

Clustering-based Taxonomy Construction for Semantic Web Services

Abstract—Taxonomy construction for web services has several benefits. Its main benefit is that it reduces the amount of time needed for discovering services. In this paper, we propose an algorithm for semi-automatically creation of service taxonomies from services registered in some registries. This algorithm is based on agglomerative clustering of services and most of the steps of the algorithm are done automatically. In fact, assignment of names to the created clusters is the only semi-automatic part of the algorithm. We implemented our method and the experimental results of applying it on a sample test collection show the effectiveness of the idea.

I. INTRODUCTION

Web Services are loosely coupled software components that are published, located and invoked across the web. Recently, web services have gained an increasing popularity. Today, not only web services are used in dynamic business-to-business interactions, but they are also used in the area of business-to-consumer or even peer-to-peer interactions.

Universal Description, Discovery, and Integration (UDDI) is one of the main industry standards for web service registries. The search mechanisms provided by most UDDI registries are based either on keyword search or browsing through predefined hierarchical business categories [1]. Keyword search is known to have the drawback that users may be confronted with many irrelevant search results due to such issues as synonyms and word ambiguity. Consequently, users may have to spend a lot of time, browsing through the results to identify the web services that most closely meet their demands.

The Semantic Web approaches to web services give us the ability to describe the semantics of web services and their capabilities in a formal and machine-processable manner. A lot of researches such as [2], [3], [4] are done for improving the precision of service discovery techniques using semantic annotations that are provided for web services. However, in most of these techniques, the service requester must specify the outputs he requires and the inputs he can provide for the service. But the problem is that most of the service requesters do not have a clear vision of their request, and therefore can not precisely describe their requests by specifying the input/output parameters.

If the web services are categorized well, browsing through predefined categories may be a better alternative for service discovery. Currently, there are some standard service categorization schemes, such as the North American Industry Classification System (NAICS) [5] and the United Nations Standard Products and Services Code (UNSPSC) [6]. But the problem with these standards is that they are developed for categorizing general services and not specifically for web

services. The result is that when using these standards for categorizing web services existing in a service registry, there would be a lot of categories that contain no service and on the other hand some categories may contain a lot of services. Therefore using these standards for categorizing web services would not lead to good results.

In this paper, we introduce a clustering based approach for semi-automatically taxonomy construction for web services that are registered in a service repository. Section II, describes how hierarchical clustering algorithms can be used for automatic taxonomy construction and describes application of these algorithms for the web services domain. Sections III & IV introduce two clustering methods that use two different similarity measures for determining the similarity of services using their corresponding service descriptions. Experimental results of our proposed algorithms are presented in section V. Related works are explored in section VI and finally a conclusion is given in the last section.

II. CLUSTERING-BASED TAXONOMY CONSTRUCTION

By clustering we mean the grouping of services into classes of similar services. A cluster is a collection of services that are similar to one another and dissimilar to services in the other clusters [7]. Clustering algorithms seek to segment the entire data set into relatively homogeneous subgroups or clusters such that similarity within a group is very high while the similarity to services in other groups is very low. In other words, as shown in Fig. 1, clustering algorithms seek to construct clusters of records such that the between-cluster variation (BCV) is large compared to the within-cluster variation (WCV) This is somewhat analogous to the concept behind analysis of variance (Fig. 1).

Clustering algorithms are either *hierarchical* or *non-hierarchical*. In hierarchical clustering, a tree-like cluster structure is created through recursive partitioning (divisive methods) or combining (agglomerative) of existing clusters. Agglomerative clustering methods initialize each object to be a tiny cluster of its own. Then, in succeeding steps, the two closest clusters are aggregated into a new combined cluster. In this way, the number of clusters in the data set is reduced by one at each step (Fig 2). Eventually, all web services are combined into a single huge cluster. We can use several criteria for determining distance between arbitrary clusters A and B such as *single linkage*, *complete linkage* or *average linkage*. Single linkage is based on the minimum distance between any object in cluster A and any object in cluster B. Complete linkage is based on the maximum distance between any object in cluster

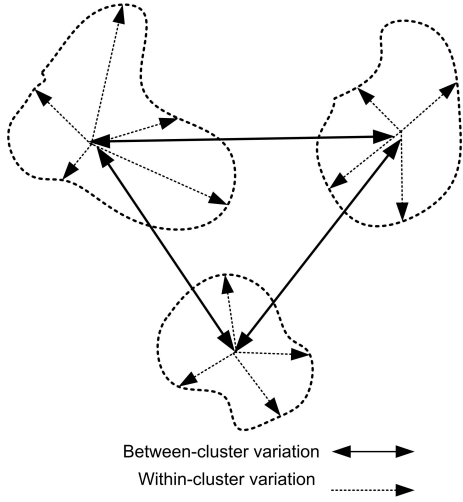


Fig. 1. Clusters should have small within-cluster variation compared to the between cluster variation

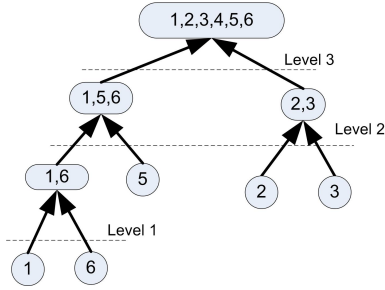


Fig. 2. Bottom up Clustering

A and any object in cluster B. Average linkage is designed to reduce the dependency of the cluster-linkage criterion on extreme values, such as the most similar or dissimilar objects. The most widely used nonhierarchical clustering methods are k -Means and k -Medoids. These methods require the user to specify the number of clusters, k , prior to a cluster analysis.

Because of producing a tree-like cluster structure, hierarchical clustering algorithms can be used for taxonomy construction. In this paper, we describe how a service taxonomy can be built using a variation of the agglomerative clustering method. In our method, services are represented as vectors (in section III, it is showed how it is possible to represent services as vectors). After creating the hierarchical clusters, the Within Cluster Variation (WCV) is calculated for each cluster as follows:

$$WCV_{C_i} = \frac{\sum_{S_j \in C_i} (1 - \text{CosineSim}(S_j, \vec{C}))}{|C_i|}$$

Where $|C_i|$ is the number of services that are located in cluster C_i and \vec{C} is the *centroid* of this cluster. Having n data points $(a_1, b_1, \dots, c_1), (a_2, b_2, \dots, c_2), \dots, (a_n, b_n, \dots, c_n)$, the centroid of these points is the center of gravity of these points and is located at point $(\sum a_i/n, \sum b_i/n, \dots, \sum c_i/n)$. $\text{CosineSim}(\vec{S}_j, \vec{C})$ is the cosine similarity of vectors of S_j

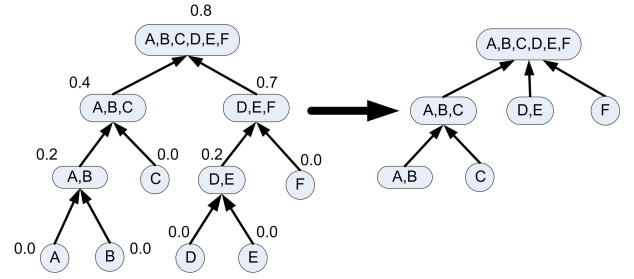


Fig. 3. Construction of Taxonomy from hierarchical clusters

and \vec{C} :

$$\text{CosineSim}(\vec{S}_j, \vec{C}) = \frac{\vec{S}_j \cdot \vec{C}}{|\vec{S}_j| \times |\vec{C}|}$$

The Within Cluster Variation is a measure of the quality of the cluster. The smaller the value of the WCV the more similar the clusters are. Left side of Fig. 3 shows the result of an agglomerative clustering method. In this figure the WCV value of each cluster is showed near that cluster. Our purpose is to create a good taxonomy based on this structure. As you can see in the figure the $\{A, B\}$ cluster has a rather low WCV value. This means that services A and B are very similar to each other. Therefore why should we divide this cluster into separate clusters A and B in our taxonomy. In fact, whenever the WCV value of a cluster is lower than a threshold value T_α it is not needed to further divide this cluster into smaller clusters. Therefore after creating the hierarchical clusters, we traverse the structure using a top-down fashion and whenever we reach to a cluster with a WCV value lower than T_α all of the sub trees that are children of this cluster are deleted from the structure. Applying this algorithm with $T_\alpha = 0.3$ on the left hand side structure of Fig. 3 will result in deleting clusters $\{A\}$, $\{B\}$, $\{D\}$, $\{E\}$ from the structure.

On the other hand, when the WCV value of a cluster is higher than a threshold value T_β , services in this cluster are not so much similar to each other and therefore clustering them into a group is not good enough and it is better to remove these clusters from the structure. The root cluster of the structure which contains all of the services is an exception of this algorithm and it is preserved in the structure without considering its WCV value. Applying this algorithm with $T_\beta = 0.6$ on the left hand side structure of Fig. 3 will result in deleting the $\{D, E, F\}$ cluster from the structure. After finishing this, it may happen that some of the clusters do not have parent clusters. In these cases the root cluster would be considered as their parent.

The right hand side of Fig. 3 shows the final structure that is obtained by applying the above steps on the algorithm. Clusters of this structure contain services that are similar to each other and therefore it can be used as a taxonomy of services.

Cluster Naming

An important part of taxonomy construction algorithms is choosing relevant names for final clusters. For naming clusters,

Cluster Services	Suggested Naming Keywords
Book information finder, ISBN Finder, Get Book ISBN, ISBN Book Search	ISBN, Book
Get Birth Place, Get Birth Date	Born, Person
Get Therapist Available Schedule, Get Therapy Price, Make Therapy Schedule	Therapist , Therapy
Yahoo Stock Quote, Stock Quote Company, Stock Quote	Symbol, Quote

TABLE I
SAMPLE SUGGESTIONS FOR NAMING CLUSTERS

we used a semi-automatic approach. In our approach, terms of descriptions of services in each cluster are extracted and using the tf-idf [8] weighting method the most weighted terms are suggested to a human expert for using them in naming the clusters. Table I shows some of the proposed keywords for naming some of the clusters that are created after applying our taxonomy construction technique on the test collection which is introduced in section V.

Detecting outliers

Sometimes it happens that although the WCV value of a cluster is within the threshold values T_α and T_β , a few services in the cluster are far from the centroid of the cluster. These kind of services are called *outliers* and if a cluster contains such services it is not a good cluster. *Z-score standardization*, is a very widespread method in the world of statistical analysis for finding outliers. It works by taking the distance of the service to the centroid and scaling this distance by the standard deviation of the distances of other services to the centroid of the cluster. According to Chebyshev theorem [9] almost all the observations in a data set will have Z-score less than 3 in absolute value. Therefore, the services with Z-score greater than 3 or smaller than -3 will be outliers.

Clustering Web Services

Clustering methods described in the previous section, use the degree of similarity of services to find out whether they should be grouped in the same cluster or not. Therefore a similarity measure is needed for clustering web services.

OWL-S [10] is a proposed language for describing semantic metadata about web services, which is based on the OWL [11] ontology language. The OWL-S ontology is organized in three modules: the *Service Profile* module describes the functionality of the service; the *Service Model* module describes how it does it; and the *Service Grounding* module describes how to access the service. The semantics of inputs and outputs of a service are specified in the service profile section by mapping each input/output parameter to a concept which is defined in an OWL ontology.

In the next two sections, we introduce two clustering methods that use two different similarity measures for determining the similarity of services using their corresponding OWL-S annotations.

III. ATTRIBUTE-BASED CLUSTERING

Each web service has a number of features or attributes that describe its behavior. Using these attributes, we can find similar services (i.e. those services with similar attributes). Input/output parameters and textual descriptions of services are good attribute candidates. For example, services that receive/generate information about a `Hotel` as their inputs/outputs parameters, with a high probability must be clustered in the same cluster.

In the attribute-based clustering method, an *attribute vector* is created for each of the services. Services can be represented as vectors $\vec{s}_k = (a_{1k}, a_{2k}, \dots, a_{nk})$ where a_{ik} is the weight of attribute i in the description of service s_k . If a_{ik} is an input/output parameter then its weight is calculated as follows:

$$a_{ik} = \begin{cases} 1/f_i, & \text{if } s_k \text{ contains } a_k \\ 0, & \text{otherwise} \end{cases}$$

Where f_i is the frequency of this attribute in the service collection. Attributes that are very frequent in the collection of services are not good distinguishing attributes. Therefore their weight must be reduced in the attribute vector. This is why attribute weights are divided by the frequency of their occurrences.

On the other hand, the *tf-idf* weighting method can be used for determining weight of textual attributes. According to this weighting method, the importance of a term i in service description j is:

$$a_{ij} = tf_{ij} \times idf_i = tf_{ij} \times \log\left(\frac{N}{n_i}\right)$$

where tf_{ij} is the frequency of term i in service description j ; N is the number of descriptions in the collection; and n_i is the number of the descriptions in the collection that contain term i . Terms are extracted from textual parts of the service description after removing common words and extracting the base form¹ of them using WordNet [12].

Using terms as attributes of services has also some problems. For example, although terms `Book` and `Reserve` in services `Book Hotel` and `Reserve Hotel` has the same meaning in the vector space model they are represented as different terms. For handling this problem, we used WordNet for finding synonyms of terms. Before adding a new term to the terms vector, we first check whether one of its synonyms

¹For example *Reservation* would be converted to *Reserve*

is currently available in this vector and only if non of its synonyms is in the terms vector this term is added.

As a very simple example suppose the registry contains only two services, namely S_1 and S_2 . Service S_1 receives a person #SSN, returns his/her #Address and its description is “This service receives the SSN of a person and returns his/her address”. On the other hand, service S_2 receives the #SSN of a person, returns his/her #BirthDate and its description is “This service receives the SSN of a person and returns his/her birth date”. The extracted attributes would be {#SSN, #Address, #BirthDate, service, receive, SSN, person, return, address, birth, date}, where *SSN, Address, BirthDate* are extracted from input/output parameters and the other attributes are extracted from the textual description of these services. Note that some of the words such as *this* and *the* are very common in English language and therefore they are not good candidates for being attributes. These kinds of words are ignored when extracting attributes using a list of common English words. Table II shows the extracted attributes and their corresponding weights in each service. Each row of this table represents the attribute vector of one of the services.

IV. SIMILARITY-BASED CLUSTERING

This clustering technique, is based on a measure of similarity of services. Suppose that we have a similarity measure that given two services S_1 and S_2 returns their similarity, $Sim(S_1, S_2)$, which is a number between 0 and 1. In our similarity-based clustering, based on this similarity measure, the similarity of each pair of services is determined. Next the most similar services are selected and if they are not in the same cluster, their clusters are merged.

The semantics of inputs and outputs of a service are specified in the service profile section of OWL-S description of that service by mapping each input/output parameter to a concept which is defined in an OWL ontology. Therefore, in order to calculate the semantic similarity between inputs and outputs of two services the semantic similarity of their corresponding OWL concepts can be measured. In section IV-A, we describe a method for calculating the degree of semantic similarity between two OWL concepts. This method will then be used in section IV-B for measuring the degree of functional similarity between two services. Section IV-C shows how the textual descriptions that are provided in service descriptions can be extracted and be used as an alternative measure of similarity of services. In section IV-D, we describe how the functional and textual similarity measures can be combined to provide a compound measure for semantic similarity of web services.

A. Concept Similarity

In this section, we describe a method for measuring the degree of similarity of two OWL concepts. This measure will then be used in next section for determining the degree of functional similarity of two services.

Definition 1 (Concept Similarity): A similarity $\sigma : \mathcal{C} \times \mathcal{C} \rightarrow [0, 1]$ is a function from a pair of concepts to a real number

between zero and one expressing the degree of similarity between two concepts such that:

- 1) $\forall x \in \mathcal{C}, \sigma(x, x) = 1$
- 2) $\forall x, y \in \mathcal{C}, \sigma(x, y) = \sigma(y, x)$

Ontologies define properties of concepts and relationships between them. One of these relationships is the *IS-A* relationship. We use this relationship for extracting concept taxonomies from ontologies. If \mathcal{C} is the set of concepts defined in an ontology then the concept taxonomy \mathcal{T} is defined as $\mathcal{T}(\mathcal{C}, \preceq)$ such that $c \preceq c'$ means that c *IS-A* c' (concept c is subsumed by concept c').

For calculating the degree of similarity of two concepts, we use a slightly modified version of the algorithm which is introduced in [13]. The concept similarity is defined as: $\sigma(c, c') = 1 - \delta(c, c')$, where $\delta(c, c')$ denotes the weighted distance of these two concepts in the taxonomy. A weight value $\omega(c)$ is assigned to each concept c in the concept taxonomy. Since the distance between two given concepts represents the path over their closest common parent, $ccp(c_1, c_2)$, it is calculated as the sum of the their distances two their closes common parent:

$$\delta(c_1, c_2) = [\omega(ccp(c_1, c_2)) - \omega(c_1)] + [\omega(ccp(c_1, c_2)) - \omega(c_2)]$$

The weight values of concepts in the taxonomy are calculated with the following formula:

$$\omega(n) = \frac{1}{k^{l(n)+1}}$$

where $l(n)$ is the length of the longest path from the root concept to node n in the taxonomy and k is a predefined factor larger than 1 indicating the rate at which the weight values decrease along the taxonomy (currently we set k equals 2). This formula has to desirable properties: the semantic differences between upper level concepts are bigger than those between lower level concepts (in other words: two general concepts are less similar than two specialized ones) and that the distance between sibling concepts is greater than the distance between parent and child concepts. For example, in Fig. 4, similarity degree between *Truck* and *Sport Car* can be calculated as follows:

$$\begin{aligned} \omega(Vehicle) &= 0.5 \\ \omega(Truck) &= 0.25 \\ \omega(SportCar) &= 0.125 \\ \delta(Truck, Vehicle) &= 0.5 - 0.25 = 0.25 \\ \delta(SportCar, Vehicle) &= 0.5 - 0.125 = 0.375 \\ \delta(Truck, SportCar) &= 0.25 + 0.375 = 0.625 \\ \sigma(Truck, SportCar) &= 1 - 0.625 = 0.375 \end{aligned}$$

For calculating the closest common parent of two concepts, first their common parents are determined as follows:

$$\begin{aligned} PS(c, T) &= \{c' \in T; c \preceq c'\} \\ CP(c_1, c_2) &= PS(c_1, T) \cap PS(c_2, T) \end{aligned}$$

Service Name	Functional Attributes			Textual Attributes							
	#SSN	#Address	#BirthDate	service	receive	SSN	person	return	address	birth	date
S_1	0.5	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0
S_2	0.5	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.3

TABLE II
ATTRIBUTE VECTORS OF TWO SAMPLE SERVICES

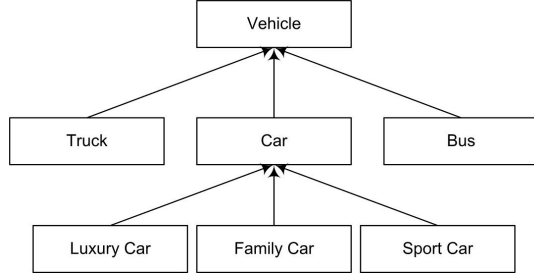


Fig. 4. An example concept taxonomy

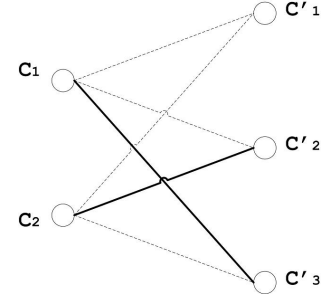


Fig. 5. A Mapping between inputs of two services

In the above formulae, $PS(c, T)$ is the set of parents of concept c derived from taxonomy T , and $CP(c_1, c_2)$ is the set of the common parents of concepts c_1 and c_2 . The closest common parent of these two concepts is the concept in $CP(c_1, c_2)$ with the smallest ω value (this is because that the concept with the smallest ω value has the longest path to the root of the taxonomy and hence it is the most specific concept in the set):

$$x = ccp(c_1, c_2) \Leftrightarrow$$

$$x \in CP(c_1, c_2) \wedge \forall y \in CP(c_1, c_2); \omega(y) \geq \omega(x)$$

B. Functional Similarity

A good measure for calculating the degree of similarity of two services is the degree of similarity between their functional properties such as their inputs and outputs. In this section, we describe a method for calculating the degree of input/output similarity of services based on the semantic similarity measure that we introduced in the previous section.

Each service has a set of inputs and a set of outputs and each input/output parameter is semantically annotated with mapping it to an OWL concept. Therefore, the problem here is to calculate the similarity of two sets of concepts. Using the concept similarity measure of the previous section, we are able to calculate the similarity of each pair of concepts in these two sets. Then the best mapping between the elements of these two sets of concept must be found. This can be shown by a bipartite graph in which each part shows one of the sets of concepts and edges show the similarity of concepts that are connected to them. Fig. 5 shows one such mapping.

Function $\Pi(C_1, C_2)$ which is defined in Fig. 6 computes the best mapping that can be obtained between concepts of sets C_1 and C_2 (the mapping that maximizes the overall degree of similarity of pairs). By using this recursive function, we can determine the similarity of input sets of two services,

$\Pi(I_1, I_2)$, and also the similarity of output sets of these two services, $\Pi(O_1, O_2)$. Then the functional similarity of the two services can be defined as the average of these values:

$$FunctionalSim(S_1, S_2) = \frac{\Pi(I_1, I_2) + \Pi(O_1, O_2)}{2}$$

C. Text based Similarity

Sometimes two similar services have inputs and outputs that are mapped to different concepts in the reference ontology. In these situations, using the pure functional similarity measure of the previous section would fail and these two services would not be detected as similar. Therefore, it is needed to combine another similarity measure with the functional similarity measure for better results.

Service descriptions often contain parts which include textual information and the similarity of terms which are in these parts of the service descriptions can be used as an alternative measure for similarity of services. Information retrieval methods for finding similar documents can be used here. Vector model [8] is a well known method in this area. Using this model, services can be represented as vectors $\vec{s}_k = (w_{1k}, w_{2k}, \dots, w_{nk})$ where w_{ik} is a weight for term i in the description of service s_k . Then similarity of services would be the cosine of the angle between their corresponding vectors. Terms are extracted from textual parts of the service description after removing common words and extracting the base form² of them using WordNet [12]. Then the *tf-idf* [8] weighting method is used for determining weight of each term. According to this weighting method, the importance of a term

²For example *Reservation* would be converted to *Reserve*

$$\Pi(C_1, C_2) = \begin{cases} \frac{\max_{c_1 \in C_1, c_2 \in C_2} \{\Pi(C_1 - c_1, C_2 - c_2) + Sim(c_1, c_2)\}}{\min\{|C_1|, |C_2|\}}, & C_1 \neq \phi, C_2 \neq \phi \\ 0, & C_1 = \phi \vee C_2 = \phi \end{cases}$$

Fig. 6. Function that computes the best mapping between concepts of sets C_1 and C_2

i in service description j is:

$$w_{ij} = tf_{ij} \times idf_i = tf_{ij} \times \log\left(\frac{N}{n_i}\right)$$

where tf_{ij} is the frequency of term i in service description j ; N is the number of service descriptions in the collection; and n_k is the number of service descriptions in the collection that contain term i . Textual similarity of services would be calculated as follows:

$$TextSim(S_1, S_2) = \frac{\vec{S}_1 \cdot \vec{S}_2}{|\vec{S}_1| |\vec{S}_2|}$$

D. Compound Similarity

Our experimental results which are presented in the next section, show that neither the functional similarity nor the textual similarity measure individually can have a good precision and it is better to combine these two measures for reaching to a better similarity measure. We used a linear combination of these two measures as follows:

$$\begin{aligned} Sim(S_1, S_2) &= \omega_f \cdot FunctionalSim(S_1, S_2) \\ &+ \omega_t \cdot TextSim(S_1, S_2) \\ \omega_f + \omega_t &= 1 \end{aligned}$$

The question here is how to find the best values of ω_f and ω_t . We used the Neural Networks [7] model for this purpose. Since neural networks produce continuous outputs, they may quite naturally be used for estimation and prediction. After creating the neural network with a data mining tool and training it with the test collection which is discussed in the next section, we found that in this test collection the best values for ω_f and ω_t are $\omega_f = 0.603$ and $\omega_t = 0.397$ (the estimated accuracy of this result was 99.29). The neural network that was constructed for our training set had two hidden layers with 13 and 8 neurons respectively.

The advantage of this clustering method over the attribute-based clustering is in its time complexity. The time complexity of the attribute-based clustering method is $O(n^3)$ [14], while time complexity of the Similarity-based Clustering method is $O(n^2)$ (n is the number of services). This is because that in $O(n^2)$ similarity of the n^2 pairs of services is determined and then in one path over the result clusters are created.

V. EXPERIMENTAL EVALUATION

For evaluating the methods which are proposed in previous sections, we semi-automatically built a test collection of 241 OWL-S service descriptions which use a reference ontology with 494 concepts³. In order to have better results, most of

the service descriptions are extracted from real word services that are publicly available on the web. Most of the structure of the taxonomy of the referenced ontology is also based on the hierarchy of the words in WordNet. Services are grouped into 25 different domains such as Travel, Education, Shopping and News.

For evaluating the proposed clustering technique, we used the *Recall* and *Precision* metrics. *Recall* is the extent to which the method creates *all* of the good clusters while *Precision* is the extent to which the method creates *only* good clusters. In order to determine whether a cluster is good or not we use the domain of services that are specified by a human expert for each of the services in the test collection.

In the agglomerative clustering method which we used for creating the taxonomy, in each step the two most similar clusters are merged with each other. Therefore in each step, a new cluster is created. After creating clusters, the domain information of services is used to determine whether this cluster is a good cluster or not. Table III shows some of the first steps of one such clustering. In this table, five of the created clusters are good clusters while one of them is not a good cluster. In the first row of the table when we detect one of the five good clusters, the recall value would be 0.2 and the precision value is 1.0 (because no bad cluster is created up to that point). Similarly, in the 4th row of the table, a bad cluster is created and because before that row we have seen three of the five correct clusters the recall value would be 0.6. On the other hand, because three of the four created clusters are good clusters the precision would be 0.75. But because we have another precision value for the 0.6 recall level the maximum of these precision values would be considered for this recall level. The other recall and precision values are calculated similarly.

Fig. 7 shows the precision and recall values of the two clustering algorithms of section II on our test collection. As can be seen in this figure, the precision of the two algorithms in most of the recall levels is greater than 0.8 which is a quite acceptable precision. However the similarity-based clustering method results in better clusters. It should be noted that in our experiments, we evaluated the three methods of determining clusters distances, namely single linkage, average linkage, complete linkage. Our results showed that the average linkage method has a better precision that the two other methods. Therefore the results which are shown in Fig. 7 are based on the average linkage method.

Fig. 8 shows parts of the taxonomy that was created by applying our proposed method on the test collection. As can be seen in this paper, most of the services that are related to book shopping and book information gathering are clustered in the same cluster.

³This test collection is publicly available at <http://projects.semwebcentral.org/projects/sws-tc>

Created Cluster	Belong to the same domain?	Recall	Precision
{SSN Finder, Social Security Number Finder}	Yes	0.2	1.0
{Get Book ISBN, ISBN Finder}	Yes	0.4	1.0
{Address Finder, Address Book}	Yes	0.6	1.0
{SSN Finder, Social Security Number Finder, Get State Center}	No	0.6	$\max\{1.0, 0.75\} = 1.0$
{Send Email, Email Sender}	Yes	0.8	0.8
{Send Email, Email Sender, Send Simple Email}	Yes	1.0	0.83

TABLE III
A SAMPLE CALCULATION OF THE PRECISION AND RECALL VALUES

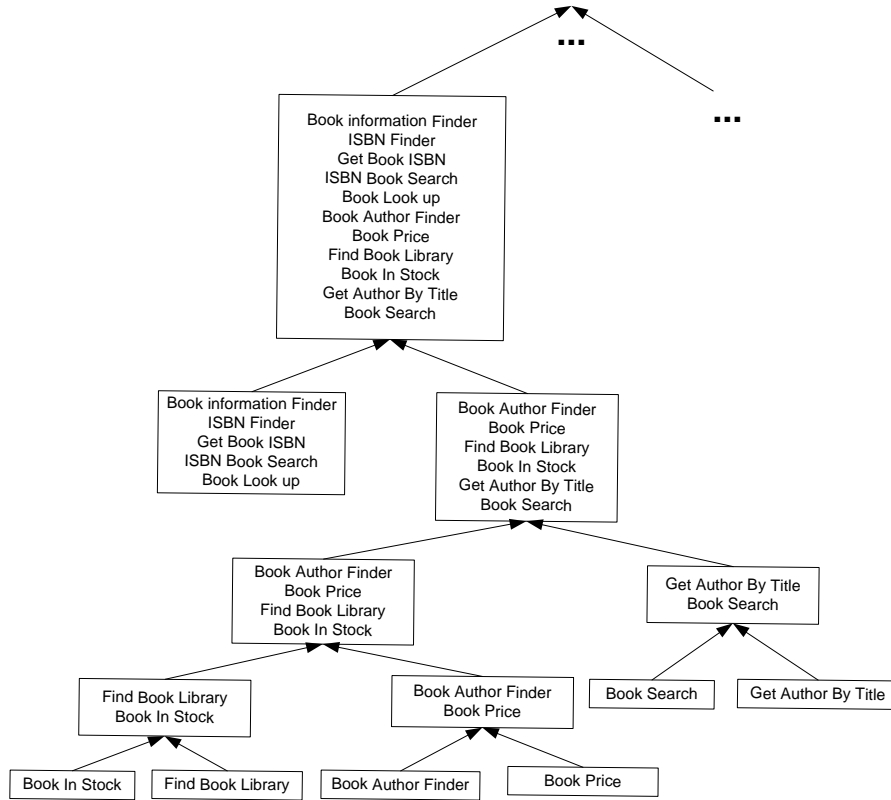


Fig. 8. Parts of the taxonomy created for services of the sample test collection

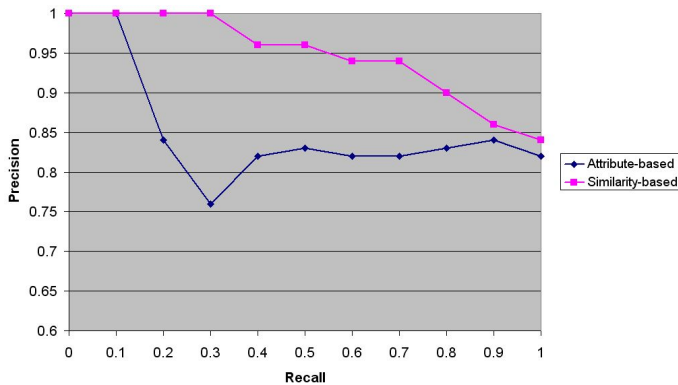


Fig. 7. Recall and Precision diagram of the proposed algorithm

VI. RELATED WORKS

As far as our knowledge, this paper is the first work that discusses taxonomy construction for web services. [15], [1] have used clustering methods for improving service discovery techniques. In [15] a similarity-based approach is used for searching Web Services described in WSDL. Authors have built a web service search engine, Woogole, that supports searching for web service operations similar to a given one. Their tool also supports searching for web service operations composable with a given one. It only uses information available in WSDL files, but clusters it, based on the names of the fields, in an effort to extract semantically meaningful concepts.

In [1], authors have used some of the clustering algorithm for dividing services into similar groups. They have also provided some experimental results, but their data set contains

only 33 WSDL service descriptions and it is hard to evaluate their work.

[16] describe how machine learning and clustering techniques can be used to attach semantic metadata to Web forms and services. It explore the use of machine learning techniques to automatically create such metadata training data. The authors investigate the following three sub-problems, which are essential components to any tool for helping developers create Web Services metadata. They propose to cluster Web Services in order to automatically create category taxonomy. They begin by describing an algorithm for classifying HTML forms into semantic categories, as well as assigning semantic labels to each form field.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a clustering based taxonomy construction algorithm for automatic creation of service taxonomies. We showed how taxonomies can be generated using the result of agglomerative clustering methods. Then we introduced attribute-based clustering and similarity-based clustering as two different approaches for detecting the most similar clusters in the bottom-up clustering method. Experimental evaluations of the algorithm show that clustering algorithm that is used in the process of taxonomy construction has a high precision.

We plan to improve the precision of the attribute-based clustering by using Natural Language Processing (NLP) methods in extracting textual attributes of services. We also plan to use the generated taxonomies to improve the precision of service discovery algorithms.

REFERENCES

- [1] Y. H. Sudan Ram and H. Zhao, "A clustering based approach for facilitating semantic web service discovery," in *15th Annual Workshop on Information Technologies & Systems (WITS)*, 2005.
- [2] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *Proc. of the 1st International Semantic Web Conference (ISWC)*, Springer, 2002, pp. 333–347.
- [3] M. Klusch, B. Fries, and M. Khalid, "OWLS-MX: Hybrid semantic web service retrieval," in *Proceedings of 1st International AAAI Fall Symposium on Agents and the Semantic Web*, Arlington VA, USA, 2005.
- [4] B. Benatallah, M. S. Hacid, C. Rey, and F. Toumani, "Semantic reasoning for web services discovery," in *WWW Workshop on E-Services and the Semantic Web (ESSW03)*, Budapest, Hungary, 2003.
- [5] North American Industry Classification System (NAICS). [Online]. Available: <http://www.census.gov/naics>
- [6] United Nations Standard Products and Services Code (UNSPSC). [Online]. Available: <http://www.unspsc.org/>
- [7] D. T. Laros, *Discovering Knowledge in Data*. New Jersey, USA: John Wiley and Sons, 2005.
- [8] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, 1999.
- [9] D. Quang and B. Hong, *Statistical Data Analysis*. [Online]. Available: <http://www.netnam.vn/unescocourse/statistics/statistics.htm>
- [10] D. Martin and et al. OWL-S: Semantic markup for web services. [Online]. Available: <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>
- [11] S. Bechhofer and et al. OWL Web Ontology Language Reference, W3C recommendation 10 february 2004. [Online]. Available: <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
- [12] C. Fellbaum, *An Electronic Lexical Database*. Cambridge, MA: MIT Press, 1998.
- [13] J. Zhong, H. Zhu, J. Li, and Y. Yu, "Conceptual graph matching for semantic search," in *The 2002 International Conference on Computational Science (ICCS2002)*, Amsterdam, Apr. 2002.
- [14] P. Cimiano, A. Hotho, and S. Staab, "Learning concept hierarchies from text corpora using formal concept analysis," *Journal of Artificial Intelligence Research (JAIR)*, vol. 24, pp. 305–339, 2005.
- [15] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proceedings of the 30th VLDB Conference*, Toronto, Canada, 2004, pp. 372–383.
- [16] A. Hess and N. Kushmerick, "Automatically attaching semantic metadata to web services," in *Proceedings of the 2nd International Semantic Web Conference (ISWC 2003)*, 2003.
- [17] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Francisco, USA: Morgan Kaufmann Publishers, 2000.