An Approach for Semantic Web Discovery Using Unsupervised Learning Algorithms

Yan Shen and Fangfang Liu(✉)

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
ffliu@shu.edu.cn

Abstract. With the huge amount of available web services, it becomes increasingly difficult to find target web services for users accurately and effectively. For this reason, research in web service clustering has recently gained much attention. Most existing clustering methods perform well when dealing with long text documents. However, the textdescriptions of web services are in the form of short text. Meanwhile, it is meaningful to consider word order information in the textdescriptions of web services. Hence, we presented a service discovery approach based on web service clustering considering this issue. In our approach, web service discovery was divided into two parts: web service clustering and web service selection. In the process of web service clustering, the textdescriptions of web services were represented as vectors. In order to make vector representations reflect as much as possible the semantic information contained in the web service textdescriptions, we tried four different unsupervised sentence representations. In another part, LDA was used to mine topic semantic information of web services after user's web request was placed into a specific cluster according to its web service textdescription vector. The final efficiency of web service discovery was used to measure the effectiveness of our approach.

Keywords: Web service discovery · Web service description · Unsupervised learning algorithm · Web services clustering · Semantic information

1 Introduction

With the development of web technologies, the number of web services is increasing rapidly. Hence, finding suitable web services for users quickly and efficiently has become the content of many scholars. In recent years, the development of semantic web and machine learning algorithms have injected a lot of new vitality into the field of web service discovery [1–4]. Semantic web makes it possible to access web resources by content rather than just by keywords. OWL-S is a set of mark up language constructs that can be used to define properties and capabilities of web services in computer understandable way. It aims at providing an ontological description of web services to facilitate dynamic and automated discovery. Several machine learning algorithms have been applied in the field of web service discovery. However, it is still an open problem and can be further improved.

H. Ning (Ed.): CyberDI 2019/CyberLife 2019, CCIS 1137, pp. 56–72, 2019.
https://doi.org/10.1007/978-981-15-1922-2_4

Using the definition of OWL-S language, each web service has a textdescription attribute, as shown in Fig. 1. This attribute offers the function of web service in the form of a sentence or short text, which provides a short but clear description of web service. The previous methods usually consider this attribute as a lexical vector or further introduce an external knowledge base to expand the lexical vector. However, the word order information in them is also important which contains rich semantic information.

Web service clustering [5] is an important step in web service discovery. A suitable method of web service clustering can improve the efficiency of web service discovery. The relevant web services are able to be returned to users if similar web services are placed into the same cluster. Most existing clustering methods perform well when dealing with long text documents. However, textdescription of web service is in the form of short text. In this paper, we attempt to represent the textdescriptions of web services as vectors for the purpose of more suitable web service clustering.

In this paper, we proposed an approach of semantic web service discovery which considered word order information in the textdescription of web service. The textdescriptions of web services were expressed as vectors which were used for web services clustering. These vectors not only contain the information of words, but also fully consider word order. We tried four different unsupervised vector representation methods to express the textdescriptions of web services. In our approach, web service discovery was divided into two parts: web service clustering and web service selection. In the process of web service clustering, textdescriptions of web services were first represented as vectors. And then these vectors were used to cluster web services by using K-means. In another part, LDA was applied to mine topic semantic information of web service request and each web service in cluster which was located in before. At last, the final web services were selected to be returned to user by calculating the similarity of their document-topic distribution. We measured the effectiveness of our approach through the final web service discovery. More details are shown in Sect. 4.

Our approach needs to answer two critical questions: (i) which unsupervised sentence representation methods are used to represent the textdescriptions of web services, and (ii) what these value of parameters should be, including cluster numbers, topic numbers in LDA and some similarity thresholds.

```
<profile:textDescription xml:lang="en">
This service is a recommended service to know about Heidelburg's, a
</profile:textDescription>
```

Fig. 1. TextDescription attribute of web service.

As to the first question, four different methods were used to represent textdescriptions of web services in our approach, including TF-IDF weighted word2vec, mean of word2vec, skip-thought and doc2vec. All of them are unsupervised sentence representation methods.

In terms of the second question, the best value of parameters, including cluster numbers, topic numbers in LDA and some similarity thresholds, were selected according to the final result of web service discovery. See more details in Sect. 5. Recall and Precision were used as evaluation metrics in our approach, which will be described in detail later.

In the reminder of this paper, we discuss related works in Sect. 2 and introduce some techniques used in our method in Sect. 3, including LDA model and the four different unsupervised sentence vector methods used in our paper. We describe the process of our web service discovery approach in detail in Sect. 4. And then we elaborate our experimental study in Sect. 5. The future work and conclusion are presented in Sect. 6.

2 Related Work

As mentioned before, web service clustering [6] is an effective solution to enhance the performance of web service discovery. Many researchers focus on web service clustering by applying various methods. There are a number of approaches proposed in recent years for web service clustering.

Some relevant works have been presented by using information retrieval techniques. Helmy et al. [7] proposed a text mining approach to automatically group services to specific domains and identify key concepts inside service textual documentation. Heyam et al. [8] presented an intensive investigation on the impact of incorporating feature selection methods (filter and wrapper) on the performance of four state-of-the-art machine learning classifiers. The purpose of employing feature selection is to find a subset of features that maximizes classification accuracy and improves the speed of traditional machine learning classifiers. Similarly, Elgazzar et al. [9] cluster web services based on content, types, messages, ports, and service name from WSDL documents. Approaches based on matrix factorization are also adopted to find relations between services and operations by co-clustering them since the duality relationship can help achieve better quality.

In order to enhance the accuracy of web services clustering, external knowledge and predefined ontologies are also applied in many existing works, which are utilized to compute the semantic similarity between Web services. More specifically, Pop et al. [10] proposed an approach of web services clustering by using an ant-based method based on semantic similarity. Du et al. [11] grouped web services into functionally similar service clusters by calculating semantic similarity with wordnet. Towards the sparseness of useful information in web services, Tian et al. [12] proposed a web service clustering approach based on transfer learning from auxiliary long text data obtained from Wikipedia. And they presented a topic model in order to handle the inconsistencies in semantics and topics between service descriptions and auxiliary data. Due to depend on ontologies and external knowledge, these approaches can accurately cluster services.

In addition, probabilistic topic model Latent Dirichlet Allocation (LDA) receives the attention of some scholars due to it can extract latent topic features of web services. Chen et al. [13] proposed an approach that integrated tagging data and WSDL

documents through augmented Latent Dirichlet Allocation (LDA). And further, three strategies were presented to preprocess tagging data before they were integrated into the LDA framework for clustering. Similar utilizing both WSDL documents and tags, Wu et al. [6] proposed a hybrid Web service clustering strategy which extracted five features from WSDL documents and computing the WSDL-level similarities and the tag-level similarities among Web services. WSDL-level similarity and tag-level similarity were combined into composite similarities for clustering Web services. Shi et al. [14] proposed an augmented LDA model using the high-quality word vectors which were obtained by Word2vec and then clustered into word clusters by K-means algorithm. These word clusters were integrated into the LDA training process. The results of experiment showed their approach had an average improvement of the clustering accuracy with metrics they used.

Zhao et al. [15] proposed a clustering method based on the heterogeneous service network model considering the relationship between service, user and provider for web service classification to improve the accuracy of service recommendation. Based on ontology and SVM, Rupasingha et al. [16] proposed a hybrid approach combining the summary of hybrid term similarity (HTS) and summary of context-aware similarity (CAS) methods to cluster WSDL files. SVM was used to calculate the semantic similarity in a generated ontology of web services. Liu et al. [17] used the Naive Bayes model to classify web services which can enhance service classification accuracy.

3 The Techniques Used in Our Method

3.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [18] is a generative probabilistic model of a corpus based on Bayesian theory, which views documents as bags of words. In LDA [18], documents are represented as a set of latent topics, where each topic is characterized by a distribution over words. LDA [18] generates the distribution of word-document-topic and maps words and documents into a semantic space according to the analysis of the co-occurrence information of words. The training process of the LDA model combined with Gibbs sampling is shown in Fig. 2. The Gibbs sampling formula is shown in [18].

1: random initialization: randomly assign a topic number z to each word w in each document in the corpus.
2: rescan the corpus, resample its topic according to the Gibbs sampling formula for each word w, and update it in the corpus.
3: repeat the resampling process of the above corpus until the Gibbs sampling converges.
4: the LDA model is the topic-word co-occurrence frequency matrix in the statistical corpus.

Fig. 2. The training process of the LDA model.

After the training process, the trained LDA model can be used to generate the distribution of topic for new documents following the steps of Fig. 3.

1: random initialization: randomly assign a topic number z to each word w in current document.
2: rescan the corpus, resample its topic according to the Gibbs sampling formula for each word w.
3: repeat the resampling process until the Gibbs sampling converges.
4: obtain the topic distribution of current document.

Fig. 3. The inference process of the LDA model.

3.2 Word2vec

Word2vec [19] produces a distributed representation of words learned by two-layer neural networks. Word2vec takes a large amount of text as its input and produces a vector space, usually several hundred dimensions, in which each unique word in the corpus is assigned a corresponding vector. Word vectors are located in the vector space, which make words that share common context in the corpus very close to each other in the space. Two model architectures (CBOW and skip-gram) are proposed for learning continuous vector representations of words from huge data sets. In the CBOW architecture, the model predicts current word from a window of surrounding context words. In the skip-gram architecture, the model uses current word to predict surrounding window of the context word. For word2vec in our approach, we focus on the skip-gram architecture.

3.3 Doc2vec

Inspired by word2vec, doc2vec [20] is proposed to achieve phase-level or sentence-level representations, which can consider the ordering of the words and semantics of the words. Doc2vec is an unsupervised framework that learns continuous distributed vector representations for pieces of texts. Two model architectures (PV-DM and PV-DBOW) are proposed for learning continuous vector representations of texts from huge data sets. The architecture of PV-DM is similar to the architecture of CBOW in word2vec. The only one change is the additional paragraph token that is mapped to a vector and can be thought of as another word. In PV-DM, the paragraph vector is concatenated with several word vector from a paragraph to predict the next word in the given many contexts sampled from the paragraph. Because paragraph vectors are learned from unlabeled data, they can work well for tasks that do not have enough labeled data. The architecture of PV-DBOW is also similar to the skip-gram model in word2vec. In PV-DBOW, the paragraph vector is used to predict the words from the text window.

3.4 Skip-Thought

Skip-thought [21] is an unsupervised learning approach of a generic, distributed sentence encoder. Combining the vocabulary expansion method proposed by [21], the off-the-shelf encoder is proposed to produce highly generic sentence representations. And further, the experimental result in [21] shows these generic sentence representations are

robust and perform well in practice. The skip-thought model is based on the skip-gram model in word2vec model, which encodes a sentence to predict the sentences around it instead of using a word to predict its surrounding context. A large collection of novels written by yet unpublished authors were used for training the skip-thought model. Three separate models including uni-skip, bi-skip and combine-skip were trained on dataset. The uni-skip model is an unidirectional encoder with 2400 dimensions and the bi-skip is a bidirectional model with forward and backward encoders of 1200 dimensions each. The combine-skip model consists of the concatenation of the vectors form uni-skip and bi-skip, having 4800 dimensions. After training on the dataset, the learned encoder was used as a feature extractor to extract skip-thought vectors for arbitrary sentences. Experiment shows that combine-skip is better than the uni-skip and bi-skip. Hence, we use the combine-skip model to learn the skip-thought vectors of the textdescriptions in web service.

4 Our Method

In this section, we introduce our web service discovery approach based on web service clustering considering word order information in the textdescriptions of web services. The architecture of our approach is shown in Fig. 4. As shown in Fig. 4, web service discovery is divided into two parts: web service clustering and web service selection. In the process of web service clustering, the owl-s web service documents are first parsed to get the textdescriptions of web services which provide a short but clear description of web services. And four different unsupervised sentence representations are used to represent the textdescriptions of web services, including TF-IDF weighted word2vec, mean of word2vec, skip-thought and doc2vec. TF-IDF is term frequency-inverse document frequency, which is a statistical measure used to evaluate how important a word is to a document in corpus. TF-IDF is composed by two terms: the normalized Term Frequency (TF) and the Inverse Document Frequency (IDF). The Term Frequency (TF) measures how frequently a term occurs in a document. The function of TF is defined as:

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ terms\ in\ the\ document} \tag{1}$$

The Inverse Document Frequency (IDF) measures how important a term is. All terms in documents are considered equally important. As we all known, some terms, such as “am”, “are” and “that”, may appear a lot of times but have little importance. Hence, it is necessary to weigh down the frequent terms while scale up the rare ones. The function of IDF is defined as:

$$IDF(t) = log\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ corpus} \tag{2}$$

And the function of TF-IDF is defined as:

$$TF\text{-}IDF(t) = TF(t) * IDF(t) \tag{3}$$

After obtaining the representing vectors for the textdescriptions of web services, they are clustered using K-Means algorithm. And the clusters of web services are obtained at this stage. When a web service request coming, it can be quickly located in a particular service cluster by calculating the similarity of the vector of textdescription between web service request and cluster centers. In this way, web service selection can be performed only in the specific cluster, which can greatly reduce the selection space and improve the speed of overall web service discovery.

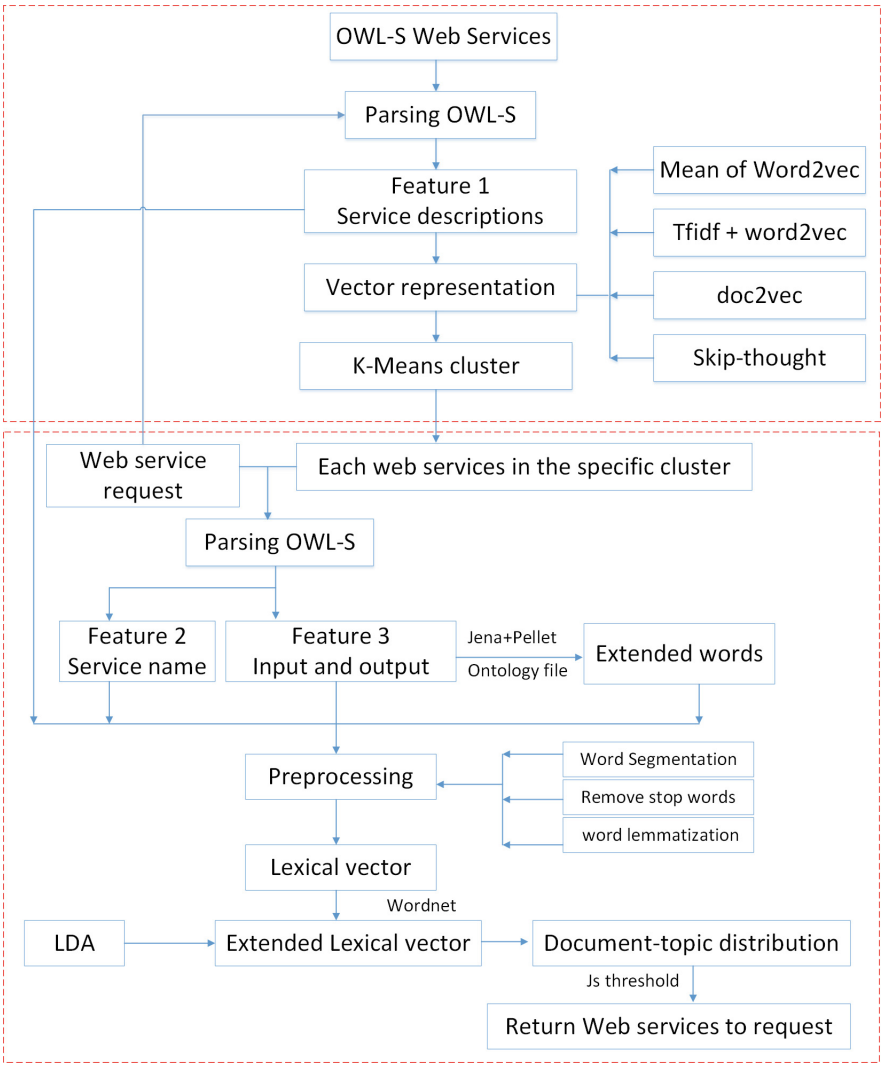

Fig. 4. The architecture of our approach

In the process of web services selection, the selection operation is performed only in the specific cluster. For all web services in the specific cluster, the owl-s web service documents are first parsed to get the information of service name, input and output. For input and output vocabularies, their parent class vocabularies, ancestor class vocabularies and subclass vocabularies are reasoned by using relevant ontology file. Service descriptions which are obtained in the process of web service clustering and service name undergo a series of pre-processing operations, including word segmentation, word lemmatization and removing stop words etc., to obtain the initial vocabulary vector. And then these initial vocabulary vectors are expanded by the wordnet dictionary, adding the top 10 vocabularies of each vocabulary. The expanded vocabularies, input vocabularies, output vocabularies, and the reasoned ontology vocabularies are combined to obtain the final lexical vector. LDA is used to mine topic semantic information according to the final lexical vector of web services in the specific cluster. Finally, we calculate the similarity of document-topic distribution between web service request and web services in the specific cluster. At last, some web services are returned to the web service request which meet a certain threshold.

5 Experiment

5.1 Experiment Data

OWLS-TC4 is the fourth version of the OWL-S service retrieval test collection which is intended to support the evaluation of the performance of OWL-S service matchmaking algorithms. OWLS-TC4 provides 1083 semantic web services and 42 test queries written in OWL-S 1.1 from nine different domains (education, food, medical care, economy, weapons, simulation, travel, geography and communication). The collection also provides ontology files used by web services and service requests. And all relevance sets created for OWLS-TC4 in test collection are available. These relevance sets are used to measure the effectiveness of our approach.

5.2 Evaluation Metrics

As mentioned before, web service discovery is divided into two parts: web service clustering and web service selection in our approach. In the process of web service clustering, we need to calculate the similarity between the vectors of web service textdescriptions and the similarity between web service requests and cluster centers. We use euclidean distance to measure this similarity. The euclidean distance between $\mathrm{x} = (\mathrm{x}_1,\ \mathrm{x}_2 \ldots, \mathrm{x}_\mathrm{n})$ and $\mathrm{y} = (\mathrm{y}_1,\ \mathrm{y}_2 \ldots, \mathrm{y}_\mathrm{n})$ is defined as follows:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \tag{4}$$

In the process of web services selection, LDA is used to mine topic semantic information. Hence, Jensen-Shannon (JS) is used to measure the similarity between the

document-topic distribution. The JS similarity between two probability distributions P_1 and P_2 is defined as follows:

$$JS(P_1||P_2) = \frac{1}{2}KL(P_1||\frac{P_1+P_2}{2}) + \frac{1}{2}KL(P_2||\frac{P_1+P_2}{2}) \tag{5}$$

And KL is defined as follows:

$$D_{KL}(P||\mathrm{Q}) = \sum\nolimits_{x\in X} P(x)log\frac{P(x)}{Q(x)} \tag{6}$$

And for evaluation, we use Recall, Precision, F1 to measure the final web service discovery in our approach. These evaluation indicators are defined as follows:

$$Precision = \frac{TP}{TP+FP} \tag{7}$$

$$Recall = \frac{TP}{TP+FN} \tag{8}$$

$$F1 = \frac{2*Precision*Recall}{Precision+Recall} \tag{9}$$

TP is true positive, FP is false positive, FN is false negative. They are elements in confusion matrix. For a binary classification problem, the confusion matrix can be obtained as shown in Table 1. As shown in Table 1, row represents predicted class and column represents actual class.

Table 1. Confusion matrix.

	Positive	Negative
True	True positive (TP)	True negative (TN)
False	False positive (FP)	False negative (FN)

5.3 Baselines

For our approach, it is important to make the vector representations reflect the semantic information contained in the web service textdescriptions as much as possible. Hence, we measure the performance of the four unsupervised sentence representations (mean of word2vec, TF-IDF weighted word2vec, doc2vec and skip-thought) according to the efficiency of the final web service discovery.

5.4 Implementation

In our approach, K-Means and LDA methods are used to mine semantic information of web services. Hence, the value of some parameters is critical to the efficiency of the

final web service discovery, including cluster numbers, topic numbers in LDA, JS threshold and the parameter values of α and β in LDA. Some experiments were performed to select the optimal values of these parameters according the efficiency of the final web service discovery. In these experiments, skip-thought was used to represent textdescriptions of web service as vectors. Next we will describe these experiments in detail.

Impact of Cluster Numbers (K). In our approach, K-Means is used to cluster vectors of web service textdescriptions. The number of clusters (K) is important for web service clustering and discovery. The best value of K was selected from 5 to 20 according to the efficiency of the final web service discovery. And other parameters were randomly set according to experience. The results are shown in Fig. 5. Obviously, Fig. 5 shows that the precision reach its highest point when the number of clusters K is 17. Therefore, we set K as 17 in our paper.

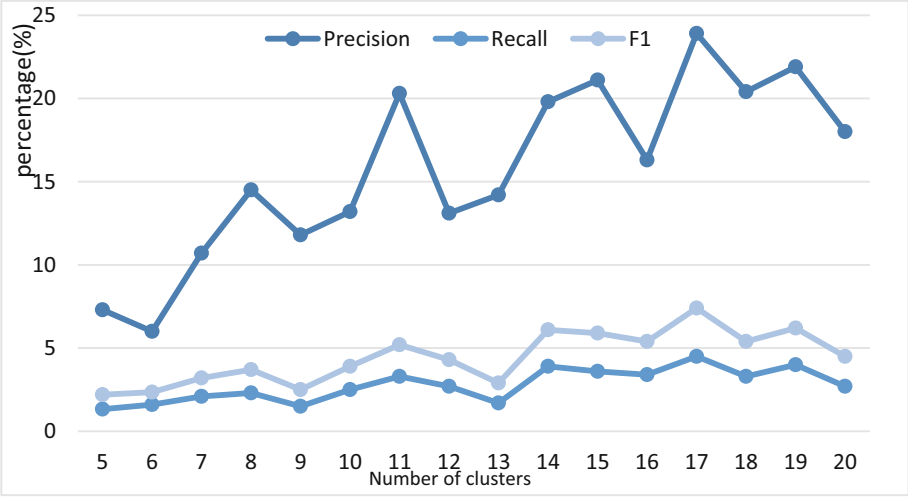

Fig. 5. The result with different cluster numbers (K).

Impact of Topic Numbers (T). In our approach, LDA is used to mine topic semantic information of web services after the user's web request is placed into the specific cluster. The number of topics (T) is important for LDA. The best value of T was selected from 10 to 20 when the number of clusters (K) was 17. And other parameters were randomly set according to experience. The results are shown in Fig. 6. Obviously, Fig. 6 shows that the Precision, Recall and F1 reach their highest point when the number of topics T is 15. Therefore, T was set as 15 in our paper.

Impact of JS Threshold. After the best value of T and K, the best value of JS threshold was selected among (0.01, 0.02, 0.03, 0.04, 0.05). The JS is used to measure the similarity of document-topic distribution between web service request and web services in the specific cluster. And further, web services which satisfy the JS thresh-

old are returned to user request. The results are shown in Fig. 7. As Fig. 7 shown, the performance of Precision, Recall and F1 are better than other values when JS is 0.04 though overall performance changes gently. Hence, JS threshold was set 0.04 in our paper.

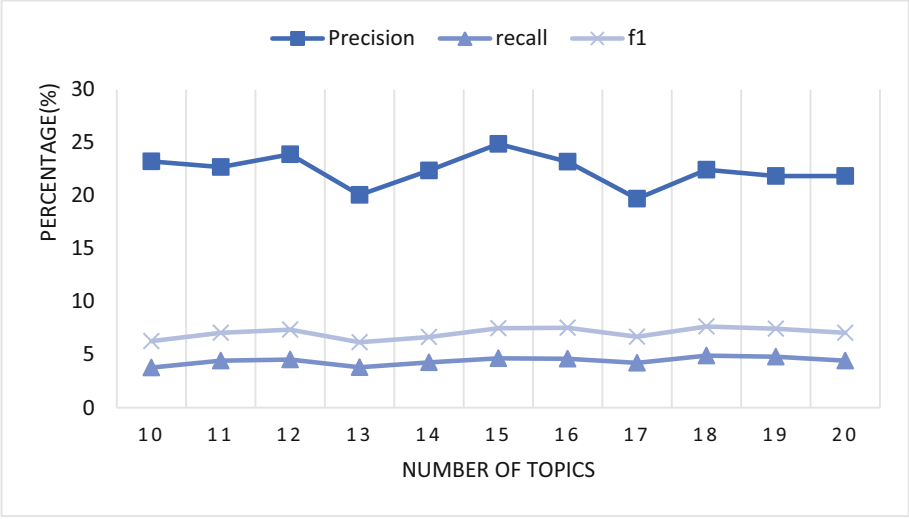

Fig. 6. The result with different topic numbers (T).

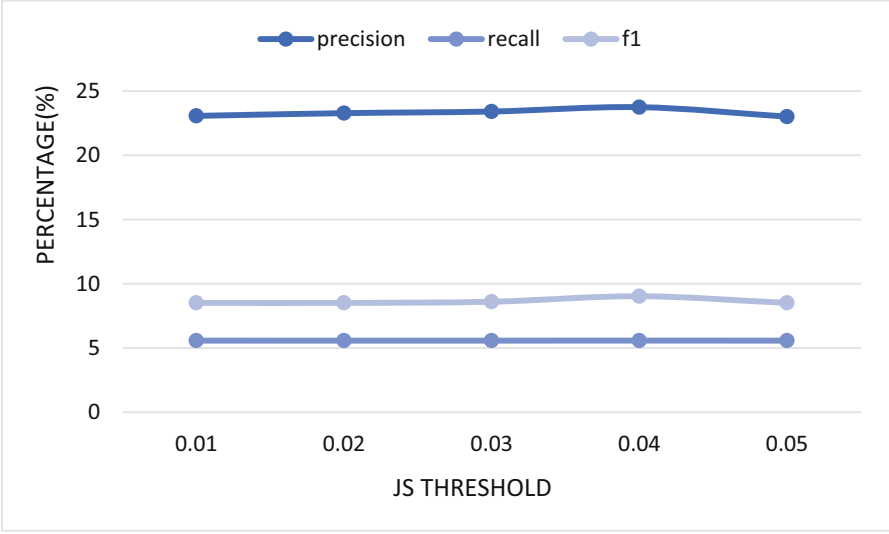

Fig. 7. The result with different JS threshold.

Impact of α and β in LDA. α and β are hyperparameters in LDA. α represents document-topic density and β represents topic-word density. The larger the α value, the more the document is composed of more topics. Similarly, the larger the β value, the more words make up the topic. Hence, the values of α and β are important for LDA. After the best value of T, K and JS threshold, the best value of α and β was selected among {1, 2, 3, 4} and the best value of β was selected among {0.4, 0.5, 0.6}. The results are shown in Fig. 8. As Fig. 8 shown, the precision reach their highest point when the α is 2 and β is 0.5. Therefore, we set α to 2 and set β to 0.5.

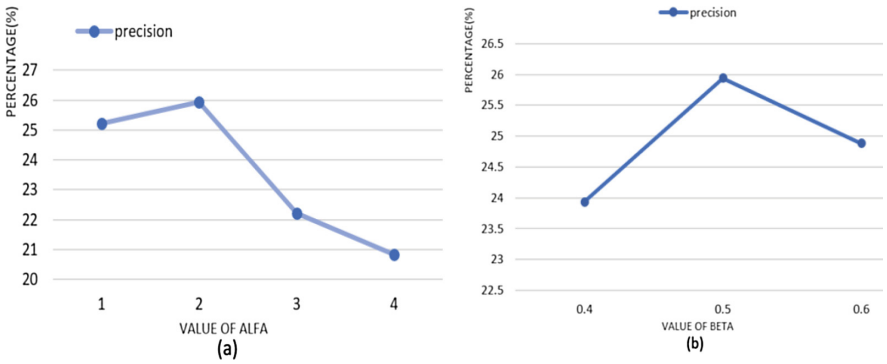

Fig. 8. The result with different α and β.

5.5 Results and Analysis of Web Service Discovery

We performed experiments in accordance with the procedure described in Fig. 4. After preprocessing web services, four different unsupervised sentence representations were used to represent the textdescriptions of web services as vectors. And then K-Means algorithm was performed to cluster these vectors for web service selection. When facing a web service request, LDA was used to mine topic semantic information according to the final lexical vector of web services in the specific cluster. At last, some web services are returned to the web service request which meet a certain threshold. Results on experiments are demonstrated in Figs. 9, 10 and 11. We separately list the each value of Precision, Recall and F1 of the final web service discovery for 42 web service requests which are provided from data collection. The four different unsupervised sentence representations include mean of word2vec, TF-IDF weighted word2vec, doc2vec and skip-thought.

Figures 9, 10 and 11 show that the final performance of different web service requests fluctuates greatly. Although the Precision, Recall and F1 of some web service requests are very good, some web service requests have zero value. And all of the four unsupervised sentence representation methods have zero value. Obviously, the performance of web service discovery using skip-thought is the worst among four unsupervised methods. We believe that this generic sentence representation

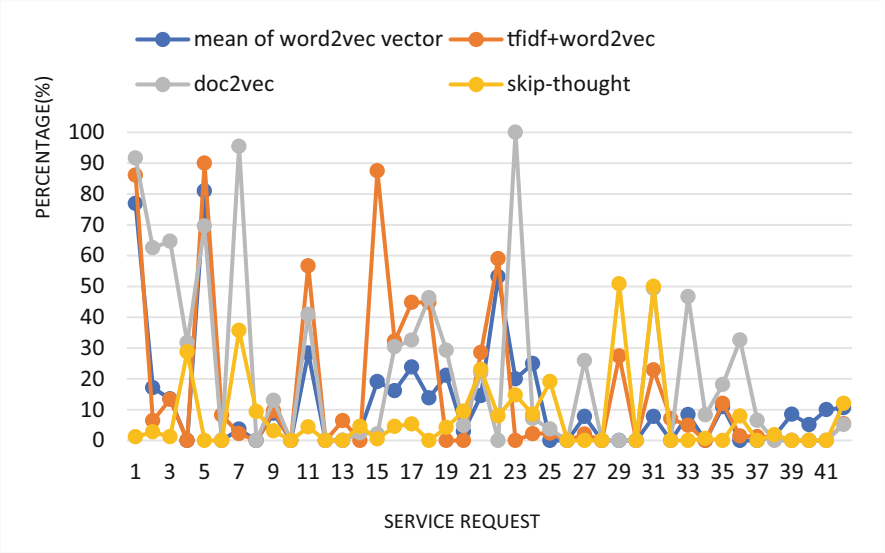

Fig. 9. The result of precision on web service discovery experiment.

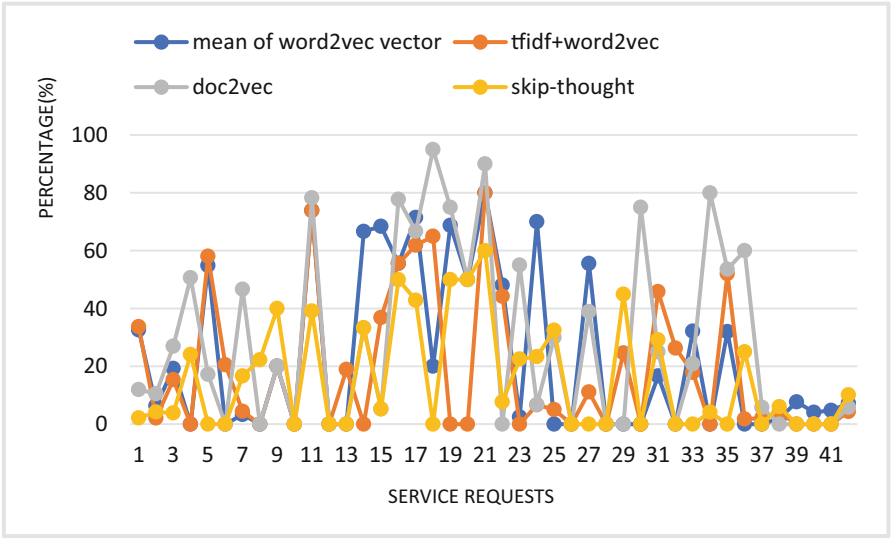

Fig. 10. The result of recall on web service discovery experiment.

skip-thought does not apply to the domain of web service. The skip-thought model was trained with a large collection of novels written by yet unpublished authors. Although vocabulary expansion method in skip-thought model can encode words that not seen as

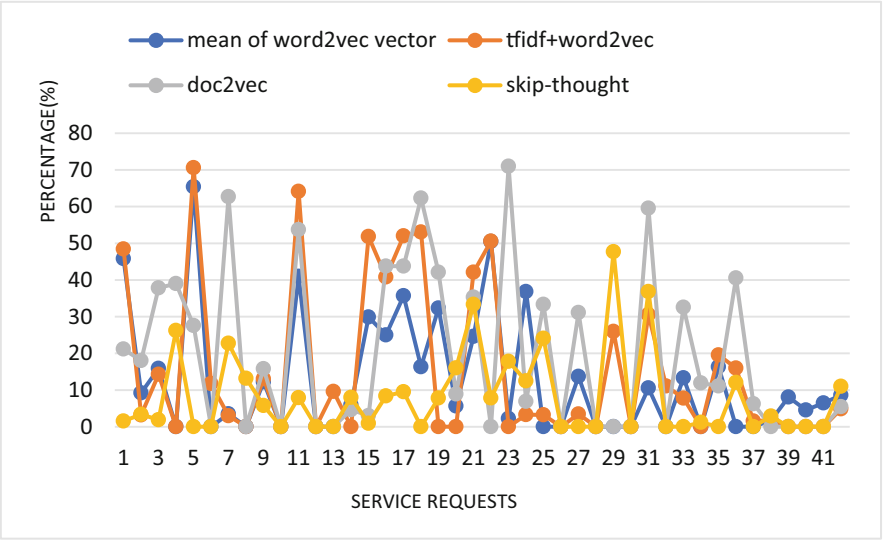

Fig. 11. The result of F1 on web service discovery experiment.

part of training. But the filed of novel and web services is quite different, which makes the vocabulary expansion method not effective. In addition to this, the performance of web service discovery using TF-IDF weighted word2vec is better than using word2vec. We believe that the TF-IDF method can highlight important words and make them have higher weights which are important for the expression of semantic information. We can see that the performance of web service discovery using doc2vec is the best among four unsupervised methods. Because the doc2vec method takes into account word order, the vector representations contain more semantic information than other methods.

In order to further analysis the experimental results, we select some different number of web service request subsets among 42 service requests and use average Precision, average Recall and average F1 to describe the final experimental results. We select four web service request subsets and the number of web service requests is 10, 14, 17, 19 respectively. There are duplicate web service requests in these subsets. Due to the performance of skip-thought is the worst, we only compare the remaining three methods (word2vec, doc2vec and TF-IDF weighted word2vec). Results are demonstrated in Fig. 12. The results shown in Fig. 12 are the same as those we summarized. The performance of web service discovery using doc2vec is the best. Meanwhile, the performance of web service discovery using TF-IDF weighted word2vec is better than using word2vec. Because the doc2vec method can capture the word order information of a sentence, while TF-IDF can highlight important words.

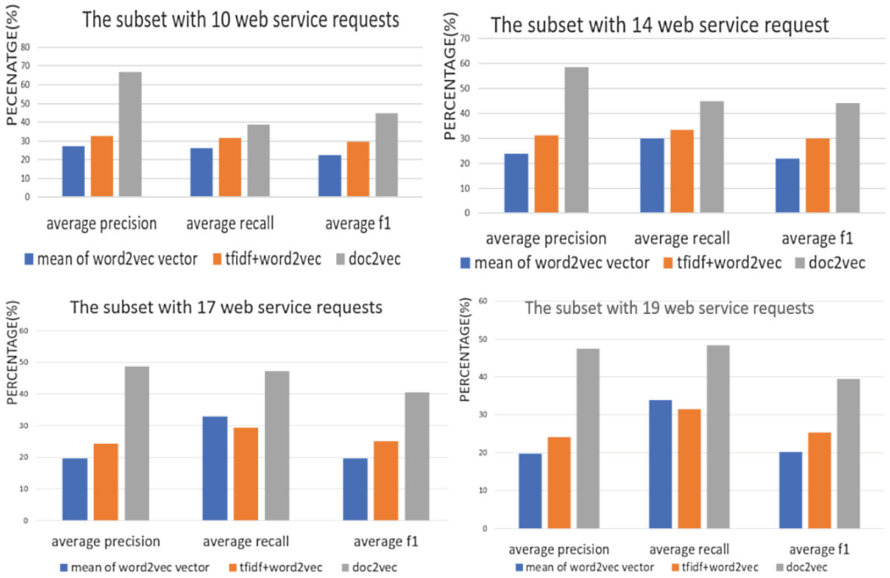

Fig. 12. Results of subsets with different web service requests.

6 Conclusion and Future Work

We proposed a web service discovery approach using unsupervised learning algorithms which is mainly divided into two parts: web service clustering and web service selection. In the process of web service clustering, the textdescriptions of web services are represented as vectors using four unsupervised sentence representation methods, including word2vec, TF-IDF weighted word2vec, doc2vec and skip-thought. Meanwhile, K-Means is used to cluster these vectors. In the process of web service selection, LDA is used to mine topic semantic information in a specific cluster when facing a web service request. At last, some web services are returned to the web service request which meet a certain threshold. In experiment, we selected the best value of K, T, JS threshold, α, β according to the final performance of web service discovery. And then we performed the experiment of whole web service discovery. The results indicate the performance of web service discovery using doc2vec is the best among TF-IDF weighted word2vec, doc2vec, word2vec and skip-thought. The performance of TF-IDF weighted word2vec is better than word2vec. The reason we analyzed is that doc2vec can capture the word order information of a sentence and TF-IDF can highlight important words. Meanwhile, the performance of web service discovery using skip-thought is the worst. The reason we analyzed is that the training data of skip-thought is mainly focus on novels rather than words in web service fields. Hence, this generic sentence representation model skip-thought model does not apply to the web service discovery.

In future, we will concentrate on reducing web service discovery time and discovering unsupervised sentence representations which are more suitable for web

services. Meanwhile, semantic web service described by owl-s also contains a lot of other information. In this paper, we only use the common information (service name, service description, input and output) in web services. Hence, we will focus on adding other useful information in web service to improve the efficiency of service discovery.

References

1. Mier, P.R., Pedrinaci, C., Lama, M.: An integrated semantic web service discovery and composition framework. IEEE Trans. Serv. Comput. **2015**, 537–550 (2015)
2. Cheng, B., Zhao, S., Li, C.: A web services discovery approach based on mining underlying interface semantics. IEEE Trans. Knowl. Data Eng. **29**(5), 950–962 (2017)
3. Fuzan, C., Chenghua, L., Harris, W.: A semantic similarity measure integrating multiple conceptual relationships for web service discovery. Expert Syst. Appl. **67**, 19–31 (2017)
4. Pawar, S., Chiplunkar, N.: Discovery and invocation of web services using multi-dimensional data model with WSDL. Indian J. Sci. Technol. **10**(17), 1–11 (2017)
5. Helmy, A., Salah, A.I., Geith, M.H.: Web services clustering approaches: a review. IOSR J. Eng. (IOSRJEN) **6**, 38–44 (2016)
6. Wu, J., Chen, L., Zheng, Z.: Clustering web services to facilitate service discovery. Knowl. Inf. Syst. **38**(1), 207–229 (2014)
7. Helmy, A., Geith, M.H.: An enhanced approach for web services clustering using supervised machine learning techniques. Int. J. Sci. Eng. Res. **1**(8), 158–170 (2017)
8. Albaity, H., Alshowiman, N.: Towards effective service discovery using feature selection and supervised learning algorithms. Int. J. Adv. Comput. Sci. Appl. (IJACSA) **10**(5), 191–200 (2019)
9. Elgazzar, K., Hassan, E., Martin, P.: Clustering WSDL documents to bootstrap the discovery of web services. In: International Conference on Web Services, pp. 147–154 (2010)
10. Pop, C.B., Chifu, V.R., Salomie, I., Dinsoreanu, M., David, T., Acretoaie, V.: Semantic web service clustering for efficient discovery using an ant-based method. In: Essaaidi, M., Malgeri, M., Badica, C. (eds.) Intelligent Distributed Computing IV. SCI, vol. 315, pp. 23–33. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15211-5_3
11. Du, Y., Zhang, Y., Zhang, X.: A semantic approach of service clustering and web service discovery. Inf. Technol. J. **12**(5), 967–974 (2013)
12. Tian, G., Wang, J., He, K.: Leveraging auxiliary knowledge for web service clustering. Chin. J. Electron. **25**(5), 858–865 (2016)
13. Chen, L., Wang, Y., Yu, Q., Zheng, Z., Wu, J.: WT-LDA: user tagging augmented LDA for web service clustering. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 162–176. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_12
14. Shi, M., Liu, J., Zhou, D.: WE-LDA: a word embeddings augmented LDA model for web services clustering. In: International Conference on Web Services (ICWS), pp. 9–16 (2017)
15. Zhao, H., Wen, J., Zhao, J.: A new model-based web service clustering algorithm. In: IEEE Region 10 Conference, pp. 3468–3472 (2016)
16. Rupasingha, R., Paik, I., Kumara, B: Calculating web service similarity using ontology learning with machine learning. In: IEEE International Conference on Computational Intelligence and Computing Research, pp. 1–8 (2015)
17. Liu, J., Tian, Z., Liu, P.: An approach of semantic web service classification based on naive bayes. In: IEEE International Conference on Services Computing, pp. 356–362 (2016)

18. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
19. Mikolov, T., Chen, K., Corrado, G.: Efficient estimation of word representations in vector space. In: International Conference on Learning Representations, pp. 1–12 (2013)
20. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
21. Kiros, R., Zhu, Y., Salakhutdinov, R.: Skip-thought vectors. In: Advances in Neural Information Processing Systems, pp. 3294–3302 (2015)