Abstract

Web service discovery is becoming a challenging and time consuming task due to large number of Web services available on the Internet. Organizing the Web services into functionally similar clusters is one of a very efficient approach for reducing the search space. To cluster Web services, take out the Web services description languages documents and extract the features (e.g., service name) to measure the similarities. Complex terms are used as Web service features in some contexts. Current approaches do not consider about the hidden semantic pattern exists within the complex terms. We present an approach to cluster the Web services into functionally similar Web service clusters that mine Web Service Description Language (WSDL) documents and generate ontologies by using complex terms for the measuring purpose of similarity. We use both logic based reasoning and edge count base similarity measuring techniques for calculating the similarity using generated ontology. Experimental results show our clustering approach with ontology learning, has better performance comparing with approach which is not considering about the ontology learning.

Keywords: Web Service, Web Service Clustering, Ontology Learning

1. Introduction

Web services are loosely coupled software component which are popular implementation of Service Oriented Architecture, published, located and invoked across the web. Web services have shown potentiality as a distributed computing paradigm and are appropriate to publish and describe business processes and models as services. So now most of the business organizations are moving towards the Web services. Hence, numbers of Web services publish on the Internet are being increased in recent years [1, 2]. Discovery of services which can match to a client query is becoming a challenging and time consuming task due to large number of Web services.

Service discovery is the process of finding the appropriate services from set of services and service matchmaking is the very important operation of the discovery process. Matching algorithm used to determine whether the capabilities of a requested service and an advertised service conform to each other. The matching algorithm used in matcher module is at the core of each Web service discovery system [3]. There are several methods for calculating the Web service similarity in matchmaking process such as Information Retrieval (IR) techniques like TF-IDF [4], ontology based techniques [5] and hybrid methods like SAWSDL-MX2 hybrid match maker which used both logic based reasoning and Information Retrieval (IR) techniques [6, 7]. However if the query space is very large matchmaking can be very expensive due to lot of expensive similarity computation. So pruning the query space is a very important task in Web service discovery. Organizing the Web services into functionally similar clusters is a one of very efficient approach for search space pruning process. This allows the clients to identify appropriate services according to his or her requirements and then browse similar results; those found within the same cluster efficiently.

Web service similarity computation is the key operation of clustering process. Extract the features from Web services which can be used to describe the functionality of Web service in order to compute the similarity. Most of the time Web services use complex terms for the service names for an example AuthorizePhysicianService, operation names, message names, etc. Current similarity computation approaches in Web service matchmaking consider about the similarity of individual terms in complex terms but not considering about the hidden pattern exists within the complex term. By investigating the hidden pattern (e.g., AuthorizePhysician is a subclass of Physician) in complex terms can generate the ontologies to increase the performance of similarity computation.
This paper proposes an approach for clustering Web services into functionally similar clusters by using data mining techniques to extract features from WSDL files and proposes an ontology learning method to increase the performance of clustering approach.

The rest of this paper is organized as follows. Section 2 we discuss the related works. In section 3 we propose clustering approach. Section 4 describes our proposed ontology learning method. In section 5 we propose an approach to compute the similarity of two concepts in generated ontology based on the proposed matching filters. Section 6 explains how to extract features from WSDL documents. Section 7 presents the integration of the extracted features and clustering algorithm. Section 8 discusses the experiment and evaluation. Section 9 concludes the study.

2. Related Works

Web service clustering approach has been used to reduce the search space in order to improve the efficiency of service discovery. Liu and Wong [8] extracted four features content, context, host name and name from WSDL documents and integrated in order to group Web services into functionality based cluster by applying text mining techniques. Sometimes mining service context or considering the host name would not provide good contribution in clustering Web services. Because some providers advertise the services through their own website, so it is resulted different Web services on the same site. Instead of using context name and host name Khalid et al [9] extracted features; WSDL messages, WSDL types (complex types), WSDL ports with WSDL content and Web service name and integrated in order to cluster the Web services. In measuring similarity of WSDL types, WSDL messages and WSDL ports determine the number of matches between a pair of Web services and in measuring similarity of WSDL content and Web service name Normalized Google Distance (NGD) has been used. In this research for the measuring purpose of the similarity of WSDL types only considered the element type but not the name attribute or structure of the complex data type.

Lee and Kim [10] presented an approach that automatically generates ontologies from Web Application Description Language (WADL) documents to cluster parameter names of Web services resources into meaningful concepts. This research utilizes the heuristic as the basic of clustering, in that parameters tend to express the same concept if they frequently occur together. Association rule was used to group parameters. However in this research, ontology was not used to measure the similarity of Web services. It was used to capture the relationships between words using the patterns exists between them to increase the precision.

Xianzhi et al [11] partitioned Web services and historical requirements into clusters and identified the probabilistic correspondence between service clusters and requirement clusters using statistical analysis to increase the efficiency of service composition. Euclidean distance and Pearson correlation were used in similarity measurement. Florian et al. [12] arranged Web service in a functionality graph, which is based on the Plug-in relationship. Every connected component in the graph forms a service cluster.

3. Proposed Clustering Approach

The proposed approach uses WSDL files to cluster the Web services. WSDL is an XML-based format that describes the Web services. Clients are hoping to access a Web service can read and interpret WSDL file of the service. Through the WSDL file, a Web service client learns where a service can be accessed, what operations the service performs, the communication protocols, the service supports and the correct format for sending messages to the service. WSDL file contains main elements; types, messages, port type, binding and service. In this approach the features which describe the functionality of Web service are extracted. Service name, messages, operation name and domain name are the selected features. For each and every selected feature, generate ontologies by applying the proposed ontology learning method to measure the similarity of features. By integrating these features together Web services can be grouped into functionally similar clusters. Figure1 shows the block diagram of proposed approach.
Ontology Learning Method

Idea of this step is to increase the semantic similarity of two Web services. The construction of high-quality ontology manually is difficult, costly and need huge amount of human effort [13]. In this research ontologies are generating automatically by using complex terms and their underling semantics. In order to create the ontologies, initially extract the relevant feature (e.g., Service name) from the WSDL file. If the feature is a complex term then split it into individual terms based on several assumptions. For example, split the *AuthorizePhysicianService* name into three parts (*Authorize*, *Physician*, *Service*) based on the assumption that a capital characters indicate the start of a new word. And split the *lecturer-of- University* into three parts (*lecture*, *of*, *University*) based on the assumption that a hyphen (-) uses to join two words. After tokenizing the name stop-word filtering is performed to remove the stop words (*lecture*, *University*).

The most important key point of the ontology construction is identifying the semantically meaningful concepts and relationships that exist between the concepts. After the pre-processing as the next step find the TF-IDF value of all the tokenized words. It is important to identify the importance of the used words for complex terms. After finding the average TF-IDF value, arrange the words in ascending order according to the TF-IDF value and rank the words by giving lowest rank to the word with lowest TF-IDF value and highest rank to the word with highest TF-IDF value.

Various pattern analysis techniques applied to capture latent patterns which are hiding in complex terms. Mainly there are two types of relationships: relationship subclass and relationship property.

**Definition 1. Subclass Relationship**

Concept $C_i$ is a subclass of concept $C_j$, if $C_i \in LSC(C_j)$. Where $LSC(C_j)$ is the set of least specific concept (Direct Children) of $C_j$.

**Rule 1. Head-Modifier relationship**

Head-Modifier relationship can use to capture the relationship. This is because Heads and modifiers express hyponymy relations between lexical items. Most of the time the right most element is the head of the construction and the element to the left is the modifier of the head [14].

---

**Figure 1** Block Diagram of the clustering approach
Example 1. *EducationalOrganization*
Here organization is modified by the Education. So *Educational Organization* is subclass of *Organization*.

Definition 2. Property Relationship
Concept $C_i$ is a property of concept $C_j$, if $C_i \in \text{PROP}(C_j)$. Where $\text{PROP}(C_j)$ is the set of properties of $C_j$.

Rule 2. Noun1+Noun2
According to the rule proposed by Lee et.al [10] Noun1+Noun2 form a relationship property.

Example 2. *OrganizationAddress*
In this case *OrganizationAddress* is a property of *Organization*.

Rule 3. Concept equal Modifier
If one concept is equal to modifier term of another concept name, then there exists a property of relationship.

Rule 4. Common Modifier
If the modifier of one concept is equal to the modifier term of another concept and if the modifier is not a concept then there exists a property of relationship.

Example 3. *EducationalEmployee, EducationalOrganization*
Relationship can be taken as *EducationalEmployee* has *EducationalOrganization* and *EducationalOrganization* has *EducationalEmployee*.

As the final step of the ontology creation, by using TF-IDF rank and pattern analysis techniques generate the concepts and relationships between concepts. We choose the highest rank word first and then select all complex terms which make use of this word to build the complex term. Next select other tokenized words which join with highest rank word in building complex terms. Then apply pattern analysis technique to identify the relationships. This process has to be repeated until all the tokenized words are considered.

5. Similarity Computation

Service similarity is measured by calculating term similarity together with the generated ontologies. Both logic based reasoning and edge count base similarity measuring techniques are used for calculating the similarity. In the following part we define filters which are used in similarity computation.

5.1. Matching Filters

Definition 3. Exact Match
Service $S_i$ exactly matches with service $S_p$, if $C_i \equiv C_j$ where $C_i$ and $C_j$ are concepts in ontology $O_p$ and they represent the same feature of $S_i$ and $S_j$ respectively (e.g., Service name).

Definition 4. Property Match
Service $S_i$ is property matches Service $S_p$, if $C_i$ is a property of $C_j$ where $C_i$ and $C_j$ are concepts in ontology $O_p$ and they represent the same feature of $S_i$ and $S_j$ respectively.

Definition 5. Property-&-Property Match
Service $S_i$ is property-&-property matches $S_j$ if $C_i \in \text{PROP}(C_k) \land C_j \in \text{PROP}(C_k)$. Where $C_i$ and $C_j$ are concepts in ontology $O_p$, and they represent the same feature of $S_i$ and $S_j$ respectively and $C_k$ is another concept in ontology $\text{PROP}(C_k)$ is the set of properties of $C_k$.

**Definition 6. Plug-in Match**
Service $S_i$ plugs into $S_j$, if $C_i \in \text{LSC}(C_j)$ where $C_i$ and $C_j$ are concepts in ontology $O_p$ and they represent the same feature of $S_i$ and $S_j$ respectively. $\text{LSC}(C)$ is the set of least specific concept(Direct Children).

**Definition 7. Sibling Match**
Service $S_i$ sibling matches $S_j$, if $C_i \in \text{LSC}(C_k) \land C_j \in \text{LSC}(C_k)$. $C_i$ and $C_j$ are direct children of concept $C_k$. Where $C_i$ and $C_j$ are concepts in ontology $O_p$ and they represent the same feature of $S_i$ and $S_j$ respectively. $C_k$ is more specific than $C_j$.

**Definition 8. Subsumes Match**
Service $S_i$ subsumes service $S_j$, if $C_i > C_j$ where $C_i$ and $C_j$ are concepts in ontology $O_p$ and they represent the same feature of $S_i$ and $S_j$ respectively. $C_i$ is more specific than $C_j$.

**Definition 9. Logic Fail Match**
Service $S_i$ logic failed to match $S_j$, if $C_i$ and $C_j$ are in same ontology $O_p$ but fail in above 6 matches.

**Definition 10. Fail Match**
Service $S_i$ failed to match $S_j$, if $C_i$ and $C_j$ are in heterogeneous ontologies $O_p$ and $O_q$.

If there exists, an exact match between two concepts then similarity of that two concepts is equal to the highest value, 1. If the match is Plug-in, Subsumes, Property, Property-&-Property, Sibling or Logic Fail then uses the equation 1 to calculate the similarity.

$$\text{Sim} \left(C_i, C_j\right) = W_m + W_a \times \text{ESim} \left(C_i, C_j\right)$$

(1)

Values of weight $W_m$ and $W_a$ are determined by the matching filters. ESim($C_i$, $C_j$) is the edge base Similarity which is calculated by using equation 2[15]. Where $d(C_i, C_j)$ is the shortest distance between concepts $C_i$ and $C_j$. $D$ is the maximum depth of the ontology. Add 1 to both $d(C_i, C_j)$ and $2D$ to avoid log (0) when the shortest path length is 0.

$$\text{ESim} \left(C_i, C_j\right) = -\log \left(\frac{d(C_i, C_j)}{2D}\right)$$

(2)

WorldNet[16] is used to calculate the similarity if the two concepts are in heterogeneous ontologies (fail Match).

### 6. Extracting Features from WSDL Document

By using data mining techniques, features from WSDL documents are extracted. Ontologies are constructed for extracted feature and check the similarity of Web services by using concepts in ontologies.

#### Feature 1. Service Name
Web services names use in the WSDL files are extracted. Composite names are used as Web service names. Our proposed ontology learning method is applied to the collection of service names in order to generate the ontologies. Similarity of the service names are computed by using above defined filters and similarity computing equations.

#### Feature 2. WSDL Messages
Message element in WSDL document describes the names and format of the messages which are sent or received by the Web services. Messages are composed of <part> elements. Each part stands for an instance of a particular type. In this research we consider the part element for measuring the similarity of request messages and response massages in Web services. As in a similar way in service name first generate the ontologies for the collection of part names and then calculate the similarity of part names.

If the massage has multiple logic units, multiple part names are used, such as `get_PRICERequest` in `UserScience-fiction-novelPriceService` has science-fiction-novel and user part elements. So to get the average similarity value uses the equation 3.

\[
\text{Sim}_\text{key} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \left( \frac{\text{sim}(s_{xi}, r_{yj})}{m+n} \right)}{(m+n)} \quad (3)
\]

Where \( s_{xi} \) and \( r_{yj} \) denotes the individual parts element of request or response massages in service \( S_i \) and \( S_j \) respectively. \( m \) is the number of part element in service \( S_i \) and \( n \) is the number of part element in service \( S_j \).

**Feature 3. WSDL Operation**

Operations are listed in the main element port type in WSDL document. Operations describe an abstract description of action supported by the service. Operation has several parameters such as name and another optional attributes specifying the order of the parameters used in this operation. As an example we can define an operation called “GetSymbol” that has the “GetSymbolInput” massage as an input that produce the “GetSymbolOutput” message as an output. In this paper we extract the operation name as a feature. And as similar way in service name, measure the similarity of operation names in Web services.

**Feature 4. Domain Name**

Web services domain names use in the WSDL files are extracted. Like in Web service names here also Composite names are used. Our proposed ontology learning method is applied to the collection of domain names in order to generate the ontologies. Similarity of the service names are computed by using above defined filters and similarity computing equations.

**7. Feature Integration and Clustering**

Integrate the features by using the following formula 4 to get the similarity score \( \text{Sc}(S_i, S_j) \) between Web services \( S_i \) and \( S_j \) in order to cluster the Web services

\[
\text{Sc}(S_i, S_j) = W_x \times \text{sim} (\text{Name}_i, \text{Name}_j) + W_y \times \text{sim} (\text{Op}_i, \text{Op}_j)
+ W_z \times \text{sim} (\text{Message}_i, \text{Message}_j) + W_d \times \text{sim} (\text{Dom}_i, \text{Dom}_j) \quad (4)
\]

The weights \( W_x, W_y, W_z \) and \( W_d \) are real values between 0 and 1.

Agglomerative algorithm is used as the clustering algorithm. It is a bottom up hierarchical clustering method. Algorithm starts by assigning the each services into their own clusters and then starts to merge together the most similar clusters based on proximity of the clusters at every iteration until stopping criterion is met (e.g., number of clusters).

The algorithm 1 represents the algorithm that we have used as the clustering algorithm. Key operation of the algorithm is the computation of the proximity of two clusters. We use similarity score value to determine the proximity of two clusters. If we have more than one service in a cluster then we calculate the average similarity score of services in two clusters as the proximity value.
Algorithm 1. Clustering algorithm

```
1. RequiredClusters = n
2. Let each service be a cluster
3. Compute the proximity matrix
4. CurrentClusters = NoOfServices
5. while CurrentClusters != n do
   6. Merge the two closest clusters
   7. CurrentClusters = getCurrentClustersNo
   8. Update the proximity matrix
   9. end while
```

8. Experiment and Evaluation

The experiments were conducted on Microsoft windows 7, Intel core i7-3770, 3.40 GHz and 4GB RAM. Java was used as programming language and Jena Framework was used to build ontologies. Jena Framework provides a collection of tools and Java libraries to develop ontology. WSDL documents were gathered from the real-world Web service providers and Web service repositories. We performed manual classification in order to categorize the Web service data set to compare the results. Education, Medical, Food, Film and Vehicle were the identified categorize.

Selected features were extracted and applied the ontology learning method to each and every feature. Figure 2 shows the sample output put of a part of a generated ontology for service name.

![Figure 2. Sample output of an ontology generation phase](image)

We set the values of $W_m$ and $W_e$ in equation 1 as shown in table 1. according to the matching filter. We used Precision, Recall and F-measure to evaluate the performance. Precision is the fraction of a cluster that consists of services of a specified class. Recall is the fraction of a cluster that consists of all services of a specified class. F-measure is the combination of both precision and recall that
measures the extent to which a cluster contains only services of a particular class and all services of that class. Following equations 5, 6 and 7 are used to calculate the three criteria.

\[
\text{Precision}(i,j) = \frac{N_{M_f}}{N_{M_f}}
\]  

(5)

\[
\text{Recall}(i,j) = \frac{N_{M_f}}{N_{M_j}}
\]  

(6)

Where \( N_{M_f} \) is the number of members of class \( i \) in cluster \( j \). \( N_{M_j} \) is the number of members of cluster \( j \), and \( N_{M_i} \) is the number of members of class \( i \). The F-measure of cluster \( i \) with respect to class \( j \) is

\[
F(i,j) = \frac{2 \times \text{precision}(i,j) \times \text{recall}(i,j)}{\text{precision}(i,j) + \text{recall}(i,j)}
\]  

(7)

To check the performance of our approach we implemented the clustering approach without using ontology learning method. In this case also service name, operation, messages and domain name were selected as features and Agglomerative algorithm was used as the clustering algorithm. But, WorldNet is used to calculate the similarity. In composite name case, consider a pair of candidate matching. Experimental results in the Table 2. shows that the recall value of education group and vehicle group as 100%. That means all the Web services that are belonging to education group successfully placed into one cluster as well as the entire Web services that are belonging to vehicle group also successfully placed into one cluster. And also there are no any false positives in education and food clusters. Because precision value of these two clusters is 100%.

Vehicle and medical clusters obtained the lowest precision value and recall value respectively. Because some of the services which are belonging to medical group, are incorrectly placed in the vehicle cluster. When analyzing the WSDL documents we observed that extracted features are not providing sufficient details to identify their ontology and this may result to fail our define filters to apply the similarity measurement. So in this case we have to use edge count based term similarity method to measure the similarity. For example, CheckEquipmentAvailability service which is belonging to medical category was not successfully placed in medical cluster. In this case service failed to join with other services like HospitalDiagnosticProcessTimeMeasure, MedicalClinicDiagnosticProcessTimeInterval and many more in medical group to generate ontology.
According to Table 2, both approaches get 100% precision for education cluster. But in other clusters our ontology learning clustering approach get higher precision, recall and F-measures values than method using edge count based term similarity. As examples ontology learning approach improved the recall value by 58% in vehicle category and improved the precision value of food category by 48%.

9. Conclusion

In this paper we proposed an approach to increase the performance of Web service clustering by introducing the ontology learning mechanism. We identified the semantically meaningful concepts and captured the latent semantic pattern by using complex terms which are used as features of Web services and then automatically generated the ontology. We used both logic based reasoning and edge count base similarity measuring techniques for calculating the similarity using generated ontology. In this case we defined seven matching filters in order to compute the similarity. Experimental results show our clustering approach with ontology learning has a better performance comparing with approaches which are not considering about the latent pattern exists within the complex terms.

As future work, we plan to improve performance of our clustering approach by considering more features and increase the precision by removing the noise data in clusters. Also more experiment will be carried out to visualize clusters.

10. References

