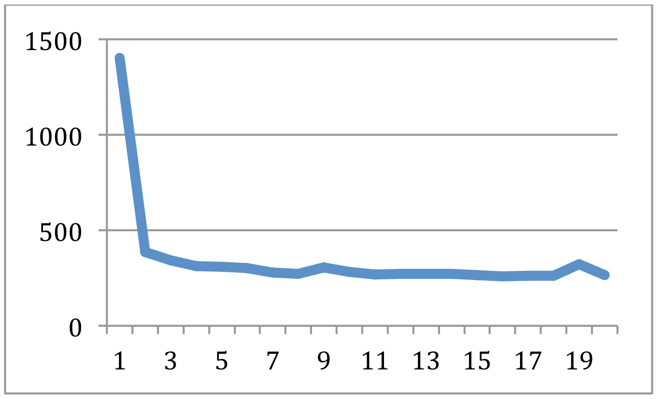

Figure 5. Avarge execution time of HGCA

When the number of services is small it takes longer time because of loading the domain hypergraphs for the first time into the memory.

Figure 6 shows the execution time per domain, we computed the execution time considering only one domain at a time to see the impact of each domain on HGCA.

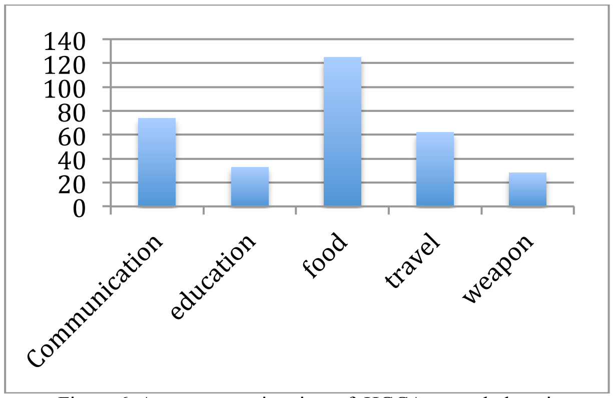

Figure 6. Avarge execution time of HGCA on each domain

The food domain takes longer time than the other domains because of the size of the domain itself to be loaded in the memory. The average execution time of all the domains is 64.4.

## V. CONCLUSION AND FUTURE WORK

In this research, we presented a new unsupervised technique for classifying Web services into functional domains based on conceptual graphs. A conceptual graph helps in identifying functional domains and classifying Web services into these domains. Such classification would reduce the search time of specific Web services. In our future work, we are planning to improve our classification method by trying different semantic similarity equations to compute the score of each classified domain; this would give us more accurate results. Furthermore, we are planning to leverage this technique to find the similarity between Web services based on their interfaces that are described in WSDL. The similarity issue between two Web services is equivalent to a matching problem between two conceptual graphs.

## REFERENCES

[1] H. Wang, Y. Shi, X. Zhou, Q. Zhou, Sh. Shao, and A. Bouguettaya, "Web Service Classfication using Support Vectore Machine", IEEE International Conference on Tools with Artificial Intelligence, vol. 1, pp. 3-6, 2010.

[2] I. Katakis, G. Meditskos, G. Tsoumakas, N. Bassiliades, and I.P. Vlahavas, "On the Combination of textual and semantic descriptions for automated semantic Web service classification", In AIAI, vol. 296 of IFIP, Springer, pp. 95-104, 2009.

[3] A. Segev, and E. Toch, "Context-Based Matching and Ranking of Web Service for Composition", IEEE Transactions of Services Computing, 2(3): 201-222, 2009.

[4] K. Elgazzar, A. Hassan, and P. Martin, "Clustering WSDL Documents to Bootstrap the Discovery of Web Services", IEEE International Conference on Web Services, Florida, Miami, pp. 147-154, 2010.

[5] E. Boujarwah, "A New Approach to Measuring Similarity Between Web Services", Third Kuwait e-Services and e-Systems Conference (KCESS-2012), Kuwait, 18-20th December, 2012.

[6] K. Sparck Jones, and P. Willet, Readings in Information Retrieval, San Francisco: Morgan Kaufmann, ISBN 1-55860-454-4, 1997.

[7] G.A. Miller, "WordNet: A Lexical Database for English", Communications of the ACM, vol. 38(11): 39-41, 1995.

[8] Ch. Fellbaum, "WordNet: An Electronic Lexical Database", Cambridge, MA: MIT Press, 1998.

[9] Princeton University, "About WordNet", Princeton University, http://wordnet.princeton.edu, 2010.

[10] I. Borislav, K. Vandev, C. Costa, M. Marinov, M. de Queiroz, I. Holsman, A. Picard, and I. Bogdahn, "HypergraphDB 2010", www.hypergraphdb.org, Last visit: 13th October, 2012.

[11] Ch. Kiefer, and A. Bernstein, "Application and Evaluation of Inductive Reasoning Methods for the Semantic Web and Software Analysis", Reasoning Web 2011, LNCS 6848, pp. 460-503, 2011.

[12] M. Klusch and P. Kapahnke, "SemWebCentral OWL-S", http://projects.semwebcentral.org/projects/owls-tc/, Last visit: 21st September, 2010.

[13] E. Christensen,F. Curbera,G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL)", www.w3.org/TR/wsdl, Last visit: 15th March, 2010.