A Semantic-based Approach to Service Clustering from Service Documents

Bo Jiang, Lingyao Ye, Jialei Wang,Ye Wang
School of Computer and Information Engineering
Zhejiang Gongshang University
Hangzhou, China
{nancybjiang,yewang}@zjgsu.edu.cn

***Abstract*—With the rapid growth of service volumes and types, discovering services in an efficient and accurate manner has become a significant challenge in service computing. Service clustering is an important technology to improve the efficiency of service discovery. In this paper, we propose a new service clustering approach, which starts from service documents and is based on the functional semantics of services. This approach, first, extracts service goals from service description documents by using natural language processing technologies. Then it obtains the semantic similarity between two service goals and clusters the services by the K-means algorithm. Experiments conducted on a real-world service dataset crawled from ProgrammableWeb demonstrate the feasibility and the effectiveness of the proposed approach.**

Keywords-Service discovery; service clustering; natural language processing; service goals; service similarity.

I. INTRODUCTION

Service-oriented Architecture (SOA) is a coarse-grained, loosely coupled service architecture. Service-oriented software system design, has become one of the popular research topics in the field of software [1]. With the development of SOA technology and Software as a Service (SaaS), the number and the types of services on the internet has maintained a rapid growth trend. For example, web services are emerging from traditional web services based on Simple Object Access Protocol (SOAP) to lightweight RESTful web services based on the representational state transfer protocol. But with the surge in the service numbers and types, how to accurately and efficiently discover services to meet user needs becomes a problem in service-oriented computing.

Traditional UDDI (Universal Description, Discovery and Integration) service registration and discovery mechanism only supports the operation of the service syntax level. For example, the keyword-based service matching is often unable to meet user requirements. Therefore, we propose a semantics-based service clustering approach for service discovery. Service clustering is usually used as a preprocessing algorithm for service discovery and complex service matching. First, aggregate similar services into a cluster; second, the service request is directed to a specific service cluster, and henceforth the service search space is reduced, which consequently improves the efficiency of service discovery. A lot of existing research shows that service clustering based on the similarity of service goals can improve the efficiency of service discovery [4]. At present, service clustering based on the similarity of service functionality has been studied extensively. Elgazzar et al. [4] proposed a WSDL (Web Service Description Language) document mining approach. This approach extracts five key features of service functionality from WSDL documents, and then clusters services with similar functionality based on these features. Liu and Yang [5] proposed a text-based service clustering method, which uses the vector space model to represent and deal with the description in the service documents. Then, the service clustering is performed using the multi-hybrid clustering (MHC) algorithm. However, there are two shortcomings in the existing service clustering methods:

1) The types of service documents are limited to WSDL documents or OWL-S documents, while little attention has been paid to RESTful styles which are described in natural language.

2) Few approaches take the semantics of service functionality into consideration. Most approaches use the space vector model to reduce the dimensions of the service documents.

To overcome the above problems, this paper proposes a service clustering method based on the semantics of service functionality. First, service documents containing service meta information is processed by natural language processing techniques; Then the service goal set is extracted; Third the similarity of the service goals is calculated from the semantic perspective; finally, the service clustering is conducted by the K-means clustering algorithm.

This paper is structured as below. Section 2 introduces the whole process of the proposed service clustering method, including the automatic extraction of service goal sets, service similarity computation, and the service clustering algorithm. Section 3 takes the data crawled on the ProgrammableWeb as input of our approach and shows the effectiveness of the approach. Section 4 introduces the related work. Section 5 concludes the paper and discusses the future work.

II. THE SEMANTICS-BASED SERVICE CLUSTERING APPROACH

The approach can be divided into the following steps:

1) Acquire service meta information from Programmable Web, including the service name, service description and other information.

2) Use natural language processing technique to extract the service goal set from service documents.

2474-2473/17 $31.00 © 2017 IEEE
DOI 10.1109/SCC.2017.41

3) Calculate service similarity based on service goal set.
4) Cluster services based on the K-means algorithm.

Step 1 was done by web crawling. We will introduce step 2, 3, 4 in detail in the following.

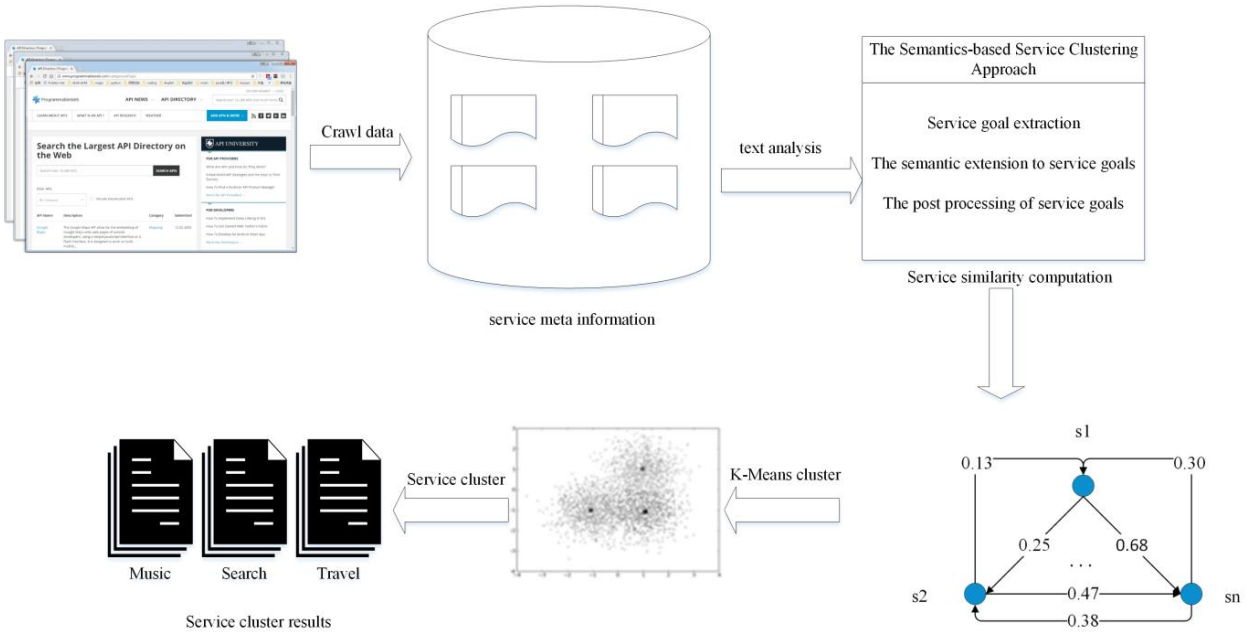

Figure 1. Semantic-based service clustering framework

A. Service Goal Extraction

The meta information of the service, containing the service description and explanation, are used to help service users quickly and intuitively understand the service functionalities. They are usually documented in natural language.

In this step, the natural language processing tool – Stanford Parser – is utilized to work out the grammatical structure of each sentence of service descriptions. It performs two main tasks: (1) identify and assign Part-of-Speech (POS) tags to the word in a sentence; (2) create grammatical relations or type dependencies among elements in a sentence (which are called as *Stanford Dependencies*). Stanford Parser works as follows:

Step 1: A tokenization splits up the text into individual sentences. Each sentence is parsed by Stanford Parser to obtain Stanford Dependencies (SD) Set.

Step 2: The specific Stanford Dependencies will be analyzed to get Verb and Object which can comprise initial service goals.

Step 3: Detect the Conjunction relationship in the Stanford Dependencies and find out whether there are potential service goals in sentences.

Step 4: Lemmatize the words in initial service goals using WordNet and generate service goal set finally.

A service goal is generally described as a verb and a set of related parameters of phrase. The most important parameter is generally a noun or noun phrase, which is used to indicate the entity affected by the service goal. Thus a basic service goal can be represented in terms of Verb + Object. The definition of the service goal is given as follows. Service Goal is defined as a binary relation: (p(verb), p(object)). The p(verb) is a verb or a verb phrase, which represents an executable action, whereas the p(object) is a noun or a noun phrase, which represents an operational objective in a service.

The service goal is extracted via a service goal extraction algorithm as below.

We defined the service goal as sg= <action,object>, where action is a verb, which denotes the action of the service goal, object is a noun , which denotes the entities affected by the action.

Algorithm 1. Service Goal Extraction Algorithm

Input: An analyzed SD set D from one sentence and stopwords list S

Output: The service goal set N in this sentence

1. Initialize the service goal set N
2. for sd $\in D$
3. if sd belongs to the four scenarios
4. sg $\leftarrow$ generate(sd) //generate the initial service goal
5. if sg needs semantic extension
6. esg $\leftarrow$ extendSG(sg) //extend the semantic of initial service goal
7. $N \leftarrow$ esg
8. else $N \leftarrow$ sg
9. end if
10. end for
11. for sg $\in N$
12. if p(verb) $\in S$ // if the verb belongs to the stopword list

13. delete this sg
14. else lemmatize the words in the service goal
15. $N \leftarrow$ sg
16. return

The SD set is a set of grammatical relations parsed by Stanford Parser. The current representation contains 50 grammatical relations, each SD is a binary relation: SDType(governor, dependent), a grammatical relation holds between a governor and a dependent. For instance, in relation dobj(make, products), the noun products is the direct object of the verb make. In this section, we identify four scenarios to obtain the initial service goals by analyzing the SD set.

Scenario 1: Relation dobj(governor, dependent) generally appears in active voice, in which the governor is a verb, and the dependent is a noun or noun phrase as the direct object of governor. In this case, we can obtain the initial service goal directly:

dobj(governor, dependent) → (p(governor),p(dependent))

Scenario 2: Relation nsubjpass(governor, dependent) appears in passive voice, in which the governor is a verb, and the dependent is a noun or noun phrase as the syntactic subject of a passive clause. In this case, we can obtain the initial service goal directly:

nsubjpass(governor, dependent) → (p(governor), p(dependent))

IF prep_p(governor) = nsubj(governor)

THEN prep_p(governor, dependent) → (p(governor), p(dependent))

Scenario 3: Combine relation prep_p(governor, dependent) with nsubj(governor, dependent): when the Verb part in a service goal is a verb phrase contain a preposition, then we can obtain the service goal through the relation prep_p. However, not every relation prep_p can extract the service goal correctly, only when the governor in prep_p is the main verb in clause which can be determined by analyzing the relation nsubj. In this case, we can obtain the initial service goal as follows:

IF prep_p(governor) = nsubj(governor)

THEN prep_p(governor, dependent) → (p(governor), p(dependent))

For instance, for the sentence "The department will search for more music information" can obtain SD relations nsubj(search, department), prep_for(search, information). So, we can obtain initial service goal (search, information).

Scenario 4: Use the relation conj(governor, dependent) to find out more potential service goals. This case generally occurs when a sentence contains multiple service goals in conjunction. For the initial service goal (p(verb), p(object)), if there is a conj(p(verb), dependent), then we can obtain the service goal (p(dependent), p(object)); if there is a conj(governor, p(object)), then we can obtain the service goal (p(verb), p(governor)).

For instance, for the sentence "The department will search for more music information" can obtain SD relations nsubj(search, department), prep_for(search, information). So, we can obtain initial service goal (search, information).

Scenario 4: For the initial service goal (p(verb), p(object)), if there is a conj(p(verb), dependent), then we can obtain the service goal (p(dependent), p(object)); if there is a conj(governor, p(object)), then we can obtain the service goal (p(verb), p(governor)).

For instance, for the sentence "we will inspect and authorize the request" can obtain SD relations conj_and(inspect, authorize), dobj(inspect, request). So, we can obtain initial service goals {(inspect, request); (authorize, request)}. The basic service goals extracted above may have lost their functionality semantics. For example, for the sentence "The user can search for music information." can be resolved to the basic service goal {search information}. But the service goal we expect is {search music information}. Therefore, the semantic extension of basic service goal is necessary.

The semantic extension mainly refers to the noun part of service goals and the semantic extension of nouns mainly includes qualifiers, adjectives, nouns, gerunds and adverbs. Through the analysis of description, we found that qualifiers, adverbs do not provide the semantic information we need and only a few adjectives contain useful semantic information. Therefore, we only consider nouns and gerunds as modifiers for semantically extending service goals. In the Stanford Parser, we mainly consider the *nn(governor, dependent)* relationship, which indicates that both the governor and dependent are nouns, and dependent is treated as a modifier to modify the governor. Finally, we created a stopword list to remove those meaningless service goals which contain weak verbs.

The service goal is extracted based on the four scenarios via a service goal extraction algorithm that has been developed by us in [18].

B. Service Similarity Computation

Similarity between each service goals is used as the benchmark of cluster analysis. Since the number of service goals extracted from different service description documents is different, we use Jacard Similarity Coefficient to calculate the similarity between two services. Jacard similarity coefficient is calculated as follows:

$$J(E,F) = \frac{|E \bigcap F|}{|E \bigcup F|} \tag{1}$$

In the formula, E and F are two given sets. The service similarity calculation formula is defined as follows:

$$S_s(s_1, s_2) = \frac{\sum_{i=0}^{m} S_g(g_{i1}, g_{i2})}{n} \tag{2}$$

Among them, m is the number of service goals of a service goal set, which contains the fewer service goals in s_1 and s_2 set. n is the number of service goals in the set with

more goals. $S_g(g_{i1}, g_{i2})$ denotes the similarity of the service goals contained in the set s_1 and s_2 .

The service similarity is defined as follows:

Definition 1. (Service similarity). V_1 and V_2 respectively denote the verbs in the service goal g_1 and g_2; N_{i1} and N_{i2} respectively denote terms in the service goal g_1 and g_2; w_1, w_2 denote the weight of the verb part and the noun part. m is the number of nouns of the service goal g_1 or g_2 with fewer nouns, whereas n is the number of nouns of a service goal g_1 or g_2 with more nouns. The service similarity calculation formula is:

$$S_g(g_1, g_2) = w_1 \times S_w(V_1, V_2) + w_2 \frac{\sum_{i=1}^{m} S_w(N_{i1}, N_{i2})}{n} \quad (3)$$

In the formula, $S_w(N_{i1}, N_{i2})$ denotes the similarity of two words and the definition of word similarity is as follows:

Definition 2. (Word Similarity) [8]. $F(w)$ denotes the feature set owned by word w. $F(w)$ can be considered as a description of the word w. Definition $I(S)$ represents the number of goals contained in feature set S, The similarity between the two words is calculated as:

$$S_w(w_1, w_2) = \frac{2 \times I(F(w_1) \bigcap F(w_2))}{I(F(w_1)) + I(F(w_2))} \quad (4)$$

In the formula, $I(S)$ is calculated as:

$$I(S) = -\sum_{f \in S} \log P(f) \quad (5)$$

$P(f)$ represents the occurrence probability of feature f. When two words have the same feature set, then the maximum similarity is 1. The minimum similarity is 0 when the intersection of two words' features is empty.

The calculation of the word similarity is based on the WordNet, which is a vocabulary database and organizing vocabulary information according to its semantics. The feature set $F(w)$ based on WordNet will contains the synonym set, generic word and interpretation of the word w; $P(f)$ is estimated by calculating the occurrence probability of feature f in the entire WordNet library database.

C. K-means clustering based on service similarity

Data clustering refers to the process of dividing data into different aggregation classes according to its property. Elements belonging to the same aggregation class encompass the same properties as many as possible, whereas elements belonging to different aggregation classes vary significantly. There are a lot of clustering algorithms for data clustering that have been proposed in the past years, which can be divided into the partition clustering algorithm, hierarchical clustering algorithm, density-based clustering algorithm, and graph theory clustering method. Although so many methods are proposed, the K-means algorithm is widely used, especially in the application of image and voice data compression, and task decomposition in heterogeneous neural networks [9].

The traditional K-means algorithm is a classical algorithm which is based on the distance, which is widely used in the field of data mining. The key idea is to divide the data into k clusters to ensure that every two elements in each cluster have a high similarity, and the similarity between every two clusters' elements is small. In the traditional K-means algorithm, the distance is usually calculated by Euclidean distance or Manhattan distance. The closer the two elements are, the higher their similarity is. In this paper, we propose a K-means algorithm based on service similarity for service clustering as follows:

Algorithm 2. K-means algorithm based on service similarity.

Input: service goal set S_G, cluster number k.

Output: k clusters

1. **Initialize** k clusters C_i $(i = 0 \ldots k)$
2. Choose k services randomly from S_G as the initial center point of k clusters
3. **While** the results of the last two clustering are not the same
4. {
5. Calculate the similarity of each service $s \in S_{Gi}$ to these clustering center points, and save to the array similarity[k];
6. Get the index corresponding to max(similarity[k]), and put the service into clusters corresponding to the index.
7. Reset the center point in each cluster
8. }
9. output k clusters
10. **Return**

III. CASE STUDY

In order to illustrate and evaluate the approach, we use the Mashup application services and API services on the ProgrammableWeb website as examples for the experimentation and analysis.

A.Source of the dataset

ProgrammableWeb is currently one of the most famous API service and Mashup application service registration directory site, which provides a large number of Web-based and mobile applications API information [6]. Up to now, ProgrammableWeb has listed more than 10,000 API services and more than 6,000 mashup application services. Mashup application service is composed of one or more API services, which can be regarded as coarse-grained services. For each API and mashup service, ProgrammableWeb provides its services name, description, label, service provider and other information. For the Mashup application service, Programm-ableWeb provides an API combination list.

To obtain the data we need, we use the crawler tool to crawl the names, descriptions, tag information of the APIs and mashup services on the ProgrammableWeb and save them to the local database. During the crawl process, there are two problems arising:

1) The service description or classification is empty;

2) One service has been registered for multiple times.

To solve the first problem, we filter the service data out. To solve the second problem, we only save one service into the database. Eventually, we crawled 6679 documents of API services and 4508 documents of mashups to meet the experimental requirements.

The next step is to syntactically parse the description text of the service documents. As the service information on the ProgrammableWeb is provided by the users, the description of services are usually arbitrary and ill-formed. For example, there are some special characters, such as "/", "€" in the service description. Besides, the descriptions are not in line with syntax standard. These problems will affect the parsing precision. Therefore, we need to do some preprocessing to the service descriptions before extracting them:

1) Remove or replace special characters, such as use "and" to replace "&", use "or" to replace "/", remove "€".In doing so, the semantic of special characters does not change;

2) Improve the grammatical norms without changing the semantics, such as capitalize the first letter of a sentence.

B.Service Goal Set Extraction

By analyzing the description of 6679 APIs and 4508 mashup services, we obtain the corresponding goal sets of these services. Fig. 2 shows the extraction of service goal sets. Among the 6679 API services, there are 6665 non-null services and the extraction rate is 99.8(%). In 4508 mashup services, there are 4010 non-null services, and the extraction rate is 89.0(%). The main reason that the extracted service goal set is null is that the service description on the ProgramableWeb is not standard. By analyzing null services, we find that the description document of these services is short and the statement in the document does not conform to the correct English grammar specification and lacks the basic subject-predicate structure, therefore such services cannot be correctly parsed by Stanford Parser and therefore get the corresponding set of service goal.

In the non-null service, we analyze the results obtained in Fig. 3. API service contains 8 service goals on average, whereas most services contain 5-11 service goals; Mashup service extracted contains 2 service goals on average, whereas most services contain 1-4 service goals. The length of the service description is the major reason which results in the differences. The average number of statements for the API service is 4 and the average number of words is 65. The Mashup service description document has an average number of statements is 1 and an average number of words is 15.

In order to verify the effectiveness of the method of extracting service goals, we randomly selected 20 services as experimental data. We ask three postgraduates to manually extract the service goals of the description, and then automatically extracted the service goals, then we got four different sets of service goals. Finally, we evaluate the experimental results by calculating the precision and the recall rate. The formula precision and recall rates are defined as follows:

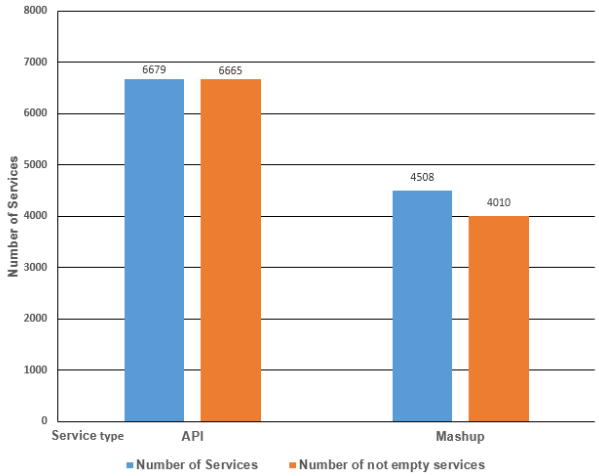

Figure 2. Service goals information extraction.

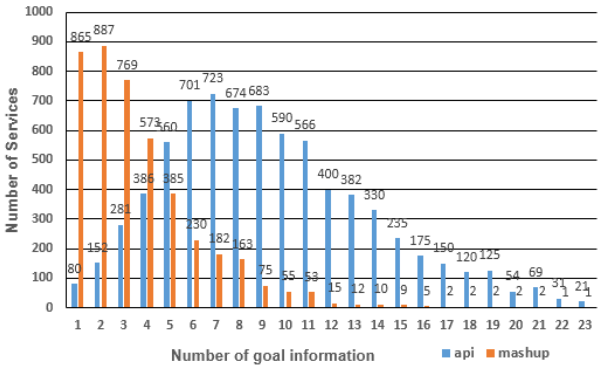

Figure 3. Number of service goals information sets.

$$precision = \frac{|S_{GM} \bigcap S_{GU_i}|}{|S_{GM}|} \tag{6}$$

$$recall = \frac{|S_{GM} \bigcap S_{GU_i}|}{|S_{GU_i}|} \tag{7}$$

$$F_1 = \frac{2 * \mathrm{Pr}ecision * \mathrm{Re}call}{\mathrm{Pr}ecision + \mathrm{Re}call} \tag{8}$$

In the formula, S_{GM} represents the set of service goals that is automatically extracted, S_{GUi} represents the goal set of service extracted manually by the i^{th} graduate student. Table I shows the experimental results of service goal extraction.

TABLE I. EXTRACTED SERVICE GOALS SET RESULTS

Extracted Service Goal Set	Average precision	Average recall rate	Average F-Measure
S_{GU1}	0.704	0.983	0.820
S_{GU2}	0.685	0.951	0.796
S_{GU3}	0.772	0.964	0.857

From Table I, it can be concluded that the results of the service goals extracted by different users are different. The reason is that different users have different understanding for service goals. The recall rate of the extracted results is almost close to 1.0. This shows that the automatically extracted service goal set can cover all the goals of the each service. The precision of the extraction results is lower than the recall rate, between 0.68-0.78. The F-measure result of service clustering between 0.79-0.85. This means that the automatically extracted service goal set still contains some erroneous service goals. The reason for this result may be that the algorithm does not delete some erroneous goals in the extraction process.

C. Automatically Service Clustering

Based on the classification information of the API service on ProgrammableWeb, we selected 311 API services from three categories as the experimental data of service clustering. The three categories are Email (119), Video(117) and Medical (75). The reason for choosing these three categories is that these three categories contain the suitable amount of services: There are 282 categories for the 6679 API services on ProgrammableWeb. The largest service category contains 527 services, whereas the smallest classification contains only one service. Most of categories contain less than 10 services. The use of small-scale categories leads to a low classification precision. Therefore, we randomly select three service categories with moderate number of services as experimental data for service clustering by automatically clustering these services, and then comparing the clustering results with the original classification of services.

By analyzing the goal set of the selected service, we calculate the similarity between all services. Fig. 4 shows the highest similarity value was 0.731, whereas the smallest similarity value of the service similarity is 0. Besides, there are 8944 scenarios in which the similarity is 0 in 96410 services.

We used the similarity value between services and the number of clusters (i.e., k = 3) as the input of K-means clustering algorithm.

Fig. 5 shows we got three clusters from 311 services, which respectively contain 148, 90, 73 services. Table II shows the clustering results. The number of services in each cluster obtained from the algorithm is slightly different from that obtained according to the pre-classified services. In order to further verify the clustering results, we analyzed the clustering results. Table III shows the three central service names for the clustering results, their original classifications and their service goals.

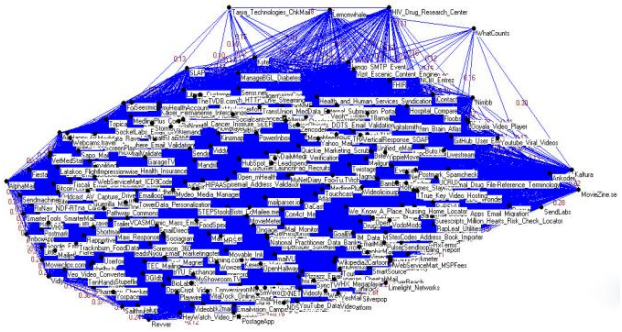

Figure 4. Similarity between services

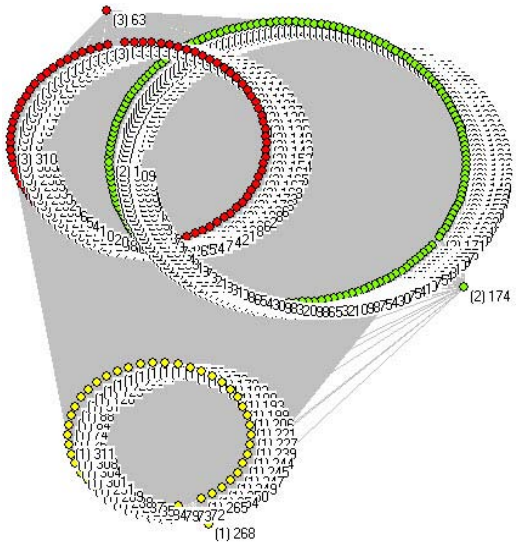

Figure 5. Service clustering results

It is known that the smaller the sum of the distance between each central service and all other services in the cluster is, the larger the similarity is. Therefore, the central service can be used to represent core goal set of the service cluster. In Table III, the service goal set of the central service of Cluster 1 -- Bitcoin-Contact, embodies a lot of functions related to sending and receiving Email. The service goal set of the central service of Cluster 2, embodies a lot of functions related to video operations. The service goal set of the central service of Cluster 3 does not clearly show its classification as the number of services in each classification (i.e. Email, Video, Medical) is very close. After analyzing the cluster and their corresponding classifications, we can find that most of the services in Cluster 1 are closely related to mail exchange and information exchange. Most of the services in Cluster 2 are closely related to video multimedia operations. Most of the services in Cluster 3 are closely related to medical information support and content delivery.

TABLE II. Service Clustering Results

Cluster Results	Central Service of Cluster	Number of Services	Classification		
			Email	*Video*	*Medical*
Cluster 1	Bitcoin-Contact	148	83	35	30
Cluster 2	Magisto	90	22	56	12
Cluster 3	Lexicomp	73	14	26	33

For those services whose descriptions are not clear, it can be found that these services are clustered according to the semantic similarity to the most similar cluster. Such as the Createsend service, its original classification is Email; however, the description and the service goal set did not mention any information related to mail sending and receiving. Instead, the service goal set of the Createsend service is related to user subscriptions and information services. Therefore we classified it to Cluster 3. The original classification of the Aetna CarePass service is Medical, but its service goal set focuses on data exchange for the API information and data access. Therefore we classified it to Cluster 1.

In order to evaluate the algorithm, we choose the Purity of Cluster [10] as a metric to analyze the effectiveness of the approach. The following is the definition of cluster purity.

Definition 3. (Purity of Cluster) [6]. Suppose that D is the set of services to be clustered and C is the result of a clustering on D. $C_k \in C$ denotes the k^{th} cluster in the clustering result, whereas S denotes the standard clustering result on D. $s \in S$ denotes a standard cluster in the standard clustering results. The cluster purity CP (C_k) is defined as:

$$CP(C_k) = \frac{1}{|C_k|}\max(|C_k^s|) \tag{9}$$

In the formula, $|C_k|$ represents the number of the C_k services, and $|C_k^s|$ represents the number of services in the standard cluster s that are divided into the C_k cluster. The standard clustering results in this paper are based on the classification on ProgrammableWeb. On this basis, the cluster purity *CP(C)* of the whole set of service to be clustered is defined as:

$$CP(C) = \sum_{k \in C} \frac{|C_k|}{|D|} CP(C_k) \tag{10}$$

The higher the clustering purity is, the better the clustering results are. Table IV shows the *CP* of the clustering results of three classifications of services, namely Email, Video and Medical. The *CP* of the whole cluster is 55.3(%). Although the *CP* value which is not very high, the results show that the similar services still can be aggregated on demand. The reason is that the service description is inaccurate. Since the classification information of the service and the service description are provided by manual, the error prone information introduces the impression of the clustering results.

TABLE III. CLUSTER CENTER AND SERVICE GOALS SET

Central Service of Cluster	Original Classification	Extracted Service Goal Set
Bitcoin-Contact	Email	{receive message, send message, validate identity, simplify communication, return address ,associate email address, return information}
Magisto	Video	{make video, upload video, edit video, access Magisto functionality, create application, create video, manage video, return video, start video play, stop video play}
Lexicomp	Medical	{offer solution, provide clinician content, make module information, use module information}

IV. RELATED WORK

At present, there are a lot of established work on service clustering. Elgazzar et al. [4] proposed a WSDL document mining method five key features of service functions are extracted from WSDL documents. Liu et al. [5] proposed a text-based service clustering method which uses the vector space model to represent and process the description in the service source file, and then cluster services using the Multi-Hybrid Clustering (MHC) algorithm.

Agglomerative Hierarchical Algorithms are widely used to cluster similar services to improve service discovery efficiency [11][12]. Chen et al. [13] proposed a WTCluster clustering method, which uses the K-means algorithm for service clustering. Wei et al. [14] proposed a WSDL document mining approach by considering structural features and reference features. However, such kind of

TABLE IV. PURITY RATE OF API SERVICE CLUSTERING RESULTS

Cluster Results	Purity (%)
Cluster 1	56.1
Cluster 2	68.5
Cluster 3	77.2

clustering method is usually developed for WSDL or OWL-S documents, which restricts the type of service documents. To our best knowledge, little attention has been paid to RESTful service documents in natural language.

Chen et al. [15] proposed a structurual Web service discovery mechanism based on LDA ((Latent Dirichlet Allocation)). Cassar et al. [16] used PLSA (Probabilistic Latent Semantic Analysis) and LDA to extract the potential topics from the service description, and then cluster services based on the topic of the OWL-S service's Profile description and functional attributes. Zhang et al. [17] proposed an approach to support goal-oriented discovery of RESTful services, which used the LDA model to cluster the services also based on the service goals extracted from the textual descriptions of services. But the the goal recommendation algorithm that they used needs to be improved.

V. CONCLUSIONS AND FUTURE WORK

Service clustering suffers from two problems: 1) the types of service documents are very restricted and 2) the semantic-level clustering are rarely taken into account. To overcome the above two problems, this paper proposes a service clustering method based on semantics. First, this approach extracts the service goal set from NL-based service documents with NLP technique, calculates the service similarity calculation based on the service goal set, and finally uses K-means algorithm to cluster services. API services and Mashup services on ProgrammableWeb are taken as experimental data to evaluate the approach. The experimental results show the effectiveness of the method.

Our work still has a large room for improvement: 1) We will improve the efficiency of service retrieval by precisely matching the corresponding services at the semantic level through the user-provided keywords. 2) We will use other datasets for testing to more accurately evaluate our approach. 3) When extracting service goals, there is still a lot of meaningless information in the service description. 4) We will compare our algorithm with other algorithms and will add other evaluation methods. In addition to NLP, we will investigate other text mining techniques and improve the precision of the service goal extraction.

ACKNOWLEDGMENT

This work is sponsored by National Natural Science Foundation of China under Grant No. 61402406 and No. 61602412, Zhejiang Provincial Natural Science Foundation of China under Grant No. LY13F02010 and No. LY15F02004

REFERENCES

[1] C. Wu, S. Deng, and J. Wu, “Service computing and service technology,” Chinese Journal of Computers, 2009, pp. 1364-1375.

[2] S.A. Mcilraith, T.C. Son, and H. Zeng, “Semantic Web services,” Journal on Intelligent Systems IEEE, 16(2), 2012, pp. 46-53.

[3] Z. Liu, J. Wang, N. Zheng, Z. Li, C. He, and K. He, “A Subject-oriented Domain Service Clustering Approach,” Journal of Computer Research and Development, 51(2), 2014, pp. 408-419.

[4] K. Elgazzar, A.E. Hassan, and P. Martin, “Clustering WSDL Documents to Bootstrap the Discovery of Web Services,” Proc. of Int’l Conf. Web Services, 2010, pp.147-154.

[5] Y. Liu, and Y. Yang, “Semantic Web Service Discovery Based on Text Clustering and Concept Similarity,” Computer Science, 40(11), 2013, pp. 211-214.

[6] W. Pan, B. Li, B. Shao, and P. He, “Research on Service Cluster and Recommendation Approach Based on Software Network,” Computer Science, 34(12), 2011, pp. 2355-2369.

[7] L. Wang, M. Li, S. Cai, G. Li, B. Xie, and F. Yang, “Internet Information Search Based Approach to Enriching Textual Descriptions for Public Web Services,” Journal of Software, 23(6), 2012, pp. 1335-1349. (in Chinese)

[8] D. Lin, “An Information-Theoretic Definition of Similarity,” Proc. of 15th Int’l Conf. on Machine Learning. 1998, pp. 296-304.

[9] W. Wang, and W. Gao, “Text Data Mining on Internet,” Computer Science, 27(4), 2000, pp. 32-36.

[10] Y. Zhao, and G. Karypis, “Criterion functions for document clustering experiments and analysis,” Machine Learning, 2005, pp.311-331.

[11] R. Nayak, and B. Lee, Web Service Discovery with additional Semantics and Clustering. Proc. of Int’l Conf. Web Intelligence, 2007, pp. 555-558.

[12] C. Platzer, F. Rosenberg, and S. Dustdar, “Web service clustering using multidimensional angles as proximity measures,” Acm Transactions on Internet Technology, 9(3), 2009, pp. 11.

[13] L. Chen, L. Hu, Z. Zheng, J. Wu, J. Yin, and Y. Li et al. “WTCluster: Utilizing Tags for Web Services Clustering,” Proc. of Int'l Conf. Service-oriented Computing 2011,pp. 204-218.

[14] D. Wei, T. Wang, and J. Wang, “Web Service Discovery by Integrating Structure and Reference Features of Description Documents,” Journal of Software, 22(9), 2011, pp. 2006-2019.

[15] J. Chen, and J. Yu, “Topic model based structural Web services discovery,” Journal of Beijing University of Aeronautics & Astronautics, 34 (6), 2008, pp.734-738.

[16] G. Cassar, P. Barnaghi, and K. Moessner, “Probabilistic Methods for Service Clustering,” Proc. of Int’l Workshop Service Matchmaking & Resource Retrieval, 2010.

[17] N. Zhang, J. Wang, K. He, and Z. Li, “An Approach of Service Discovery Based on Service Goal Clustering,” Proc. of Int’l Conf. Services Computing, 2016, pp. 114-121.

[18] Y. Wang, B. Jiang, and T. Wang, “Using workflow patterns to modeling and validating service requirements,” Proc. of Int’l Workshop on Requirements Patterns, Beijing, China, 2016, pp. 281-288.