Master Thesis

Spring 2012

School of Health and Society

Department of Computer Science

Web Service Matching based on Semantic Classification

Writer:

Feng Deng

Instructor:

Dawit Mengistu

Examiner:

Kamilla Klonowska

School of Health and Society
Department Design and Computer Science
Kristianstad University
SE-291 88 Kristianstad
Sweden

Author, Program and Year:

Feng Deng, ISY11

Instructor:

Dawit Mengistu

Examination:

This graduation work on 15 higher education credits is a part of the requirements for a Master program in Embedded System 2011.

Title:

Web Service Matching based on Semantic Classification

Abstract:

This degree project is mainly discussing about a web service classification approach based on suffix tree algorithm. Nowadays, Web Services are made up of WSDL web Service, RESTful web Service and many traditional component Services on Internet. The cost of manual classification cannot satisfy the increasing web services, so this paper proposes an approach to automatically classify web service because of this approach only relies on the textual description of service. Though semantic similarity calculation, we achieve web service classification automatically. Experimental evaluation results show that this approach has an acceptable and stable efficiency on precision and recall.

Language:

English

Approved by:

_____

Kamilla Klonowska Date

Examiner

Table of content

1. INTRODUCTION

1.1 Background and context

With the popularity of Internet, network-based distributed applications are more and more widely used, traditional software architecture gradually convert to a service-oriented architecture (SOA). The relevant technical standards for Web services are gradually maturing, more and more development tools support for Web services. Therefore, many organizations and enterprises launch services on Internet for more convenience and more profit. Facing with such a large number of services, how to find a service rapidly and precisely? Users need a dynamic, flexible web service classification mechanism to support the web service classification and discovery technology. Therefore, the service classification has been as a significant research direction in the field of service computing domain. Actually, the majority of web services are classified by manual currently, so this paper describes a modified suffix tree clustering algorithm to automatically classify Web services.

1.2 Web service

The term of Web services describes a standardized way of integrating web based applications using XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available [1]. Web services allow different applications from different sources to communicate with each other without time-consuming custom coding and also because all communication is in XML, Web services do not depend on any operating system or programming language hence it is both language and platform independent in nature [2]. Recently, some commercial Web services have emerged, such as Amazon AWS [5] (Amazon Web Services), companies can be access to AWS to get the service of computing power, storage space and other services, and then them should pay corresponding money according to actual usage. Authoritative organizations have predicted in the next few years, Web services will play an increasing important role to commercial applications.

Web service aims to solve the main problem is the data and the integration of applications. Many distributed technology are trying to solve this problem, such as CORBA (Common Object Request Broker is Architecture), DCOM / COM (Distributed Component Object Model) and RMI (Remote Method Invoke). However, these technologies are correspondingly too complex to use or cannot be easy to penetrate a firewall, do not work on the Internet, and cannot interoperate well between different protocol standards. Web service is a good solution to solve these problems, and it can also provide greater interoperability.

What is the usage of Web services? First, it can reuse application components. So we need not to make these over and over again. Web services can also offer

application components something like: currency conversion, weather reports, or even language translation. Secondly, it can connect existing software. Web services can help to solve the problem of interoperability through giving different applications a way to link their data. In essence, people exchange data with freedom between different applications and different platforms because of the utility of Web service.

1.3 Web service classification and discovery

In general, Web services discovery technology utilizes WSDL and UDDI, which based on keywords and classification to discovery services. However, this method can't help the users to discovery expected services and has low precision and recall.
According to using the Key Word matching, resulting in the service precise is not high. Many researchers have done some work:
1. Kawamura et al proposed semantic matching integrated into UDDI, WSDL language to describe Service, the service matches the WSDL annotation which contains a description of the semantic information, this method can only provide part of the semantic description is not flexible in the service discovery.[3]
2. Sycara et al proposed LARKS language definition Web services. They customized a weight network to calculate the similarity between Web services. The algorithm requires manual intervention in the face of a large amount of Web services. The workload of the weight network will become the bottleneck of Web service discovery [4].

Classification of web services is one methodology that can be used in order to enhance the speed of web service discovery process. Recently, the classification of services including: 1. Manual classification. 2. Based on text-mining service classification. 3. Based on semantic annotation service classification.

1.3.1 Manual classification.

UDDI implements the function of artificial classification of the service. Each web service has to publish on the web and needs to be registered in the UDDI registry. However, with the increase in services, the labor cost of these manual classification services is becoming increasingly high. The classification has become increasingly difficult.

Therefore, more and more organizations began to research the method of automated classification. Currently, the main automatic classification has two categories: classification based on text mining and classification based on semantic annotation.

1.3.2 Based on text-mining service classification.

Web service classification based on text mining has two modules: a text mining module, and Web services classification module: text mining module is mainly responsible extracted the key term vectors from the Web service's WSDL or the description document; Web services classification module through a set of pre-know the classification of Web services as a training set, using machine learning

algorithms(the most commonly used based Rocchio algorithm for the classification of the AWSC [22], based on Naive Bayes algorithm for the classification[23], based on support vector machine algorithm SVM for a classification[24]) to get the key term vectors, and then test a Web service which is intended to classify, using the similarity algorithm to calculate the similarity with these clusters, the service is distributed to that cluster with the greatest similarity. These methods did not consider the semantic similarity between terms and the semantic conflict, such as a description of the service is “advertising” and the other services described as a “shopping”, without considering the semantic relation, it is difficult to classify these two services in the same category.

1.3.3 Classification based on semantic annotation.

It is a research hotspot that adding semantic annotation to fully describe a service in the WSDL document. Semantic Web service technologies, such as the Ontology Web Language for Services (OWL-S) [25] are developing the means by which services can be given richer semantic specifications. There are automatic or semi-automatic semantic annotation classifications, such as METEOR-S [26], ASSAM [27]. These methods add semantic annotation of Web services, and then consider the semantic similarity to complete a classification of services. The drawback of such methods is in need of additional corresponding domain ontology library. Although different fields are actively developing their own domain ontology, only biological gene ontology is relatively robust. In other areas such as the Finance field, it is difficult to find the commonly used domain ontology. Therefore, the implementation of a classification method based on semantic annotation has a certain degree of difficulty at nowadays.

According to a lack of existing services of automatic classification method, we propose a new classification approach integrating text mining, semantic technology and suffix tree data structure.

1.4 Paper organization

This paper contains six chapters, a brief introduction as below:

1) Chapter1: we introduce the background and context of this paper. What is web service and next part is web service classification and discovery.
2) Chapter2: we introduce the key technologies about machine learning algorithm, the original suffix tree algorithm, WordNet dictionary and semantic similarity calculation algorithms based on WordNet.
3) Chapter3: we propose a framework about web service semantic classification in a detailed description.
4) Chapter4: we propose a classification approach based on the original suffix tree algorithm. Including one more combination process and constructing a concept-instance tree structure. The new approach will classify web services precisely and rapidly.
5) Chapter5: Here is an experimental evaluation section. We mention the resource of

testing data and the designing process of experiments. At last, we will also analyze and evaluate the experimental results.

6) Chapter6: we summarize a conclusion related to this paper.

2. RELEVANT THEORY

2.1 Clustering algorithms

A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way” [6]. Thus according to the previous definition of clusters, we can describe three types of clustering algorithms as following: Distance-based Vector Space Model, Model based on Frequent Word and Model based on the word sequence.

2.1.1 Distance-based Vector Space Model

Distance-based Vector Space Model is the traditional clustering algorithm. The algorithms can be broadly classified into the following types [7]:

a) Partitional clustering attempts to directly decompose the data set into a set of disjoint clusters. More specifically, they attempt to determine an integer number of partitions that optimize a certain criterion function. The criterion function may emphasize the local or global structure of the data and its optimization is an iterative procedure. In this category, K-Means algorithm [8] is a commonly used algorithm.

b) Hierarchical clustering proceeds successively by either merging smaller clusters into larger ones, or by splitting larger clusters. The result of the algorithm is a tree of clusters, called dendrogram, which shows how the clusters are related. By cutting the dendrogram at a desired level, a clustering of the data items into disjoint groups is obtained. In this category, this kind of classical algorithm includes BIRCH algorithm [9]，CURE algorithm [10], ROCK algorithm [11] and so on.

c) Density-based clustering. The key idea of this type of clustering is to group neighboring objects of a data set into clusters based on density conditions. DBSCAN algorithm [12] and DENCLUE algorithm [13] are representatives of this category.

d) Grid-based clustering. This type of algorithms is mainly proposed for spatial data mining. Their main characteristic is that they quantize the space into a finite number of cells and then they do all operations on the quantized space. A widely known algorithm of this category is STING [14] (Statistical Information Grid-based method).

2.1.2 Model based on frequent word

In this model based on frequent word, these documents share the most frequent word sets because they are on the same subject in one cluster. Different clusters describe different subject, then the sharing frequent word set is less. For example, a cluster of subject may be a NBA tournament and if two documents are sharing two subject of words {Miami Heat, L. James}. It is reasonable to divide that both

documents into a cluster. We find some representative algorithms in this model, such as FTC algorithm [15], HFTC algorithm [15] and FIHC algorithm [16].

2.1.3 Model based on word sequence

The model based on word sequence use a sentence as a unit to construct a generalized suffix tree, the given document is coding by the all possible strings. In a case that a certain string is shared by some specified documents, these strings can be extracted as the documents' labels. The description of document is common (or public) phrases rather than simply a group of key words. The common phrase is one or more orderly sequence of words, preserving the sequence of words relationship. Oren Zamir and Oren Etzioni proposed STC [17] algorithm in 1998s. The next section will depict the original STC (suffix tree clustering) in details.

2.2 Original STC (suffix tree clustering) algorithm

STC algorithm is based on identifying the phrases that are common to groups of documents by building a suffix tree. STC algorithm is different from the other kind of clustering algorithms. STC meanly includes four logical steps: first, document "cleaning"; secondly, constructing a generalized suffix tree; thirdly, identifying base clusters; the last step is to combine these base clusters into clusters.

1. Cleaning document

Document pretreatment intend to slim each document. In this step, the string of text representing the content of each document is transformed by using a light stemming algorithm. It is stripping the HTML tags, separator and common English stop words, as well as extracting word stems (deleting word prefixes and suffixes and reducing plural to singular).

2. Constructing a generalized suffix tree

Here we need to present what is a generalized suffix tree. We can cite the definition of a suffix tree from the literature [17]. A suffix tree of a string S is a compact trie containing all the suffixes of S. We treat documents as strings of words, not characters, thus suffixes contain one or more whole words. In more precise terms:

1). a suffix tree is a rooted, directed tree.

2). every internal node has at least 2 children.

3). every edge is labeled with a non-empty sub-string of S (hence it is a trie). The label of a node in defined to be the concatenation of the edge-labels on the path from the root to that node.

4). no two edges out of the same node can have edge-labels that begin with the same word (hence it is compact)

5). to every suffix s of S, there exists a suffix-node whose label equals s.

Every suffix-node is marked to designate from which string (or strings) it originated from (i.e., the label of that suffix node is a suffix of that string). Figure 1 is an

example of the suffix tree of a set of strings - "cat ate cheese", "mouse ate cheese too" and "cat ate mouse too".

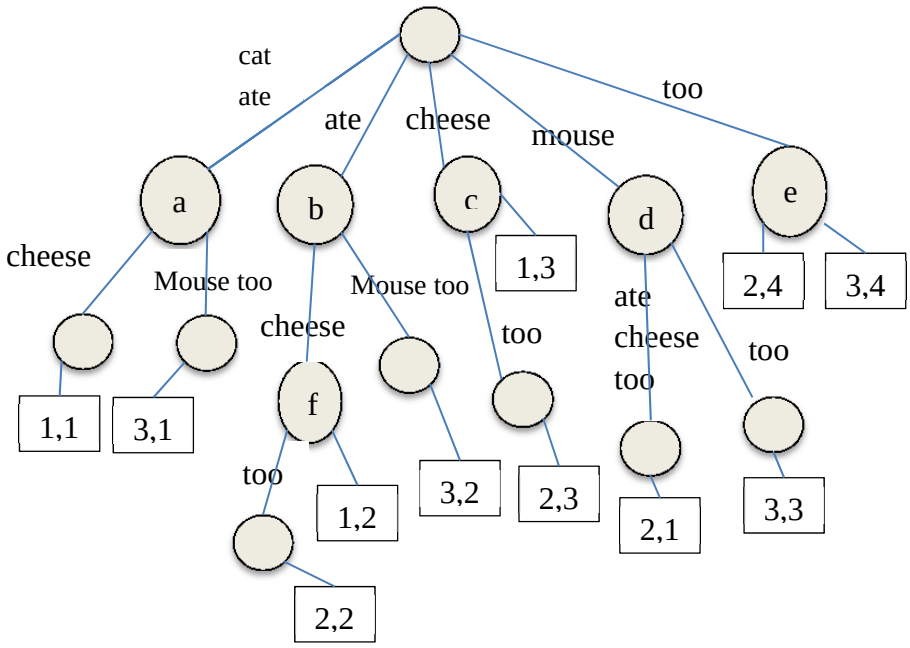


Fig 1 the generalized suffix tree constructed according to "cat ate cheese", "mouse ate cheese too" and "cat ate mouse too

In this figure, the circle represents node, the numbers in the square represent document group. (Each node of the suffix tree represents a group of documents and a phrase that is common to all of them. The label of the node represents the common phrase; the set of documents tagging the suffix-nodes that are descendants of the node make up the document group. Therefore, each node represents a base cluster. Furthermore, all possible base clusters (containing 2 or more documents) appear as nodes in this suffix tree.)

3. Identifying Base Clusters

In this section, each base cluster is assigned a score s(B) that is a function of the number of documents it contains, and the words that make up its phrase. The function is given by:

$$s(B) = |B| \bullet f(|P|) \qquad (1)$$

where |B| indicates the number of document in a base cluster and |P| is the number of word in a phrase. Table 1 shows six nodes from the example shown in Figure 1 and their corresponding base clusters.

Node	Phrase	Documents
a	cat ate	1.3
b	ate	1,2,3
c	cheese	1,2
d	mouse	2,3
e	too	2,3
f	ate cheese	1,2

Table 1 the base clusters

4. Combining these base clusters into clusters

In this step, documents may be sharing more than one phrase. To avoid the document overlapping and a nearly identical cluster, this step is assigned to merge base cluster with high overlap in the document set. They defined a binary similarity measure to calculate whether base clusters should be merged or not. The binary similarity will be 1 if match the given formula:

$$|B_m \cap B_n|/|B_m| > 0.5 \text{ and } |B_m \cap B_n|/|B_n| > 0.5 \quad (2)$$

Otherwise their similarity will be defined as 0.

Next they draw a graph to illustrate this step. The two nodes would be connected by a line if the similarity of two base clusters is 1. Each cluster consists of the union of the document of all its base clusters. Figure 2 explains the base cluster graph of the six base clusters in table.1. We notice the result of clustering that there is only a single cluster in this example. Each circle represents a base cluster in this graph.

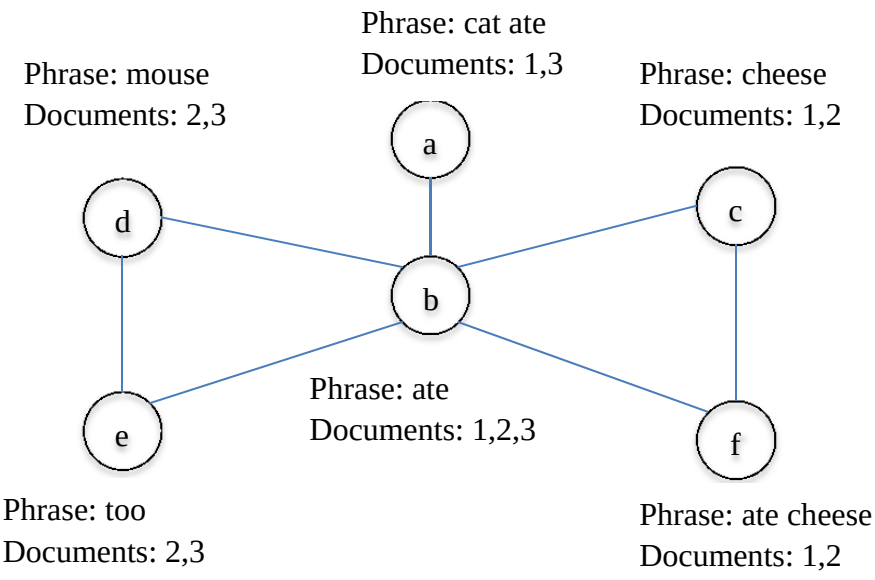


Figure 2 the clustering result according to the example figure.1

2.3 WordNet

We are familiar with some traditional dictionary organized by the alphabetical order. This kind of dictionary has a strong power to query words, but this dictionary can't provide the relation, such as synonymy, antonymy (opposing-name), hyponymy (sub-name). It obviously can't satisfy the requirement of the intelligent application if a dictionary only includes the word morphology. The emergence of semantic dictionary fills in this blank space, as well as meets the requirements of intelligent application and widely used in computational linguistics and natural language processing.

WordNet [18] is an online lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet groups words together based on their meanings, so it superficially resembles a thesaurus.

WordNet divides words into nouns, verbs, adjectives, adverbs. However these four groups are organized in different ways. WordNet organizes the nouns depending on the hierarchy of subjects, verbs according to lexical collection relationship, adjectives and adverbs according to N-dimensional space. For example, we illustrate the 25 base classes from the nouns in the following table 2.

act, activity	food	possession	animal, fauna
group, grouping	process	artifact	Location
quantity, amout	attribute	motivation, motive	Relation
body	natural_object	shape	cognition, knowledge
natural_phenomenon	state	communication	person, human_being
substance	event, happening	plant, flora	Time
feeling, emotion			

Table 2 Nouns hierarchy in WordNet

A much larger variety of semantic relations can be defined between words and between word senses than are incorporated into WordNet [19]. WordNet includes the following semantic relations:

Semantic relation	**Syntactic category**	**Examples**
Synonymy (similar)	n, v, aj, av	ripe, tube rise, ascend sad, unhappy rapidly, speedily
Antonymy (opposite)	Aj, av, (n,v)	wet, dry powerful, powerless friendly, unfriendly rapidly, slowly

Hyponymy (subordinate, is a kind of)	n	suger maple, maple maple, tree tree, plant
Meronymy (is a part of)	n	brim, hat gin, martini ship, fleet
Troponomy (manner)	v	march, walk whisper, speak
Entailment	v	drive, ride divorce, marry
Note: N=nouns, Aj=adjectives, V=verbs, Av=adverbs		

Table 3 semantic relation in WordNet

2.4 Semantic similarity calculation

Currently, various semantic similarity calculation approaches based on WordNet have been proposed. In this paper, I would like to adopt JCn [20] algorithm and Lin [21] algorithm. These algorithms focus on the calculation of information content (IC).

2.4.1 Lin algorithm

Lin algorithm is one of the similarity measure algorithms; the author had proposed the similarity between two concepts in taxonomy such as WordNet. Lin algorithm considers the distance of two concepts in semantic domain is determined by their common and difference. Assuming that the taxonomy is a tree, if $x_1 \in C_1$ and $x_2 \in C_2$, the commonality between x_1 and x_2 is $x_1 \in C_0 \wedge x_2 \in C_0$, where C_0 is the most specific class that subsumes both C_1 and C_2. The function as following:

$$\mathrm{sim}_{\mathrm{Lin}}(x_1, x_2) = \frac{2 \times \log P(C_0)}{\log P(C_1) + \log P(C_2)}$$

For example, Figure 3 is a fragment of the WordNet. The number attached to each node C is P(C). The similarity between the concepts of Hill and Coast is:

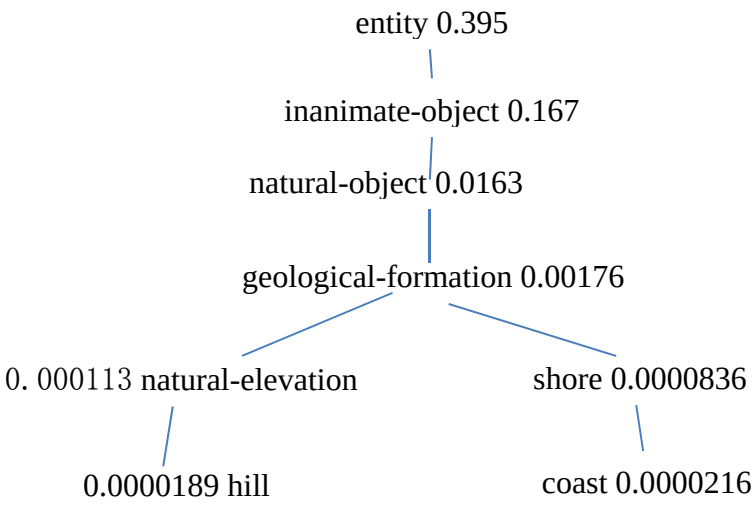


Figure 3 A fragment of WordNet

$$\text{sim}(\text{hill}, \text{coast}) = \frac{2 \times logP(geological - formation)}{logP(hill) + logP(coast)} = 0.59$$

2.4.2 JCn algorithm

JCn algorithm uses the distance between two concepts in taxonomy as the basis for his similarity. Jiang and Conrath described the two conceptual distances depending on the IC (information content) difference between two concepts with the nearest common ancestor.

According to the notation in information theory, the information content (IC) of a concept or class c can be defined as following:

$$\text{IC}(c) = log^{-1}P(c)$$

where P(c) is the probability of encountering an instance belonging to concept c. It is shown in this formula: In the case of the hierarchical structure, where a concept in the hierarchy subsumes those lower in the hierarchy, this implies that P(c) is monotonic when one node moves up the hierarchy. As the node's probability increases, its information content or its informativeness decreases. If there is a unique top root node in the hierarchy, then its probability is 1, hence its information content is 0 [20]. The upper hierarchy concept has a greater probability in the hierarchy structure, and it means the corresponding information is less.

Therefore, according to the definition of IC, the similarity/distance of two concepts (C_1,C_2) can be formally defined as:

$$sim_{jcn}(c_1, c_2) = \frac{1}{dist_{jcn}(c_1, c_2)} = \frac{1}{IC_{(c_1)} + IC_{(c_2)} - 2 \times IC_{(L_{c_1,c_2})}}$$

Where $dist_{jcn}(c_1, c_2)$ denotes the distance of two concepts. At the same time, L_{c_1,c_2} indicates a synset which includes c_1, c_2 and a closest hypernym of c_1, c_2. Obviously, the distance between the two concepts is farther, the similarity between them is lower. For example, in the figure.4, c_1 and c_2 are respectively assigned as {truck} and {bike}, then L_{c_1,c_2} is equal to {wheeled, vehicle}.

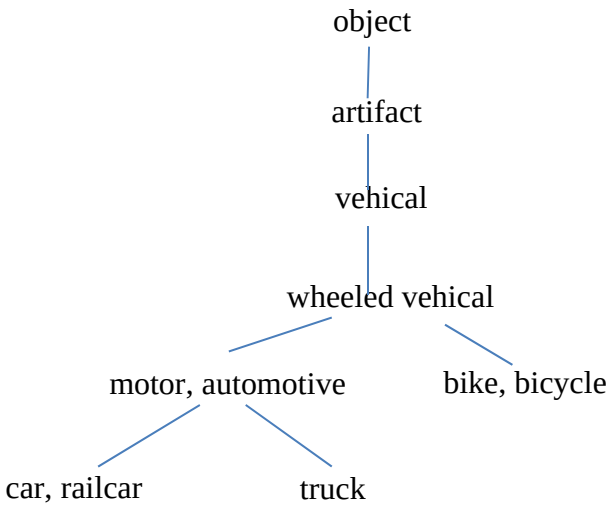


Figure 4 A fragment of WordNet

$$sim_{jcn}(\text{bick, truck}) = \frac{1}{dist_{jcn}(\text{bick, truck})} = \frac{1}{IC_{(bick)} + IC_{(truck)} - 2 \times IC_{(\text{wheeled,vehicle})}} = 0.57$$

3. THE FRAMEWORK FOR SERVICE SEMANTIC CLASSIFICATION

The framework of my idea for service semantic classification is made up with two sections, such as a document mining module and an automatic web service classification module.

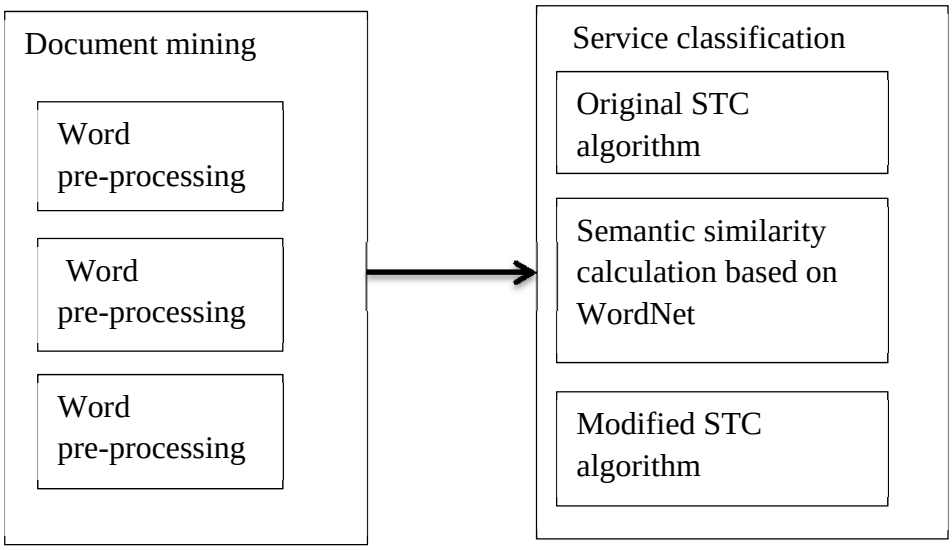


Figure 5 the framework of web service classification

3.1 Document mining module

The responsibility of this document mining is to extract some important information from web service description document. The input is the description of web service and the output is a word sequence. An acceptable output will impact the quality of service classification quite a lot.

The existing service classification only deal with WSDL document of each service. However, more and more web services based on RESTful style are emerging, such as Google AJAX Search API and ProgrammableWeb API. The majority of approaches depend on a parse based on WSDL4j, which can parse a WSDL document. Many web services rely on RESTful and many component service based on RMI cannot be processed in these approaches because of a lack of WSDL document. But this approach proposed in this paper has a more common parser which can extract key words from all kinds of description document.

This section contains two steps: deleting non-frequent word, removing the pause word and punctuation marks. Eliminating the non-frequent word and getting frequent word set are considered from two aspects. On one hand, if a word or phrase only appear once in many documents, and then this documents would be clustered in one

cluster, which being described as this word or phrase. Not considering the appearing frequency of a word or phrase will seriously impact the clustering precision. On the other hand, eliminating the frequent word can reduce clustering dimension.
The pause words and punctuation marks in service description will influence the quality of service classification. Therefore, we remove these noise including common stops word and the other stops words which are related to description, such as “platform”, “search”, “engine”, “program”, “service”, ”software”, “resource”, “management”, “application”, “data”, and so on.

3.2 Service classification module

After the document mining processing, the next task begins to classify services. This paper puts forward the modified suffix tree clustering algorithm to the service classification.

In the traditional suffix tree clustering algorithm, all documents in the document set will be constructed as a generalized suffix tree, and the common phrases in the base clusters will be to represent as the clustering description. If many base clusters have a high document overlap threshold, they need to be merged as one cluster. In the practical application, taking into account only by merging with higher document overlap threshold cluster is not enough. In a certain case, some clusters’ centers are synonymy or hyponymy relation, them should be merged even though they do not satisfy the document overlap threshold. This paper is responsible to address several issues: first, how to further merge the base clusters to improve the accuracy of service classification; secondly, the clustering process to select the suitable keywords or phrases as the description of clustering is necessary, so how to make the extraction of the cluster center as a comprehensive, accurate, concise description of document clustering. These solutions will be discussed in detail in chapter 4.

4. Classification approach base on Suffix tree clustering

In essence, STC algorithm only merges the nodes which meet the overlap threshold of documents, that is, these documents contain a set of common phrase. If two nodes are synonymous relations, these two base clusters are two different clusters. In order to solve this weakness, we intend to utilize WordNet to cluster synonymous nodes. Furthermore, we can extract the clustering center to describe each cluster.
The modified algorithm is based on the following assumptions:

a) The base cluster contains the keywords or key phrases which has the synonymous relationship. Apparently we should to remove redundant information and use more abstract keywords or key phrases to represent them. For instance, a cluster tag contains the keywords “bed” and “furniture”, we calculate the similarity by Lin algorithm and get the similarity (bed, furniture) = 0.9077155514023946. The hyponymy of “furniture” is “bed” and furniture is a base cluster in WordNet, so we can use “furniture” to take place of “bed”.
b) It is clearly reasonable that two base clusters meet a specified threshold in a certain similarity and merge them to be one cluster. In the original STC algorithm, an overlap rate of document between base clusters serves as the clustering criterion. We consider two base clusters within a synonymous relationship or a relationship of father and son, another meaning is that the base cluster described in another fundamental concept is similar, but the document does not contain a high overlap rate. In this case, we declare that the synonymous relationship between the two base clusters have a high probability to be merged.
c) After using the suffix tree clustering algorithm to cluster a collection of documents, we can successfully extract the concept from keywords and phrases as the centre of the cluster. It is not simple to extract a certain keywords or phrases in documentation set, rather than generalize the corresponding concept.

Because original STC algorithm has four steps, now we plus one more step. The purpose of this step is to process a second combination and to describe base clusters by generalizing abstract concept centre.

Step 5- second combination of base clusters

Firstly, we check the base cluster to find the hyponymy relations and synonyms relations between the keywords or phrases. Furthermore, some keywords and phrases are not the hyponymy relations and the synonymy relations, but they express the same sense in the semantics domain. In this case, we can set up a conceptual center and instance. We are only required to compare with the similarity at conceptual hierarchy in some base clusters. If a certain threshold of similarity dose not be satisfied, it means this two base clusters do not meet the mergence condition. Of course, we calculate the similarity of the two base clusters by the above method. In one case, the value of similarity is 1.0 and it does not mean that the two base clusters can be merged. For example, a conceptual centre of base cluster is state#n#2 (problem#n#1,

disease#n#1), the other is state#n#2 (dryness#n#1, waterlessness#n#1). Their similarity is 1.0, but they should not be merged into one cluster. Because the first one expresses an illness state and the second one contains the meaning of humidity. Although their hyponymy is the same word, different information is showing at the instance hierarchy.

With some of the above preparations, now we can carry out the combination of base clusters in second time. First of all, about the similarity between any two base clusters, we only need to calculate the similarity in the conceptual hierarchy. If the similarity between the two base clusters of the conceptual hierarchy does not meet the threshold (we will define this threshold value in the next experimental evaluation chapter), then the two base clusters should not be merged; If the similarity of two base clusters meets the threshold, then it is possible to judge the two base clusters are similar, and then proceed with calculation of similarity in instance hierarchy (if exist corresponding instances). By calculating the similarity in the conceptual hierarchy, we can exclude non-similar base clusters, thus reducing the number of the similarity comparison of the base clusters, and improving efficiency.

Each base cluster sometimes contains more than a concept and each concept has a different degree of contribution to the base cluster, so we add a weight value of each concept by using the term frequency "tf" to describe the weight of concept. The formula as following:

$$w(C_k) = tf(C_k) = \frac{fq_{k1} + fq_{k2} + \ldots + fq_{km}}{m}$$

fq_{km} is the frequency of No.m hyponymy of concept K.

4.1 Concept hierarchy

Calculate the similarity of concept hierarchy in base clusters:

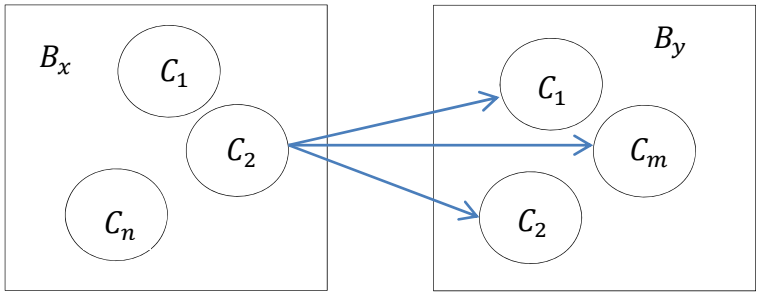


Figure 6 calculate the similarity of concept hierarchy in base cluster

First of all, we start to elect the each concept in base cluster B_x and calculate the similarity of this concept with all concepts in the base cluster B_y. Then we can get the maximum value of similarity between one concept in B_x and all concepts in B_y. The concept of similarity in the two base clusters is required to meet a certain threshold (we will define this threshold value in the next evaluation chapter).

$$\text{ConceptSim}(C_{xi}, C_y) = \max_j Sim(C_{xi}, C_{yj})$$

C_{xi} is one of concepts in base cluster B_x , C_{yj} is all concepts in base cluster B_y (j=1,2,3…). The meaning of this formula is to respectively calculate similarity which is one of concept in B_x with each concept in B_y . We choose the maximum value as the similarity of C_{xi} and C_y.

Secondly, according to the weight value of each concept, calculate the value of the similarity between all concepts in one base cluster and the all concept in another base clusters. We use it as the similarity value between base cluster B_x with base cluster B_y. We define ClusterSim(B_x,B_y) as the similarity of the two base clusters. Formula of calculation is as following:

$$\text{ClusterSim}(B_x, \text{B}_y) = \frac{\sum_{i=1}^{n} w_{xi} ConceptSim(C_{\text{xi}}, \text{C}_y)}{n}$$

This step helps us to exclude some non-similar base clusters. We need to do further similarity calculation in instance hierarchy about the remaining base clusters which may be similar. If we get two similar base clusters in the instance hierarchy, then we can assert that these two base clusters are truly similar or not. Now we introduce the detail about instance hierarchy.

4.2 Instance hierarchy

We exclude the clusters which have not satisfied the threshold in concept hierarchy. We only consider about the clusters which have met the threshold in concept hierarchy. The following task is to calculate the similarity of instance hierarchy in base clusters:

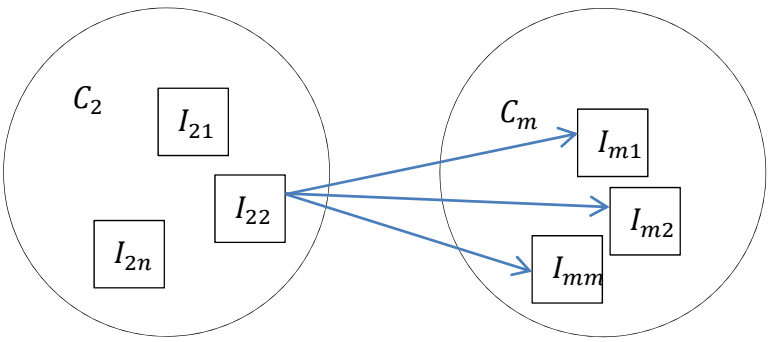


Figure 7 calculate the similarity of instance hierarchy in base clusters

We assume that the concept C_2 of base cluster B_x and the concept C_m of base cluster B_y are similar by calculating the similarity between two base clusters in the concept hierarchy. Then we need to further calculate the similarity of keywords and phrases between these two concepts in the instance hierarchy. If their similarity meets a certain threshold (we will define this threshold value in the next evaluation chapter) in the instance hierarchy, these two base clusters would be combined as a base cluster; if it is not, never merge.

We randomly select an instance of a concept (because a concept may contain a number of instances, but the common point of each instance can be abstracted as a same concept). The instance of similarity is calculated as following:

$$\mathrm{InstanceSim}(I_{xi}, I_y) = \max_j Sim(I_{xi}, I_{yj})$$

Among them, I_{xi} is an instance (this instance is some keywords or phrases) of a certain concept in the base cluster B_{x} , I_y is a collection of instances of a certain concept in the base cluster B_y. As the same, I_{yj} is an instance (this instance is some keywords or phrases) of a certain concept in the base cluster B_y. The functionality of the formula is randomly select an instance of a concept in the base cluster B_{x} and calculate the similarity with all instances of a concept in the base cluster B_y.
We find the maximum similarity as the instance hierarchy similarity. $\mathrm{InstanceSim}(I_{xi}, I_{\mathrm{yj}})$ is based on Lin-algorithm in WordNet dictionary.

The similarity of concept hierarchy can be substituted for the similarity of instance hierarchy. This equation declares the replaceable relationship: the similarity is a concept of a base cluster with all concepts of another base cluster.

$$\mathrm{ConceptSim}(C_{xi}, C_y) = \mathrm{InstanceSim}(I_{xi}, I_y)$$

According to the weight value of each concept, calculate the value of the similarity between all concepts in one base cluster with the all concepts in another base clusters. We use it as the similarity value between base cluster B_x with base cluster B_y. We define ClusterSim(B_x,B_y) as the similarity of the two base clusters. The same with the previous Formula:

$$\mathrm{ClusterSim}(B_x, \mathrm{B}_y) = \frac{\sum_{i=1}^{n} w_{xi} ConceptSim(C_{\mathrm{xi}}, \mathrm{C}_y)}{n}$$

Finally, we should compare the $\mathrm{ClusterSim}(B_x, \mathrm{B}_y)$ value with a threshold. If the value of $\mathrm{ClusterSim}(B_x, \mathrm{B}_y)$ is greater than the threshold or equal, then do combination; if less, do not combine.

4.3 Classification process

Practically, the efficiency of classification will decline when the descriptions of web service or the scale of documents is growing largely. To avoid this drawback, if the document scale reaches a certain scale, our solution is to use an STC algorithm based on concept center, which generated by our approach, to classify the non-classification documents. At the same time, we need to further add these concept centers.

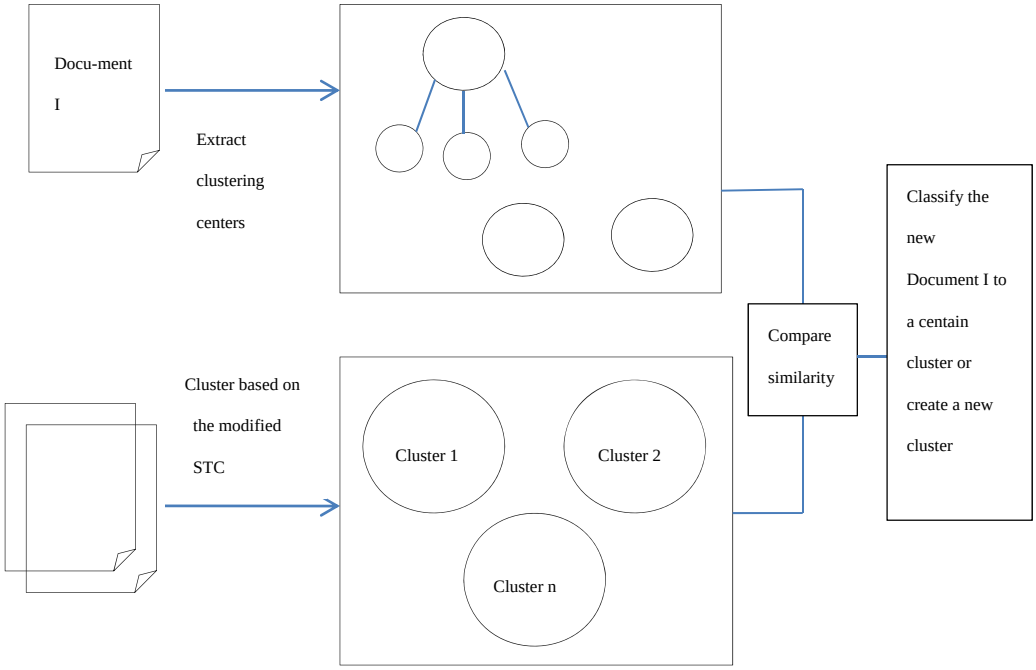


Figure 8 non-classification document's classification process

Extracting a clustering center from the non-classification document I is equivalent to each web service document is a base cluster, and each base cluster has a clustering center which is constructed by the concept-instance hierarchy. Making a similarity comparison between a center of non-classification document and each cluster, and distributing this document I to the most similar cluster when their value of similarity greater than the threshold. If there are no similar clusters, then create an independent cluster.

4.4 Algorithm implementation

Our classification approach is based on the web service descriptions downloaded from the website of "ProgrammableWeb".

First, we read external data and store to memory by using "genDocument()" function in "input.dataset.ReaderService.java" file.

Secondly, it is constructing a generalized suffix tree by using "bulidSuffixTree()" function in "GeneralizedSuffixTree.java" file.

Thirdly, we use "createBaseClusters(SequenceBuilder sb)" function to identify each base clusters. This function is located at "STCClusteringAlgorithm.java" file.

Fourthly, we make a combination between base clusters. The function "createMergedClusters(ArayyList baseCluster)" is also at "STCClusteringAlgorithm.java" file.

Fifthly, we establish a concept-instance tree structure. We use the function "WordNetSimilarility(String str1，String str2)" to calculate the similarity between two keywords. In order to get the parent of two keywords, we use the function "WordNetParent(String str1, String2)". And then, we implement a second time

combination that involves “MergeBasedWordNet(List mergedClusters)” function in “STCClusteringAlgorithm.java” file. We use “getClusterCenter(clusterCandidate mergedCluster)” to extract clustering center in the form of concept-instance tree. After second time combination, we can get the clustering labels by using “getPhaseAndTfList(List mergedClusters)”. At last, “Cluster()” function is an entrance of clusters.

5. EXPERIMENT AND EVALUATION

We did some experiments based on a set of data which is collected from the website of “programmableWeb”. We used the service descriptions as the input of experiments, and classified these services as our output. The difference between the original STC algorithm and modified STC algorithm had been shown in Table.5. Our classification results were compared with the results from manual classification. It is the primary difference that our approach can generate a description of the cluster center.

ProgrammableWeb is the collection of an API platform and now it has the 6619 APIs on the platform. Because new services will be added into ProgrammableWeb, so the amount of APIs will be updated every day. A classification of these APIs is shown in Figure 9 and the number after each category indicates how many APIs within that category.

Browse APIs by Category

Advertising (121)	Events (62)	Music (160)
Shipping (37)	Answers (18)	Fax (13)
News (62)	Shopping (228)	Backend (31)
Feeds (32)	Office (70)	Social (384)
Blog Search (10)	File Sharing (40)	Other (198)
Sports (72)	Blogging (59)	Financial (249)
Payment (157)	Storage (65)	Bookmarks (40)
Food (40)	Photos (158)	Tagging (14)
Calendar (24)	Games (85)	PIM (26)
Telephony (223)	Catalog (1)	Goal Setting (4)
Politics (3)	Tools (417)	Chat (46)
Government (212)	Project Management (82)	Transportation (94)
Database (53)	Internet (443)	Real Estate (47)
Travel (132)	Dating (2)	Job Search (39)
Recommendations (54)	Utility (110)	Dictionary (12)
Mapping (255)	Reference (259)	Video (162)
Education (106)	Media Management (36)	Retail (34)
Weather (39)	Email (146)	Media Search (8)
Science (197)	Widgets (29)	Enterprise (246)
Medical (70)	Search (192)	Wiki (16)
Entertainment (30)	Messaging (186)	Security (121)

Figure 9 the category of APIs in programmableWeb

In a classification task, the terms “true positives”, “true negatives”, “false positives”, and “false negatives” compare the results of the classifier under test with trusted external judgments. The terms positive and negative is referring to the classifier's prediction (sometimes known as the observation), and the terms true and false is referring to whether that prediction corresponds to the external judgment (sometimes known as the expectation) [29]. The relation of these terms is illustrated by the table below:

	Actual class (expectation)	
Predicted class (observation)	tp (true positive) Correct result	fp (false positive) Unexpected result
	fn (false negative) Missing result	tn (true negative) Correct absence of result

The precision and recall values are defined as: [30]

$$\mathrm{p} = \mathrm{precision} = \frac{tp}{tp + fp}$$

$$\mathrm{r} = \mathrm{recall} = \frac{tp}{tp + fn}$$

For evaluating the effectiveness of classification, we use F_1 measure [28] to measure the clustering quality. The score of F_1 will reflect the balance between recall (r) and precision (p). The F_1 measure, initially introduced by van Rijsbergen, combines recall (r) and precision (p) in the following formula:

$$F_1 = \frac{2 \cdot r \cdot p}{r + p}$$

It is necessary to set the overlap threshold of documents and the similarity threshold. In this paper, we set these two threshold=0.6. We collected various catalogues in the experiment, including “Advertising”、“Travel”、“Telephone”、“Storage”、 “Shopping”、“Payment”、“News”、“Music”、“Messaging”、“Internet”、“Games”、“Financial”. The result of experiment is showing as below:

	Cluster center description	Manual category
40 clusters	interact=[Communicate,SMS Messaging] [Mobile]; [Phone]	Telephone
	group_action=[Marketing, Advertising]	Advertising
	travel=[Trip, Travel]; [Trip Booing]	Travel
	[Storage]	Storage
	purchase=[Buying,Shopping]; act=[Deals, Local Deals] [Affiliate Marketing]; [Buying cart]; [eCommerce]	Shopping
	credit_line=[Credit Card, Card Payment] [Payment];	Payment

	[Payment processing];	
	[News]	News
	[Music], [Music streaming]	Music
	[Text], [SMS Text Messaging]; [Bulk SMS Messaging];	Messaging
	[web]; [cloud]; [Hosting]; [IP Address]; [website]; [Internet]; [Shortening]; [Domain]; [Domain Name]	Internet
	[Game]	Games
	[Accounting software]; [Stock]; [Market],[Stock market]	Financial
	[Information];[Social];[Tracking];[base];[real time]; [Network];[Content];	Other topic

Table 5 A experiment results on clustering description centers

As seen in Table 5, we believe that effect of our approach based on modified STC algorithm is nearly identical to the manual result of classification. The cluster centers are obtained by our proposed approach is significantly more than manual classification, which is a reasonable feature in the suffix tree clustering algorithm. These positive results are due in part to the usage of phrases to identify clusters in STC algorithm and also because of the fact that it naturally allows overlapping clusters. We consider a web service description can be occupied by multiple clusters in STC algorithm, as a result that it does not belong to a rigid (or hard) classification. For example, our approach divides “Financial” catalogue into [Accounting software],[Stock];{[Market],[Stock market]}. It describes web service from different aspects. We notice that the words “interact”, “group_action”, “travel”, “purchase”, “act”, “credit_line” are the center of concepts, which is generated by WordNet. These words achieve a second mergence.

We still generated the testing data sets seven times according to “ProgrammableWeb”, and calculated the values of precision and recall every time. Then the testing results are shown in Table 6.

a) Test1: the inputs of date are made up of 6 catalogues (Advertising, Travel, Telephone, Storage, Sports, Shopping)
b) Test2: the inputs of date are made up of 8 catalogues (Advertising, Travel, Telephone, Storage, Sports, Shopping, Payment, News)

c) Test3: the inputs of date are made up of 10 catalogues (Advertising, Travel, Telephone, Storage, Sports, Shopping, Payment, News, Music, Messaging)
d) Test4: the inputs of date are made up of 12 catalogues (Advertising, Travel, Telephone, Storage, Sports, Shopping, Payment, News, Music, Messaging, Medical, Job search)
e) Test5: the inputs of date are made up of 14 catalogues (Advertising, Travel, Telephone, Storage, Sports, Shopping, Payment, News, Music, Messaging, Medical, Job search, Internet, Game)
f) Test6: the inputs of date are made up of 16 catalogues (Advertising, Travel, Telephone, Storage, Sports, Shopping, Payment, News, Music, Messaging, Medical, Job search, Internet, Game, Food, and Financial)
g) Test7: the inputs of date are made up of 16 catalogues (Advertising, Travel, Telephone, Storage, Shopping, Payment, Music, Messaging, Internet, Game, and Financial). Excluding (Sports, Job search, Food, Medical, News), these catalogues have less APIs.

	The amount of Web service	Precision	Recall
Test1	840 APIs	65.98%	67.15%
Test2	1058 APIs	64.93%	65.35%
Test3	1402 APIs	61.67%	68.12%
Test4	1510 APIs	61.24%	65.67%
Test5	2037 APIs	60.63%	59.10%
Test6	2326 APIs	60.75%	54.53%
Test7	2044APIs	63.27%	58.62%

Table 6 testing results statistic on “precision” and “recall”

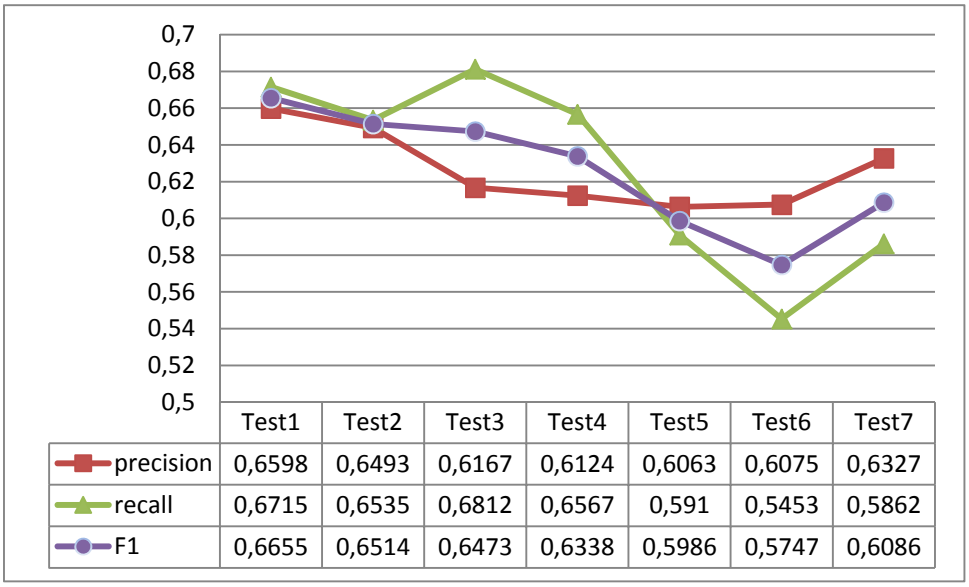
As seen from Table.6, the average precision of our approach is 62.64% and the average of recall rate is 62.65%. However, the average precision of original STC algorithm does not reach 40%, which we are able to get this corresponding percent from the literature [17]. These testing results can prove our approach satisfies high quality of classification.

Comparing with Test 2, the precision of Test 3 descends nearly 3% and the recall of Test 3 ascends a same percent. In Test 3, we add Music catalogue and Messaging catalogue which have 158APIs and 186APIs respectively. Because Messaging catalogue is very similar with Telephone catalogue. For example, their web service descriptions both contain [mobile] term and [Text, SMS Text Messaging] term. These

two catalogues have overlapping descriptions so that the precision of Test 3 decreases greatly. In essence, some APIs in Messaging catalogue should not be classified with Telephone catalogue, nevertheless, it is happening. On the other hand, we also observe the recall of Test 3 has increased. The mean reason of this change is referring to the adding of Music catalogue. Our approach generates a "music" cluster, and then a majority of Music APIs are classified into it.

Comparing with the Test 5, we add Food and Financial catalogues into the testing data set of Test 6. Financial catalogue is not related to the other catalogues in testing dataset, which makes the quality of precision increasing a little bit. But the recall rate decreases greatly. We temporarily believe that Food catalogue has not generated a cluster because of a lacking of enough APIs in Food catalogue, which only has 40 APIs. Furthermore, it is necessary to point out our approach assigns there are 40 clusters in the classification. If not set, there will be appearing hundreds of clusters, and this will be against the original intention of classification. So it is the reason of recall rate drops smoothly.

Moreover, we approve the imbalance of APIs number in different catalogue will impact the accuracy of classification. In test 7, we remove 5 catalogues (Sports, Job search, Food, Medical, News), because their APIs numbers are very less than other catalogues'. Hence, we can observe that the precision and recall rate are improved remarkably.



	Test1	Test2	Test3	Test4	Test5	Test6	Test7
precision	0,6598	0,6493	0,6167	0,6124	0,6063	0,6075	0,6327
recall	0,6715	0,6535	0,6812	0,6567	0,591	0,5453	0,5862
F1	0,6655	0,6514	0,6473	0,6338	0,5986	0,5747	0,6086

Figure 10 experiment results of precision, recall and F_1

Experimental results were obtained from Figure 10, our average F_1 score is 62.57%. F_1 can reflect the balance of precision and recall. The score of F_1 is not fluctuated strongly. We adopt precision, recall and F_1 score to evaluate the performance of classification. The results show our classification approach has an acceptable and stable efficiency.

6. CONCLUSION

In this paper, we proposed an approach for web services automatic classification. Our approach based on WordNet and suffix tree clustering (STC) algorithm can satisfy the classification requirement in the process, including high dimension, explainable and usability. Nowadays, Web Services are made up of WSDL web Service, RESTful web Service and many traditional component Services on Internet. It is suitable to classify these different type services on account of our approach only concern textual description of services as classification principle. In order to consider the accuracy and efficiency in the process of classification, this paper proposed a "concept and instance" tree structure and realized the similarity calculation in this structure. These works played a crucial role to Web Service classification. There is a case that if Web Service is increasing exponentially, as a result that the suffix tree also increase linearly because of this tree structure. This feature can deal with the rapidly growing web service on Internet. Our experimental evaluations indicate this classification approach has good efficiency.

Acknowledge

We thank instructor Dawit Mengistu for his discussion with me related to writing this paper and for his valuable comments on the draft of this paper.
We also thank examiner Kamilla Klonowska for her patience and instructive guidance on this paper.
We wish to thank my friend Lidong Cao for his suggestions and discussions at the experimental evaluation stage.
At last, we sincerely express our deepest gratitude to all people who helped us.

REFERENCE

[1] http://www.webopedia.com/TERM/W/Web_services.html

[2] IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011 ISSN (Online): 1694-0814

[3] Kawamura T, Blasio J D, Hasegawa T, et al. Preliminary report of public experiment of semantic service matchmaker with UDDI business registry[EB/OL].

[4] Sycara K, Klusch M, Widoff S. Dynamic service matchmaking among agents in open information environments[J].ACM SIGMOD Record 1999 28 1 47-53.

[5] Ainazon．Amazon Web Services(AWS)[EB/OL]．http://aws.amazon.com

[6] A Tutorial on Clustering Algorithms
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/

[7] Jain, A.K., Murty, M.N., and Flyn, P.J. (1999). Data Clustering: A Review. ACM Computing Surveys, 31(3), 264–323.

[8] MacQueen, J.B. (1967). Some Methods for Classification and Analysis of Multivariate Observations. In Proceedings of 5th Berkley Symposium on Mathematical Statistics and Probability, Volume I: Statistics, pp. 281–297.

[9] Zhang, T., Ramakrishnman, R., and Linvy, M. (1996). BIRCH: An Efficient Method for Very Large Databases. ACM SIGMOD, Montreal, Canada.

[10] Guha, S., Rastogi, R., and Shim K. (1998). CURE: An Efficient Clustering Algorithm for Large Databases. In Proceedings of the ACM SIGMOD Conference.

[11] Guha, S, Rastogi, R., and Shim K. (1999). ROCK: A Robust Clustering Algorithm for Categorical Attributes. In Proceedings of the IEEE Conference on Data Engineering.

[12] Ester, M., Kriegel, H-P., Sander, J., and Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceeding of 2nd Int. Conf.On Knowledge Discovery and Data Mining, Portland (pp. 226–23).

[13] Hinneburg, A. and Keim, D. (1998). An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In Proceedings of KDD Conference.

[14] Wang, W., Yang, J., and Muntz, R. (1997). STING: A Ststistical Information Grid Approach to Spatial Data Mining. In Proceedings of 23rd VLDB Conference.

[15] Eeil F., Ester M., Xu X.. Frequent term-based text clustering. In proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002:436-442.

[16] Fung B.C.M, Wang K. Ester M. Hierarchical document clustering using frequent itemsets. In Proceedings of SIAM International Conference on Data Mining , 2003:59-66.

[17] Zamir O., Etzioni O. Web Document Clustering: A Feasibility Demonstration. SIGIR Forum. New York, USA: ACM, 1998: 46-54

[18] http://wordnet.princeton.edu/

[19] George A. Miller. WordNet: A Lexical Database for English. COMMUNICATIONS OF THE ACM November 1995/Vol. 38, No. 11

[20] J．Jiang,D．Conrath．Semantic similarity based on corpus statistics and lexical

taxonomy．Proceedings on International Conference Research in Computational Linguistics，Taiwan, 1997，PP．19-33．
[21] Lin D.. An information-theoretic definition of similarity. In Proc. 15th International Conf. on Machine Learning. 1998: 292-304
[22] Richardson L., Ruby S.. RESTful Web Services. USA: O'Reilly Media, 2007.
[23] Lewis David D.. Naïve(Bayes) at Forty: The independence assumption in information retrieval. In 10th European Conference on Machine Learning. LNCS. 2008(1398):21-23.
[24] Zhijun W. ,Li J., Zhenghong Y., et al. Classification of plasmid vectors using replication origin, selection marker and promoter as criteria. plasmid. 2009(61): 47-51.
[25] http://www.w3.org/Submission/OWL-S/
[26] Abhijit Patil, Swapna Oundhakar, Amit Sheth, Kunal Verma. METEOR-S Web service Annotation Framework.
http://lsdis.cs.uga.edu/lib/download/POSV-WWW2004.pdf
[27] Andreas Heß and Nicholas Kushmerick. ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services.
http://iswc2004.semanticweb.org/demos/34/paper.pdf
[28] Yiming Yang. Xin Liu. A re-examination of text categorization methods. http://www.inf.ufes.br/~claudine/courses/ct08/artigos/yang_sigir99.pdf
[29] http://en.wikipedia.org/wiki/Precision_and_recall
[30] Olson, David L.; and Delen, Dursun (2008); Advanced Data Mining Techniques, Springer, 1st edition (February 1, 2008), page 138, ISBN 3-540-76916-1