Clustering of Web Services Based on WordNet Semantic Similarity

Aparna Konduri[b] and Chien-Chung Chan[a, b]
Department of Information Communication[a]
Kainan University
No. 1 Kainan Road
Luchu, Taoyuan County 338
Taiwan
chan@mail.knu.edu.tw
Department of Computer Science [b]
University of Akron
Akron, OH 44325-4003
USA
chan@uakron.edu

Abstract

As service-oriented architecture is getting popular, vast numbers of web services have been developed over a broad range of functionalities. It becomes a challenging task to find the relevant or similar web services using web services registry such as UDDI manually. Current UDDI search uses keywords from web service and company information in its registry to retrieve web services. This information cannot fully capture user's needs and may miss out on potential matches. Underlying functionality and semantics of web services need to be considered. In this paper, we explore semantics of web services using WSDL operation names and parameter names along with WordNet. We compute semantic similarity of web services and use this data to generate clusters. Then, we use a novel approach to represent the clusters and utilize that information to further predict similarity of any new web services. This approach has yielded good results and it can be efficiently used by web service search engines to retrieve similar or related web services.

Keywords: *Web Services, WordNet, Clustering, Classification*

1. Introduction

Web Services are widely popular and offer a bright promise for integrating business applications within or outside an organization. They are based on Service Oriented Architecture (SOA) [1] that provides loose coupling between software components via standard interfaces.

Web Services expose their interfaces using Web Service Description Language (WSDL) [2]. WSDL is an XML based language and hence platform independent. A typical WSDL file provides information such as web service description, operations that are offered by a web service, input and output parameters for each web service operation. Web Service providers use a central repository called UDDI (Universal Description, Discovery and Integration) [3] to advertise and publish their services. Web Service consumers use UDDI to discover services that suit their requirements and to obtain the service metadata needed to consume those services. Users that want to use a web service will utilize this metadata to query the web service using SOAP (Simple Object Access Protocol) [4]. SOAP is a network protocol for exchanging XML messages or data. Since SOAP is based on HTTP/HTTP-S, it can very likely get through network firewalls. The advantages of XML and SOAP give web services their maximum strength.

With web applications and portals getting complex and rich in functionality day after day, many users are interesting in finding similar web services. Users might want to compose two operations from different web services to obtain complex functionality. Also, users might be interested in looking at operations that take similar inputs and produce similar outputs. Let us say, web service A has an operation GetCityNameByZip that returns city name by zip code, web service B has an operation GetWeatherByCityName that returns weather by city name and web service C has an operation GetGeographicalLocationBasedOnZip that returns city name, longitude, latitude and altitude of a location by zip code. Operations from web services A and B are related, i.e., output from one operation can be used as an input to another. So, these operations can be composed to obtain weather by city name. Operations from web services A and C are similar. They take similar inputs. Outputs are also similar, i.e., output of operation from web service C is fine grained when compared to output of operation from web service A.

As more and more web services are developed, it is a challenge to find the right or relevant web services quickly and efficiently. Currently, UDDI supports keyword match just based on web service data entries in its registry. This might potentially miss out on some valid matches. For example, searching UDDI with keywords like zip code may not retrieve web service with postal code information.

Semantics of a web service in terms of the requirements and capabilities of a web service can be really helpful for efficient retrieval of web services. WSDL does not have support for semantic specifications. A lot of research is done on annotating web services through special markup languages, to attach semantics to a web service. Akkiraju et al. [5] proposed WSDL-S to annotate web services. Cardoso and Sheth [6] used DAML-S [7] annotations to compose multiple web services. Ganjisaffar et al. [8] used OWL-S [9] annotations to compute similarity between web services. But annotating all the available web services manually is a time consuming task and not feasible.

Some research has been done to extract semantics just based on WSDL. Normally the functionality or semantics of a web service can be inferred based on its description, operations along with parameters that these operations take. Dong et al. [10] built a web search engine called Woogle based on agglomerative clustering of WSDL descriptions, operations and parameters. Wu and Wu [11] provided a suite of similarity measures to assess the web service similarity. Kil et al. [12] proposed a flexible network model for matching web services.

The objective of our work is to cluster and predict similar web services using semantics of WSDL operations and parameters along with WordNet [13]. WordNet is a lexical database that groups words into synsets (synonym sets) and maintains semantic relations between these synsets. This work integrates ideas from [11] and [12] along with Hierarchical Clustering to innovatively predict similar web services.

Since there is no publicly available web services dataset, we evaluated our study using a set of WSDL files downloaded from the Internet. The general structure of our approach is as follows: first, we organized web service descriptions, operation names and parameter names from WSDL into three separate excel files respectively. We used popular natural language pre-processing techniques like Stop Words Removal and Stemming to remove unnecessary and irrelevant terms from the data. Then we use similarity measures from [11] along with WordNet to assess the similarity between web services. Once we obtain a similarity matrix of web services, we use Hierarchical Clustering [24, 25] to group or cluster related web services. One of the main contributions of this work is the representation of these clusters. We represent a cluster by a set of characteristic operations, i.e., for each web service in a cluster; take one characteristic operation that has maximum similarity to operations of other web services in the same cluster. This cluster representation is then used as a basis for predicting similarity of any new web services to the clusters using the nearest neighbor approach.

Our application has yielded good results and can be used as an add-on for any web service search engine for efficient web service matchmaking. If user has partially designed a web service or has discovered a web service and is interested in finding web services with similar operations, then our application can effectively find related services based on interface similarity of web service operations and their input and output parameters.

The remaining sections of this paper are organized as follows. Section 2 provides key information on similarity computation of web services. Section 3 presents data collection and pre-processing. In Section 4, we discuss WordNet based semantic similarity. It starts with an overview of WordNet, its organization and use for word sense disambiguation and explains similarity computation measures. Section 5 describes clustering of training set of web services using hierarchical clustering approach, cluster representation and prediction of similarity for web services in the test dataset. Section 6 discusses application setup and results. Conclusions are given in Section 7, followed by references.

2. Similarity of Web Services

A web service is described by WSDL file and is characterized by a name, description, and a set of operations that take input parameters and return output parameters. We used this WSDL information for computing similarity of web services. Specifically, we employed interface similarity assessment suggested by Wu and Wu [11]. Similarity between web services is computed by identifying the pair-wise correspondence of their operations that maximizes the sum total of the matching scores of the individual pairs. Similarity between web services S_1 with m operations and S_2 with n operations is given by the following formula:

$$Sim_{Interfaces}(S_1, S_2) = Max\sum_{i=1}^{m}\sum_{j=1}^{n} Sim_{Operation}(O_{1i}, O_{2j}) \times x_{ij} \quad (1)$$

$$x_{ij} = \begin{cases} 1 \; combine \; O_{1i} \;\; with \quad O_{2j} \\ 0 \qquad\qquad\qquad\qquad else \end{cases} \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \, i = 1,2,..m \quad \sum_{i=1}^{m} x_{ij} = 1, \, j = 1,2,..n \tag{3}$$

where O_{1i} represents an operation from web service S_1 and O_{2j} represents an operation from web service S_2. X_{ij} indicates the weight and it is set to 1, while matching operation O_{1i} with operation O_{2j}.

To illustrate interface similarity, let us consider the example shown in Figure 2.1. Here web service 1 has 2 operations, operation 11 and operation 12. Web service 2 has 3 operations, operation 21, operation 22 and operation 23. We match operation 11 to operation 21, operation 22 and operation 23 and pick the matching that gives maximum similarity. Similarly, we match operation 12 to operations in web service 2. Then we sum up the maximum similarity values from both these matching pairs to give the similarity between web services.

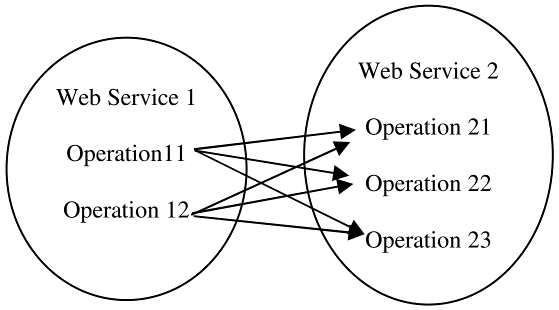

Figure 2.1. Matching of web service operations.

Similarly, the similarity of operation pairs is calculated by identifying the pair-wise correspondence of their input/output parameter lists that maximizes the sum total of the matching scores of the input/output individual pairs. Similarity between web service operation O_1 with m input parameters and u outputs; and web service operation O_2 with n input parameters and v outputs can be given by the following formula:

$$Sim_{Operations}(O_1, O_2) \;=\; Max \sum_{i=1}^{m} \sum_{j=1}^{n} Sim_{Input}(I_{1i}, I_{2j}) \times x_{ij}$$

$$+ \, Max \sum_{i=1}^{u} \sum_{j=1}^{v} Sim_{Output}\,(P_{1i}, P_{2j}) \times y_{ij} \tag{4}$$

$$x_{ij} = \begin{cases} 1 \; combine \; I_{1i} \; with \; I_{2j} \\ 0 \qquad\qquad\qquad else \end{cases} \tag{5}$$

$$y_{ij} = \begin{cases} 1 \; combine \; P_{1i} \; with \; P_{2j} \\ 0 \qquad\qquad\qquad else \end{cases} \tag{6}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \, i = 1,2,..m \quad \sum_{i=1}^{v} y_{ij} = 1, \, j = 1,2,..u \tag{7}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \, i = 1,2,..n \quad \sum_{i=1}^{u} y_{ij} = 1, \, j = 1,2,..v \tag{8}$$

Here I_{1i} and I_{2j} stand for input parameters of web service operation O_1 and web service operation O_2 respectively. P_{1i} and P_{2j} stand for outputs of web service operation O_1 and web service operation O_2 respectively. X_{ij} indicates the weight and it is set to 1 while matching input parameters I_{1i} with I_{2j}, and Y_{ij} is the weight and it is set to 1 while matching outputs P_{1i} with P_{2j}.

Parameter name similarity is computed by the lexical similarity of their names. Lexical similarity between words indicates how closely their underlying concepts are related. Similarity between Input parameter I_1 of Operation O_1, belonging to web service S_1 and Input parameter I_2 of Operation O_2, belonging to web service S_2 can be given by the following formula:

$$Sim_{Parameters}(I_1, I_2) = Sim_{Lexical}(I_1.Name, I_2.Name) \tag{9}$$

Similarly, lexical similarity can be computed for outputs of operations O_1 and O_2.

Since number of operations and in turn its parameters are not constant across web services, we normalized the similarity measures. For example, let us say web service A has 3 operations and web service B has 5 operations. Similarity between web services is computed according to the formula for interface similarity and then normalized by dividing by 3 (number of operations in A). This is done to normalize the effect of number of operations across all web services. Similarly, we normalized input and output parameters of operations.

Next two sectons explain how web service data was collected and how WordNet was used along with the formulae mentioned in this section for similarity computations.

3. Data Pre-processing

There is no publicly available web services dataset. So, we downloaded a set of web services in 5 domains from xmethods.net website. WSDL data from these web services is then organized into 3 excel files, one for web service name and description, one for web service operation names and another for web service input and output parameter names. Tables 3.1, 3.2 and 3.3 show the format of Excel files. Web service ID in these tables represents a unique numeric identifier for each web service. This is similar to ID column in a database table represents a unique numeric identifier for each web service.

Table 3.1: Format of Excel file with web service descriptions**.**

WS ID	Name	Text Description	WSDL Name	URL
1	US Zip Validator	Zip code validator	USZip	http://www.webservicemart.com/uszip.asmx
2	Phone Number Verification service	Phone number verifier	Phone 3T	http://www.webservicemart.com/phone3t.asmx

Table 3.2: Format of Excel file with web service operations**.**

Web service ID	Operation ID	Name
1	1	ValidateZip
2	1	PhoneVerify

Operation ID in Tables 3.2 and 3.3 represents a numeric identifier for each web service operation. Direction in Table 3.3 indicates whether it is an input parameter 'I' or an output parameter 'O'.

Table 3.3: Format of Excel file with web service operation parameters**.**

Web service ID	Operation ID	Parameter Name	Direction
1	1	ZipCode	I
1	1	ValidateZipResult	O
2	1	PhoneNumber	I
2	1	PhoneVerifyResult	O

We use parameter flattening similar to that described in [12] when we come across complex data structures for input parameters. For example, if the input parameter of web service operation is a data structure named "PhoneVerify" that contains Phone Number field. Then we take Phone Number as input parameter instead of PhoneVerify.

The 3 Excel files are then fed as inputs to web service pre-processing module. This module is a third party software downloaded from [21]. It internally removes Stop Words, uses stemming for preprocessing the data.

3.1 Stop Words Removal

A document is a vector or bag of words or terms. Stop Words are a list of words that are insignificant and can be easily removed from a document or a sentence or phrase. To achieve this, program is presented with a list of stop words that can be removed. Examples of stop words can be a, an, about, by, get etc. For a web service operation like GetWeatherByZip, significant words are 'Weather' and 'Zip'. 'Get' and 'By' do not convey a lot of meaning and can be safely removed.

3.2 Stemming

Normally terms that originate from a common root or stem have similar meanings. For example, the following words have similar meanings.

- INTERSECT
- INTERSECTED
- INTERSECTING
- INTERSECTION
- INTERSECTIONS

Key idea is to represent such related term groups using a single term, here INTERSECT by removing various suffixes like –ED, -ING, -ION, -IONS. This process of representing a document with unique terms is called Stemming. Stemming reduces the amount and complexity of the data while retrieving information. It is widely used in search engines for indexing and other natural language processing problems [14].

Porter Stemming Algorithm [15, 16] is one of the most popular stemming algorithms. The basic idea is to take a list of suffixes and the criterion during which a suffix can be removed. It is simple, efficient and fast.

Once WSDL data is pre-processed using stemming and stop words removal, WordNet is used in similarity computation of web services. More details on WordNet and similarity computation can be found in the next section.

4. WordNet Based Semantic Similarity

This section provides an overview of WordNet and how WordNet is used for computing semantic similarity of web services.

4.1 WordNet

WordNet is an electronic lexical database [13, 17] that uses word senses to determine underlying semantics. It differs from the traditional dictionary in that, it is organized by meaning, so words in close proximity are related. WordNet entries are organized as mapping of words and its concepts.

Multiple synonym words (synonym set or synset) can represent a single concept. For example, {Comb, Brush} are synonyms. Also, a single word can represent multiple concepts (polysemy). For example, Brush can mean Sweep, Clash, Encounter etc.

4.2 Organization of WordNet

WordNet organizes synsets of nouns and verbs as hypernyms and hyponyms [17]. For example, animal is a hypernym of cow and cow is a hyponym of animal. Beyond this hypernym /hyponym relation, WordNet also provides relations such as Meronymy/holonymy (part/whole), is-made-of, is-an-attribute-of etc. Also, each concept is quantified by a short description called "gloss". All these relations result in a large interconnection network. The logical structure of WordNet is shown as in Figure 4.1.

4.3 Word Sense Disambiguation

In general, a word can have multiple meanings or make different senses in the context where it is used. The object of word sense disambiguation is to determine the correct sense of a word. Many algorithms have been developed for this purpose. In the present work, we use Lesk algorithm [18]. The following steps can summarize it:

1. Retrieve from Machine Readable Dictionary all sense definitions of the words to be disambiguated.
2. Determine the definition overlap for all possible sense combinations
3. Choose senses that lead to highest overlap

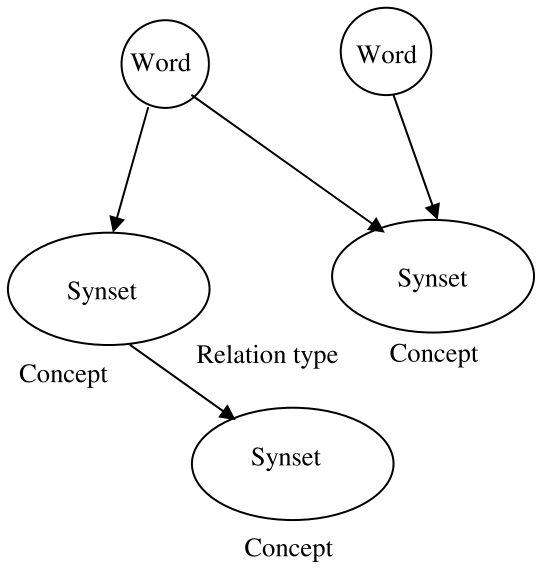

Figure 4.1. The Logical structure of WordNet.

For example, to disambiguate PINE CONE using the example introduced by Christiane Fellbaum (reproduced from [13] with author's permission),

PINE has these 2 senses

1. Kinds of evergreen tree with needle-shaped leaves
2. Waste away through sorrow or illness

CONE has these 3 senses

1. Solid body that narrows to a point
2. Something of this shape whether solid or hollow
3. Fruit of certain evergreen trees

Determine all possible combination of senses and their scores as:

$$\text{Pine\#1} \cap \text{Cone\#1} = 0$$
$$\text{Pine\#2} \cap \text{Cone\#1} = 0$$
$$\text{Pine\#1} \cap \text{Cone\#2} = 1$$
$$\text{Pine\#2} \cap \text{Cone\#2} = 0$$
$$\text{Pine\#1} \cap \text{Cone\#3} = 2$$
$$\text{Pine\#2} \cap \text{Cone\#3} = 0$$

Then, we can conclude that Pine1 and Cone3 are highly related.

4.4 Determine Word Sense Using WordNet

One idea to determine word sense in a given context by using WordNet is to look for all the paths from the context to the word and take the shortest one as the right sense. A detailed explanation of the adapted Lesk algorithm using WordNet can be found in [19, 20]. To elucidate this, consider the following example that is reproduced from [21] with author's written permission:

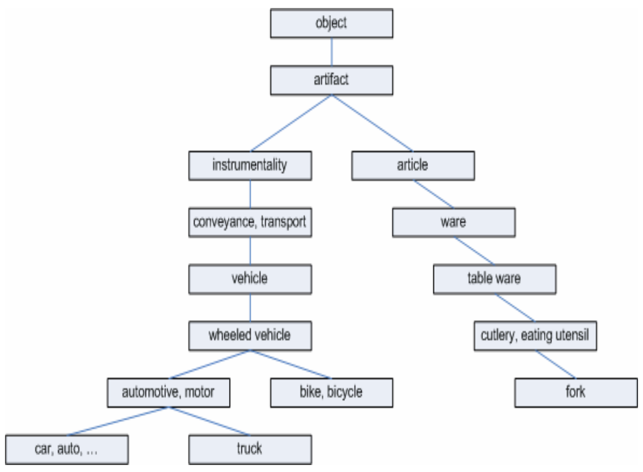

Figure 4.2. Illustration of WordNet structure.

In Figure 4.2, we observe that the length between car and auto is 1, car and truck is 3, car and bicycle is 4, and car and fork is 12.

4.5 Measure Words Similarity Using WordNet

There are six measures to obtain similarity between words using WordNet. As described in [22], three of the six measures are based on the information content of the Least Common Subsumer (LCS) of concepts. Information content is a measure of the specificity of a concept, and the LCS of concepts A and B is the most specific concept that is an ancestor of both A and B. Three similarity measures are based on path lengths between a pair of concepts.

In the present study, we use Wu & Palmer similarity measure [23]. According to this measure, similarity between two concepts is the path length to the root node from the least common subsumer of the two concepts. Least common subsumer is the most specific concept that two concepts share as an ancestor.

In Figure 4.2, the LCS of {car, auto} and {truck} is {automotive, motor vehicle}, since the {automotive, motor vehicle} is more specific than the common subsumer {wheeled vehicle}.

For synsets S1 and S2, it can be given by the formula: Similarity Score = 2*depth (LCS) / (depth (S1) + depth (S2)).

Similarity score is in the range $0 < \text{score} <= 1$. It can never be zero because depth of the LCS is never zero (the depth of the root of taxonomy is one). The score is one if the two input synsets are the same.

4.6 WordNet Based Similarity of Web Services

As we know from Section 2, parameter similarity is determined by the lexical similarity of parameter names. We use 3rd party similarity computation software module from [21] along with WordNet for this purpose. Figure 4.3 illustrates the flowchart for the pre-processing and similarity computation that we used in this study. Once parameter similarity is obtained, then we compute operation similarity and web service similarity using the formulae mentioned in Section 2. Similarity matrix for web services thus computed is used as basis for clustering or grouping similar web services.

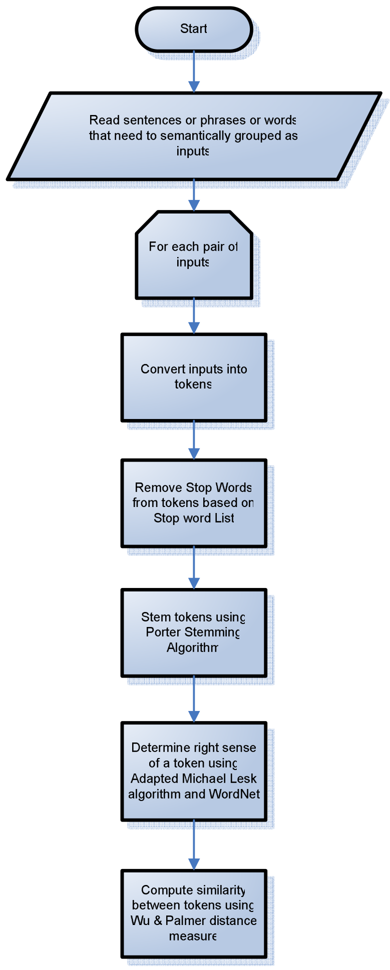
Start
Read sentences or phrases or words that need to semantically grouped as inputs
For each pair of inputs
Convert inputs into tokens
Remove Stop Words from tokens based on Stop word List
Stem tokens using Porter Stemming Algorithm
Determine right sense of a token using Adapted Michael Lesk algorithm and WordNet
Compute similarity between tokens using Wu & Palmer distance measure

Figure 4.3. WordNet based Similarity computation.

5. Clustering and Classification of Web Services

Clustering is the process of partitioning data into groups of similar objects or clusters. It is an unsupervised learning technique that is widely used in Artificial Intelligence, Data mining etc. It aims to discover patterns taking into account the entire data. There are no pre-defined conditional and decision variables. Members within a cluster are more similar or related to each other and different from members of other clusters.

Clustering is used in our work to identify related or similar web services. There are many different approaches to clustering. In the present work, we use agglomerative or bottom-up Hierarchical Clustering method [24, 25]. In this method, initially each web service is treated as belonging to a cluster. Then we use similarity matrix of web services in the training dataset to determine the nearest neighbors. Nearest clusters are then merged into one cluster. This process is repeated and in the end all the web services merge to a single cluster.

We used adapted version of Hierarchical Clustering program available at [26] for clustering web services. We employed a couple of approaches to use this clustering information to predict similarity of web services in the test dataset.

5.1 Classification of web services

First, we tried to generate rules from the generated clusters using rule-based algorithm BLEM2 [27]. Web service operation name and parameter names are taken as condition attributes and cluster number as the decision attribute. Since the number of parameters is not constant across all web services, we have to treat some of the attributes as don't cares in the training set. To illustrate this, let us take web service 1 with operation A and web service 2 with operation B. Also, assume that operation A has 5 parameters (4 input parameters and 1 output parameter) and operation B has 3 parameters (2 input parameters and 1 output parameter). We take the maximum number of parameters of all web service operations, here 5, as condition attributes. So, when we represent operation B we take its 3 parameters as condition attributes and treat rest of the columns, as don't cares. As expected, this kind of training set would cause problems for most classifier learning algorithms, because too many don't cares entries are artificially inserted into the training set.

Since classification from rule-based classifier did not yield good results, we tried a different approach to predict similarity of web services in the test dataset. Initially, we represent each cluster by a set of characteristic operations of its member web services. To obtain this set of characteristic operations, i.e., for each web service in a cluster; take one characteristic operation that has maximum similarity to operations of other web services in the same cluster. For example, say a cluster has grouped two web services "Electronic Directory Assistance" and "Weather by City". "Electronic Directory Assistance" has 3 operations namely ResidentialLookup, BusinessLookup, LookupByAddress. "Weather by City" has 2 operations namely GetWeatherByCity, GetWeatherByCityXml. If similarity between ResidentialLookup and GetWeatherByCity is the highest, then we represent the cluster by these 2 most similar operations (1 operation per web service). If a cluster has only one web service, then we take one web service operation that is very dissimilar to operations of web services in other clusters. This cluster representation is then used as a basis for predicting similarity of any new web services to the clusters using the nearest neighbor approach. To elucidate, we compute interface similarity between operations of each test web service and characteristic operations of clusters and find the nearest cluster. This approach yielded good results.

6. Experimental Results

To setup the application, first WordNet needs to be installed. WordNet can be downloaded from http://wordnet.princeton.edu/obtain website. We use C#. NET to implement our application.

There is no publicly available web services dataset on the Internet. So, we have downloaded a small set of web services from xmethods.net website. It was tedious to dig through WSDL files and organize data into three excel files as mentioned in Section 3. Table 6.1 represents our training set of web services. Our clustering algorithm generated 5 clusters based on interface similarity of these web services. These clusters are represented in Figure 6.1.

We came up with a novel idea of representing the clusters by the most similar operations (1 operation per web service) of web services in that cluster. If there is only one web service in a cluster, then we select an operation that is very dissimilar to operations in other clusters. Table 6.2 lists the characteristic operations for clusters represented in Figure 6.1.

Table 6.1: Training dataset.

Web service ID	Name	Text Description
1	US Zip Validator	Zip code validator
2	Phone Number Verification service	Phone number verifier
3	StrikeIron Foreign Exchange Rates	Current and historical foreign exchange rates
4	U.S. Yellow Pages	Access to yellow pages listings for 17 million U.S. businesses
5	City and State by ZIP	Finds the City and State for a given ZIP code
6	Electronic Directory Assistance	white pages
7	Weather by City	Enter a city name and instantly receive the current day's weather report.
8	Forecast by ZIP Code	Enter a U.S. ZIP code and instantly receive the 10-day

		weather forecast.

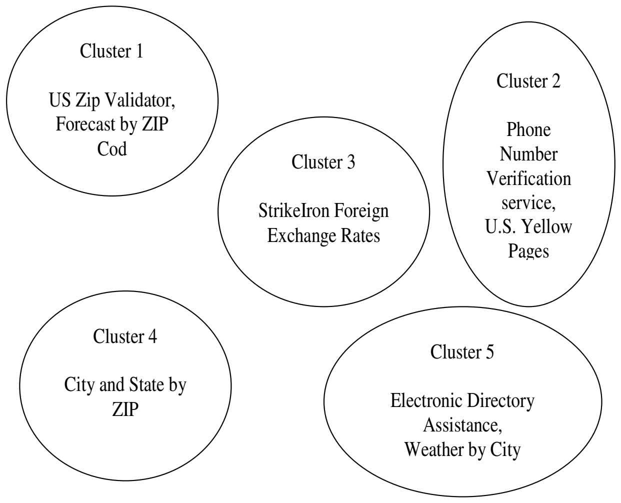
Cluster 1
US Zip Validator, Forecast by ZIP Cod
Cluster 3
StrikeIron Foreign Exchange Rates
Cluster 2
Phone Number Verification service, U.S. Yellow Pages
Cluster 4
City and State by ZIP
Cluster 5
Electronic Directory Assistance, Weather by City

Figure 6.1. Clusters obtained from training data.

Table 6.2: Clusters and their characteristic operations.

Cluster #	Operation	Web Service
1	ValidateZip	US Zip Validator
1	GetForecastByZip	Forecast by ZIP Code
2	PhoneVerify	Phone Number Verification service
2	ReversePhoneLookup	U.S. Yellow Pages
3	GetAllLatestRatesToUSD	StrikeIron Foreign Exchange Rates
4	GetCityStateByZip	City and State by ZIP
5	GetWeatherByCityXml	Weather by City
5	ResidentialLookup	Electronic Directory Assistance

We downloaded a set of web services to test our idea. We organized the WSDL data into 3 files as mentioned in Section 3. Similarly, we created another set of 3 files for representing the clusters. Then our program computed interface similarity between clusters and the test web services and found closest cluster to each test web service based on nearest neighbor approach. Table 6.3 represents test web services, their predicted nearest clusters, cluster members and the actual clusters. Actual cluster values are given based on semantic similarity of web service descriptions. Our approach yielded good results and accuracy is 70%.

Table 6.3: Test web services and nearest clusters.

Test Web Service	Nearest cluster #	Web Services in Cluster	Actual Cluster #
FastWeather	1	US Zip Validator, Forecast by ZIP Code	1
Currency Convertor	3	StrikeIron Foreign Exchange Rates	3
StrikeIron Reverse Phone Residential Intel	5	Weather by City, Electronic Directory Assistance	5
DOTS Yellow Pages	3	StrikeIron Foreign Exchange Rates	2
PHONEval	2	Phone Number Verification service	2
Levelsoft GeoServices Global Weather Service	2	Phone Number Verification service	1
Zip Codes	1	US Zip Validator, Forecast by ZIP Code	1
StrikeIron ZIP Code Information	4	City and State by ZIP	4

7. Conclusions

We developed an application for effectively finding similar or related web services. It can be used as an add-on to any web service search engine with UDDI repository. We used semantics of WSDL along with WordNet to compute similarity between various web services. For web services in the training set, we computed similarities and clustered the data using Hierarchical Clustering. Next, we represented each cluster by a set of characteristic operations. Then we used these cluster representations to evaluate similarity of any new web services using nearest neighbor approach. This has yielded good results and accuracy is 70% for our test data. At present, there is no publicly available web services dataset. More work can be done in future with evaluating our approach, when such a dataset becomes available. Also, we would like to compare our approach with other dimensionality reduction techniques while choosing the key operations of a cluster.

8. References

[1] Barry, Douglas K. (2003). *Web Services and Service-Oriented Architectures: The Savvy Manager's Guide*. San Francisco: Morgan Kaufmann Publishers. ISBN 1-55860-906-7.

[2] Christensen, E., F. Curbera, G. Meredith, and S. Weerawarana (2001), "Web Services Description Language (WSDL) 1.1", W3C Recommendation, 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315 .

[3] Clement, L. et al. (Ed.) (2004), "UDDI Version 3.0.2", http://uddi.org/pubs/uddiv3.0.2-20041019.htm .

[4] Mitra, N. (Ed.). (2003) "SOAP Version 1.2 Part 0: Primer", W3C Recommendation, 2003, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[5] Akkiraju, R. et al. (2005), "Web Service Semantics: WSDL-S", W3C member submission, 2005; www.w3.org/SubmissionWSDL-S/ .

[6] Cardoso, J. and A. Sheth (2003), "Semantic e-Workflow Composition", *Journal of Intelligent Information Systems*, vol. 21, no. 3, pp. 191--225, November 2003.

[7] Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., and Zeng, H. (2001), "DAML-S: Semantic Markup for Web Services", *Proceedings of the International Semantic Web Working Symposium* (SWWS) (pp. 39–54). Stanford University, California.

[8] Ganjisaffar, Y., Hassan Abolhassani, Mahmood Neshati, Mohsen Jamali (2006), "A Similarity Measure for OWL-S Annotated Web Services", pp. 621-624, *2006 IEEE/WIC/ACM International Conference on Web Intelligence*.

[9] Martin et al. (2004), OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/

[10] Dong, X., et. al. (2004): Similarity Search for Web Services, *VLDB Conference. Toronto, Canada*, www.vldb.org/conf/2004/RS10P1.PDF.

[11] Wu, Jian and Wu, Zhaohui (2005), "Similarity-based Web service matchmaking", *2005 IEEE International Conference on Services Computing*, Vol. 1, pp. 287-294.

[12] Kil, Hyunyoung, Seog-Chan Oh and Dongwon Lee (2006), "On the Topological Landscape of Web Services Matchmaking", *VLDB Int'l Workshop on Semantic Matchmaking and Resource Retrieval (SMR06)*, Seoul, Korea.

[13] Fellbaum, Christiane (1998), The WordNet book, WordNet: An Electronic Lexical Database.

[14] Stemming, http://en.wikipedia.org/wiki/Stemming

[15] van Rijsbergen, C.J., S.E. Robertson and M.F. Porter (1980), *New models in probabilistic information retrieval.* London: British Library. (British Library Research and Development Report, no. 5587).

[16] Porter, M.F. (1980), An algorithm for suffix stripping, *Program*, **14** (3) pp 130–137.

[17] Fellbaum, Christiane (2007), WordNet: Connecting words and concepts, http://colab.cim3.net/file/work/SICoP/2007-02-06/WordNet02062007.ppt

[18] Pedersen, Ted (2005), Word sense disambiguation, http://www.d.umn.edu/~tpederse/WSDTutorial.html

[19] Banerjee, Satanjeev and Pedersen, Ted (2002), "An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet", Lecture Notes In Computer Science Vol. 2276, *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pp. 136-145, 2002. Available online at: http://www.d.umn.edu/~tpederse/Pubs/cicling2002-b.ps

[20] Banerjee, S. & T. Pedersen (2003), "Extended gloss overlap as a measure of semantic relatedness", *Proceedings of the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, 9--15 August, 2003*, pp. 805-810. Available online at http://www.d.umn.edu/~tpederse/Pubs/ijcai03.pdf

[21] Simpson, Troy and Dao, Thanh (2005), WordNet-based semantic similarity measurement, http://www.codeproject.com/cs/library/semanticsimilaritywordnet.asp

[22] Pedersen, Ted, Siddharth Patwardhan & Jason Michelizzi (2004), WordNet::Similarity -- Measuring the relatedness of concepts, *Demonstrations of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Boston, Mass., 2--7 May 2004, pp. 267--270. Available online at http://www.cs.utah.edu/~sidd/papers/PedersenPM04b.pdf

[23] Wu and Palmer similarity measure, http://search.cpan.org/src/SID/WordNet-Similarity-1.04/lib/WordNet/Similarity/wup.pm

[24] Hierarchical Clustering, http://www.resample.com/xlminer/help/HClst/HClst_intro.htm

[25] Murtagh, F. (1985), Multidimensional Clustering Algorithms, Physica-Verlag.

[26] Murtagh, F. (2002), Multivariate Data Analysis software and resources. http://astro.u-strasbg.fr/~fmurtagh/mda-sw

[27] Chan, C.-C. and S. Santhosh (2003), "'Blem2: Learning Bayes' Rules From Examples Using Rough Sets," *NAFIPS 2003, 22nd Int. conf. of the North American Fuzzy Information Processing Society*, July 24 – 26, 2003, Chicago, Illinois, pp 187-190.