Semantic Web Service Discovery Based on LDA Clustering

Heng Zhao[1], Jing Chen[1], and Lei Xu[2(✉)]

[1] Wuhan Digital Engineering, Wuhan 430205, China
[2] Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China
xlei@nju.edu.cn

Abstract. In recent years, with the exponential growth of Web services, how to find the best Web services quickly, accurately and efficiently from the large Web services becomes an urgent problem in Web service discovery. Based on the previous work, we propose a semantic Web service discovery method based on LDA clustering. Firstly, the OWL-S Web service documents are parsed to obtain the document word vectors. Then these vectors are extended to make the documents more abundant of semantic information. Moreover, these vectors are modeled, trained and inferred to get the Document-Topic distribution, and the Web service documents are clustered. Finally, we search the Web service request records or the Web services clusters to find Web services that meet the requirements. Based on the data sets of OWLS-TC4 and hRESTS-TC3_release2, the experimental results show that our method (LDA plus semantic) has higher accuracy (13.48% and 9.97%), recall (37.39% and 24.26%), F-value (30.46% and 23.58%) when compared with VSM method and LDA method.

Keywords: Web service discovery · Document parsing · Latent Dirichlet Allocation · Clustering · Semantic Web service

1 Introduction

Based on Service Oriented Architecture (SOA), Web service is a distributed calculating model, which realizes service invokes by the interactions among service providers, agencies and requesters. The number of Web services on the Internet increase rapidly. Different Web service providers have different descriptions of resources in Web services, resulting in a large number of Web services with similar functions and different expressions. So it becomes difficult for service requesters to find the required services correctly and efficiently from a large number of Web services.

At present, Web service discovery is the main way to solve the matching of user needs and service functions. Web service discovery first selects the corresponding matching method according to the user's requirements, matches the service stored in the service center, and then selects the target service set that meets the user's requirements. In the process of service discovery, the key step is to compare and analyze the service request of the user and the service description of the service agent one by one, and make a set of matching criteria according to the user demand.

W. Ni et al. (Eds.): WISA 2019, LNCS 11817, pp. 239–250, 2019.
https://doi.org/10.1007/978-3-030-30952-7_25

Web services mainly use WSDL (Web services Description Language) documents to describe services, while WSDL can only describe syntactic information, without functional semantic information, resulting in low precision and recall of service discovery.

Therefore, this paper puts forward a LDA (Latent Dirichlet Allocation) clustering method of semantic Web service discovery, which pretreatments OWL-S (OWL Web Ontology Language for Services) document firstly, then gets the initial term vectors of each document, and extracts and parses the OWL-S document corresponding ontology document. Next, through WordNet and Word2Vec, gets the words vector of the final document, and then uses the LDA model and Gibbs sampling algorithm to get the document-subject distribution, calculates the Jensen-Shannon Divergence (JS Divergence), sets these Web services into several clusters, and lookups query first memo databases have any records of the Web service request, document vocabulary of this query and result set of service discovery are also recorded.

The experiment shows that compared with the VSM (Vector Space Model) method based on TF-IDF, the method using LDA and semantic information has higher accuracy and F-value, especially better recall rates and acceptable efficiency.

2 Related Work

Current researches try to make the existing Web services with automaticity and intelligence, and put forward the concept of semantic Web services. There are mainly three kinds of semantic Web service matching, namely based on logic, based on non-logic, and based on mixed pattern.

Semantic Web service matching based on logic [1–3] can deduce and inference semantic information. The ontology corresponding to the service is obtained through semantic annotation of the Service Profile, and logical concepts and rules are extracted from the ontology. The relationship between concepts and rules are inferred by inference machine, and finally the matching set of service requests is obtained. Paolucci et al. [4] proposed the matching idea. It took the input and output parameters of the service description as the basis for matching, and divided the service matching degree into four categories: exact matching, plug-in matching, subsume inclusion and fail. It predefined formal expressions for each type of matching degree and calculated specific values by the shortest distance of concepts on the semantic classification tree. This matching method can guarantee the inevitability and precision of the results, but the implementation of the system largely depends on the integrity of the inference engine and inference rules adopted, and the flexibility and reliability are insufficient.

Non-logical semantic Web service matching [5, 6] is used to calculate the matching degree of service description between service requesters and service providers. The calculation criteria include syntax similarity, structural graph matching degree, and concept distance in ontology. Compared with the semantic Web service matching based on logic that explicitly utilizes the IOPE (Input, Output, precondition, Effect) semantics defined in logical rules, the semantic Web service matching based on non-logic implicitly utilizes information such as pattern matching, sub-graph matching and term frequency defined in Web service description. Wu et al. [7] proposed a semantic

Web service discovery method based on ontology and lexical semantic similarity, including the construction of Web service ontology corresponding to Web service, the similarity calculation of Web service. Although the method can quantify the service matching degree and improve the accuracy of service matching, it is difficult to propose a similarity calculation function that can adapt to all different application scenarios and requirements. At the same time, how to evaluate the advantages and disadvantages of a similarity calculation function is also a difficulty currently.

Semantic Web service matching based on mixed pattern is a combination of the above two, which can give full play to their respective advantages, reduce the influence brought by the defects, and improve the performance of Web service matching. Many researchers in industry have tried this method [8–11]. Klusch [12] presented a matching model based on logical reasoning and matching model based on similarity matching model OWLS-MX: first get five kinds of compatibility, and build the syntax of the similarity between the I/O, the experimental results show that the method improves the precision of service discovery and recall. Our method is also based on the mixed ways.

3 Parsing and Semantic Extension of Web Services

Web service parsing is an important prerequisite for Web service clustering and discovery, and the parsing results will directly affect the performance of service clustering and discovery. In order to make the document parsing more accurate and effective, combining the characteristics of web services and the semantic Web technology, we proposed a method to make web services parsing and semantic extension, known as PASEBLDA4WS (Parsing and Semantic extension Based on LDA for Web Services), as shown in Fig. 1.

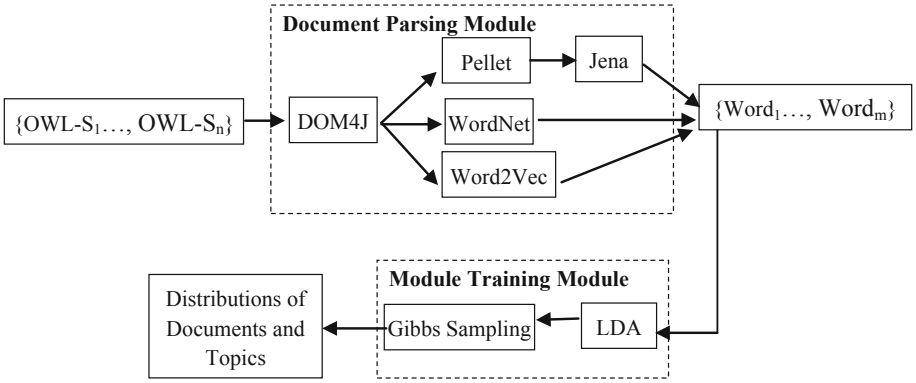

Fig. 1. Framework for PASEBLDA4WS

As can be seen from Fig. 1, the input is the collection of Web service documents (OWL-S_1, OWL-S_2, …, OWL-S_n), and the output is the document-topic distribution. The intermediate process includes two functional modules: document parsing module and model training module.

3.1 Parsing OWL-S Documents

OWL-S document is a kind of semantic annotation for Web services, including Service Profile, Service Model and Service Grounding. The document parsing module mainly completes the functions of OWL-S document parsing and semantic extension of service description. The specific process is as follows:

(1) Parsing the OWL-S file. Tool DOM4J is used to extract the service name, service description, input and output information of OWL-S document, and it conducts text preprocessing such as stop word and part of speech restoration for the service name and service description to obtain the initial vocabulary vector.
(2) Parsing OWL ontology files. By using Pellet inference machine and tool Jena to inference the OWL ontology file corresponding to OWL-S document, the concept vector related to input and output elements are obtained.
(3) Expanding WordNet semantics. Using Algorithm [13] to calculate the words in WordNet, which are more similar to the original word vectors than the threshold value and add them into the extended word vector.
(4) Expanding Word2Vec semantics. Word2Vec is an open-source, deep learning tool that Google launched in 2013 to convert a word into the vector form. After using Word2Vec, the processing of text content can be transformed into vector operation in vector space. In this paper, every article in the Wikipedia data set (version 2016.03.06, size 11.9 g) is converted into a line of text, and the punctuation and other contents are removed to obtain the training corpus. Then, neural network algorithm is used to train the language model and learn the distribution representation of each word. Then, for each word, the most similar word or the most similar word is searched from Wikipedia to expand it, and the top ten words are added to the expansion word vector. The extended word vector of all words is merged to obtain the extended word vector.
(5) Merging all the word vectors in the above steps, so as to get the final document word vector.

3.2 Training Based on LDA Topic Model

The model training module mainly performs the function of referencing document-topic probability distribution. Firstly, the LDA topic model[1] is established for the document vocabulary vector set, and then the Gibbs sampling algorithm is used to iteratively train the vocabulary vector set until the LDA topic model converges, and then the document-topic distribution in the document set is inferred.

[1] Latent Dirichlet allocation, https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation.

The process of using LDA to generate documents in the corpus [14] is shown as follows:

(1) For each topic $k \in \{1, \ldots\ldots, K\}$, sampling generates word distribution for topic k: $\vec{\varphi}_k \backsim Dirichlet\left(\vec{\beta}\right)$;
(2) For each document $m \in \{1, \ldots\ldots, M\}$, sampling generates topic distribution for document m: $\vec{\vartheta}_m \backsim Dirichlet(\vec{\alpha})$;
(3) For each word $n \in \{1, \ldots\ldots, N_m\}$ in document m, the generating process is shown as follows:

Choosing topic: in ϑ_m, sampling generates topic of word n in document m, $\mathcal{Z}_{m,n} \backsim Multinomial\left(\vec{\vartheta}_m\right)$;

Generating a word: sampling generates a word from the chosen topic, $\mathcal{W}_{m,n} \backsim Multinomial(\vec{\varphi}_{\mathcal{Z}_{m,n}})$.

LDA model [14] is shown in Fig. 2. K is the number of topics, M is the number of documents, N_m is the number of words in document m, $\mathcal{W}_{m,n}$ is the n^{th} word in document m, $\vec{\beta}$ is the *Dirichlet* priori argument of multinomial distribution of the word in each topic, $\vec{\alpha}$ is the *Dirichlet* priori argument of multinomial distribution of the topic in each document. $\mathcal{Z}_{m,n}$ is the topic of the n^{th} word in the m^{th} document, $\mathcal{W}_{m,n}$ is the n^{th} word in the m^{th} document. Two latent variables $\vec{\vartheta}_m$ and $\vec{\varphi}_k$ are the distributions of topics under the m^{th} document and words of the k^{th} topic.

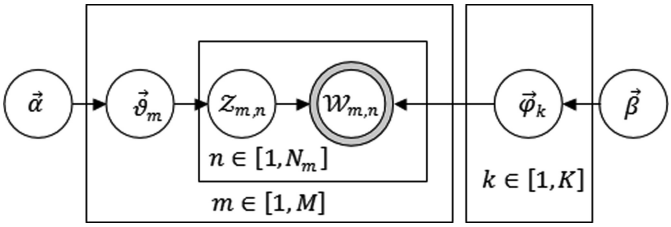

Fig. 2. Topic model of LDA

LDA topic model can extract the core semantics or features of objects from massive corpus. It is a probabilistic model for generating discrete data sets. It maps every document in the corpus to a probability distribution of a group of potential topics, and each potential topic corresponds to a probability distribution of words in the vocabulary set.

Gibbs sampling is a special case of MCMC (Markov Chain Monte Carlo), and the basic idea is as follows: select one dimension of the probability vector each time, sample the value of the current dimension with the variable values of other dimensions, and iterate continuously until the output parameter to be inferred converges.

We use LdaGibbsSampler[2] as the Gibbs sampling algorithm of LDA topic model for training and inference, until the model convergence, then evaluate the parameters of $\mathcal{Z}_{m,n}$, $\vec{\vartheta}_m$ and $\vec{\varphi}_k$, finally get the document-topic distribution of the document vector collection. And the calculating formulas of $\vec{\vartheta}_m$ and $\vec{\varphi}_k$ are shown as follows.

$$\varphi_{k,t} = \frac{n_k^{(t)} + \beta_t}{\sum_{t=1}^{V} n_k^{(t)} + \beta_t} \tag{1}$$

$$\vartheta_{m,k} = \frac{n_m^{(k)} + \alpha_k}{\sum_{k=1}^{K} n_m^{(k)} + \alpha_k} \tag{2}$$

In formula (1) and (2), V is the number of words, K is the number of topics, $n_k^{(t)}$ is the number of word t assigned to topic k, $n_m^{(k)}$ is the number of words assigned to topic k in document m, β_t is the *Dirichlet* priori argument of multinomial distribution of the word t in topic k, α_k is the *Dirichlet* priori argument of multinomial distribution of the topic k in document m.

4 Clustering and Discovery of Semantic Web Services

We present an improved clustering algorithm, which combines LDA model and k-means algorithm. The Jensen-Shannon Divergence based on KL divergence [15] was used as the similarity measurement standard. The calculation formula was as follows:

$$D_{KL}(p||q) = \sum_{i=1}^{n} p_i \ln\left(\frac{p_i}{q_i}\right) \tag{3}$$

$$D_{JS}(p||q) = \frac{1}{2}[D_{KL}(p||\frac{p+q}{2}) + D_{KL}(q||\frac{p+q}{2})] \tag{4}$$

Where D(p‖q) is the KL divergence, representing the information loss generated when the probability distribution p is used to fit the real distribution q, and p represents the real distribution and q represents the fitting distribution of p. The main reason for not using KL divergence as the similarity metric is that KL divergence is not symmetric. The specific clustering process is as follows: firstly, global variables are initialized, including randomly selecting the first initial clustering center; then calculate the distance between each document and the initial cluster center; then a vector with a large distance from all the existing initial clustering centers is calculated as the new initial clustering center. The JS divergence matrix is constructed, and the k-means clustering algorithm is called to obtain the clustering set and cluster center set of Web services.

[2] LdaGibbsSampler, http://web.engr.illinois.edu/~mqian2/upload/projects/java/TextProcessor/ocessor/doc/textProcessor/LdaGibbsSampler.html.

We use a quad to represent the Web service request record, *WSRRecord* =<*RID*, *File-Name*, *ExtWordVec*, *WSResultSet*>, *RID* represents the ID number of the Web service record, *FileName* represents the name of the Web service request document, *ExtWordsVec* represents the document extension vocabulary vector of the Web service request document processed by the document parsing module, and *WSResultSet* represents the set of Web services that the Web service request has met the requirements. Memo database mainly includes query and storage function, which provides a fast and effective way for the query of Web service request.

The query function is used for a Web service request to find a request record for the Web service that already exists in the memo database, including a direct lookup for *FileName* and a matching lookup for *ExtWordVec*. Direct lookup of *FileName* is obtained by precise comparison of the *FileName* in all *WSRRecord* with the name of the document requested by the Web service; if the matching is successful, the set of Web services meeting the requirements can be extracted directly from the database. If it is not successful, then the matching lookup of *ExtWordVec* is required: first through document parsing module for the Web service request document parsing, expand vocabulary get the document vector, then with all the *ExtWordVec* and *WSRRecord* are matched. Since the results of the expansions of lexical semantic information vector after parsing are rich enough, we use the Jaccard's similarity coefficient [16] to measure the two lexical semantic matching degree of the vector, the specific formula is as follows:

$$Jaccard(p,q) = \frac{|p \cap q|}{|p| + |q| - |p \cap q|} \tag{5}$$

In formula (5), the numerator represents the intersection of all words of vector p and q, that is, the number of the same words, and the denominator represents the sum of their words minus the number of the same words, that is, the number of words in their union. And the result is a number inside [0,1].

Web service matching module is a more accurate and complex search in the Web service clustering set, when the Web service set that meets the requirements is not found in the memo DB module. The search is mainly divided into the following two steps: matching the expanded vocabulary vector, and finding the Web service cluster with the largest similarity as the candidate set of Web services; in the Web service candidate set, the semantic distance between the extended vocabulary vector of the Web service request document and all the Web services is calculated, and the Web services whose semantic distances are less than the threshold are added to the Web service result set.

The search and matching process of Web service includes: firstly, initialize the distance between the Web service request and the nearest cluster center; then make the Web service request expanding the vocabulary vector and the cluster center; then calculate the JS divergence of these two vectors, update the minimum distance with the Web service request, and record the cluster center; so the candidate set of Web services is obtained through the cluster center with the smallest Web service request. Computing document-topic distribution of JS divergence with the request, and join these

Web services into the last result set, whose JS divergences are less than the threshold value.

When the corresponding records are not found in the memo DB module, by the Web service matching module, the Web service set that meets the requirements is obtained, and the result set is stored in the memo database in the format of *WSRRecord*.

5 Experiments and Evaluations

This paper conducts experimental evaluation on the proposed semantic Web service discovery method based on LDA clustering, with three research questions:

RQ1: How to determine the number of document topics when conducting Gibbs sampling training and inference on LDA topic model?
RQ2: Is it effective for the semantic Web service discovery method based on LDA clustering?
RQ3: Does the semantic Web service discovery method based on LDA clustering improve the performance of Web service discovery?

We use OWLS-TC4 as a test data set, comprising 1083 Web services in nine different domains. In addition, an associated set of each service query request, more commonly known as the factual basis for the query request, is also included in OWLS-TC4. More than 48 ontology files related to OWL-S files are included in OWLS-TC4, which plays an important role in semantic extension of OWL-S files.

In order to solve RQ1, we start from the fact basis of query request, and set the number of clustering as 9 according to the number of fields (9 fields are preset in the data set, namely $K2 = 9$), and considers the accuracy rate, recall rate and F-value of clustering results when $K1 = \{2,3, \ldots,50\}$, and $K1$ was determined by the experimental results. As shown in Fig. 3, when $K1 = 12$, the recall rate and F-value of clustering reached the maximum value of 0.547497869 and 0.619850749 respectively, and the accuracy reached its fourth value of 0.714238753. Since F-value is taken as the optimal standard, the value of $K1$ is finally chosen as 12.

To solve RQ2, we select the widely used VSM [17] methods, based on TF-IDF, as the contrast experiment. At the same time, in order to test the semantic expansion effect of LDA model and service description, we select the Web service discovery based on LDA clustering and the semantic Web service discovery based on LDA clustering, so as to compare these three methods in accuracies, recalls and F-values, the experimental results are shown in Figs. 4, 5 and 6.

In Figs. 4, 5 and 6, the horizontal axis shows the 42 service request, the longitudinal axis shows each service request under the three methods of accuracies, recalls and F-values, thus it can be seen that in most cases, the method based on the LDA+semantics in accuracy, recall and F-values is better than VSM method based on TF-IDF and the method based on LDA. And there is no value of 0, while the other two methods have the condition of 0value, which means that they cannot find the query results.

To solve RQ3, we compare the running time of the VSM method based on TF-IDF, the method based on LDA and our method, specifically including service parsing time, service clustering time and service matching time.

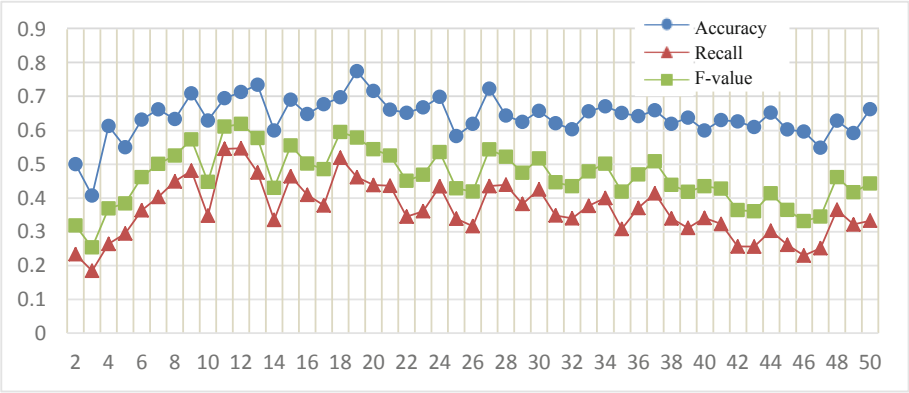

Fig. 3. Clustering effects of Web services with different K1 values (K2 = 9)

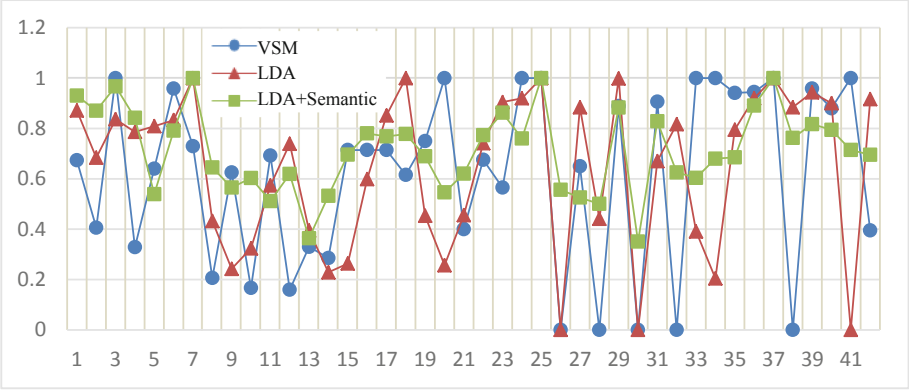

Fig. 4. Comparisons of Accuracies for experimental results

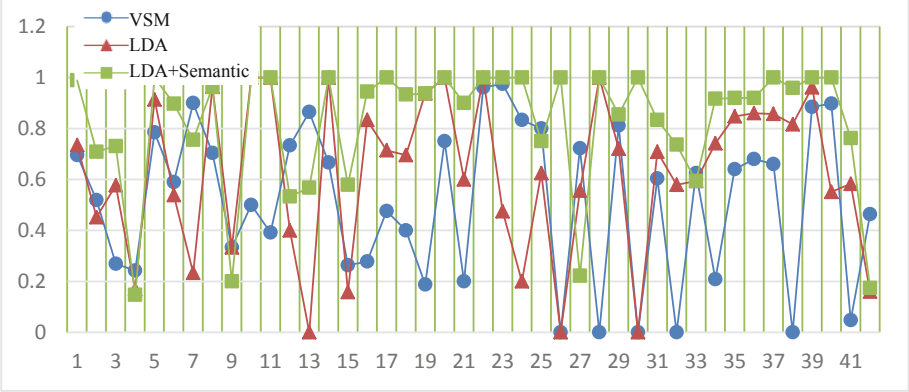

Fig. 5. Comparisons of Recalls for experimental results

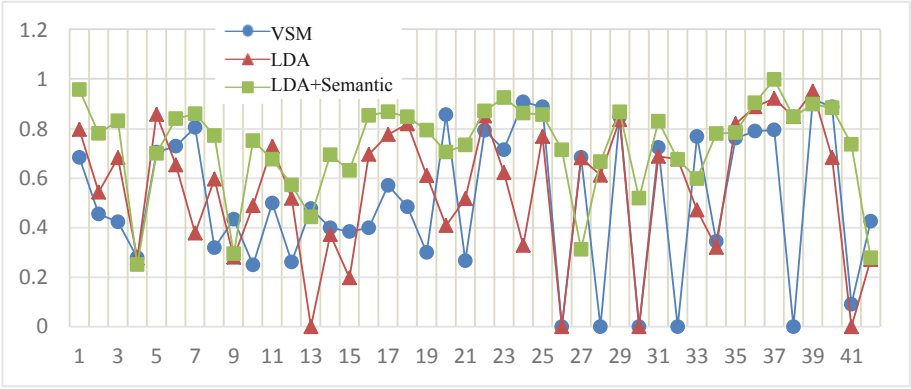

Fig. 6. Comparisons of F-values for experimental results

Within 1083 web services and 42 query request documents, the experimental results of the three methods are as follows: VSM method based on TF-IDF takes 22 min and 20 s; method based on LDA takes 28 min and 6 s; while our method takes 21 min and 13 s. Namely our method takes less time than the other two. LDA-based method takes more time since it needs more time to train and infer the LDA topic model. In addition, when querying all the web service request files, since all querying records are kept in the memo database, the time taken for semantic web service discovery is the same as the time taken for querying the database records, with an average cost of 57 ms, greatly reducing the time spent. Therefore, the semantic web service discovery method based on LDA clustering has a good efficiency. And the supervised learning is also useful in Paper [18] to handle text matching and classifying.

6 Conclusions

Users need to look for web services satisfying their requirements in the vast Web services accurately and efficiently. This paper proposes a kind of semantic Web service discovery method based on clustering of LDA, by parsing of OWL-S service document files, and the corresponding ontology extend Web services description semantics, get the expansion of document word set, then the theme of the LDA model for training and inferences from the document-subject distribution, again to document vocabulary vector clustering, and then find a set of candidate services. Finally, Web services matching the request mostly is searched in the candidate service set. Experimental results show that our method has high accuracy and F-value, and it has obvious advantages in recall rate, its efficiency is also within the acceptable range.

When parsing OWL-S documents, we only consider the description information of web services, ignore the process information and access information. And when analyzing the ontology OWL corresponding to OWL-S file, we only add the direct ontology classes of input-output concepts, sibling ontology classes are not considered. Therefore, we plan to add sibling ontology classes within a certain distance and density

in the concept, and meanwhile reduce the direct ontology classes with distant relatives. And, using WordNet expand service description semantics, semantic similarity threshold values are determined by previous experimental experience value, is not necessarily applicable to the proposed approach, therefore, we plan for these two semantic similarity threshold experiment analysis, get the best results of semantic similarity threshold, improve the efficiency of web service discovery.

Acknowledgment. The work is supported by National Key R&D Program of China (2018YFB1003901), National Natural Science Foundation of China (61832009, 61728203).

References

1. Chakraborty, D., Perich, F., Avancha, S., et al.: DReggie: semantic service discovery for m-Commerce applications. In: 20th Symposium on Reliable Distributed Systems, pp. 28–31 (2001)
2. Zhang, C., Zhao, T., Li, W., et al.: Towards logic-based geospatial feature discovery and integration using web feature service and geospatial semantic web. Int. J. Geogr. Inf. Sci. **24** (6), 903–923 (2010)
3. García, J.M., Ruiz, D., Ruiz-Cortés, A.: Improving semantic Web services discovery using SPARQL-based repository filtering. Web Semant. Sci. Serv. Agents World Wide Web **17**, 12–24 (2012)
4. Paolucci, M., Sycara, K.: Autonomous semantic web services. IEEE Internet Comput. **7**(5), 34–41 (2003)
5. Deng, S.G., Yin, J.W., Li, Y., et al.: A method of semantic Web service discovery based on bipartite graph matching. Chin. J. Comput. **31**(8), 1364–1375 (2008)
6. Bernstein, A., Kiefer, C.: Imprecise RDQL: towards generic retrieval in ontologies using similarity joins. In: Proceedings of the 2006 ACM Symposium on Applied Computing, pp. 1684–1689 (2006)
7. Wu, J., Wu, Z.H., Li, Y., et al.: Web service discovery based on ontology and similarity of words. China J. Comput. **28**(4), 595–602 (2005)
8. Bianchini, D., De Antonellis, V., Melchiori, M., et al.: Semantic-enriched service discovery. In: Proceedings of 22nd International Conference on Data Engineering, p. 38 (2006)
9. Sangers, J., Frasincar, F., Hogenboom, F., et al.: Semantic web service discovery using natural language processing techniques. Expert Syst. Appl. **40**(11), 4660–4671 (2013)
10. Amorim, R., Claro, D.B., Lopes, D., et al.: Improving Web service discovery by a functional and structural approach. In: 20111 IEEE International Conference on Web Services, pp. 411–418 (2011)
11. Paliwal, A.V., Shafiq, B., Vaidya, J., et al.: Semantics-based automated service discovery. IEEE Trans. Serv. Comput. **5**(2), 260–275 (2012)
12. Klusch, M., Fries, B., Sycara, K.: OWLS-MX: a hybrid semantic web service matchmaker for OWL-S services. Web Semant. Sci. Serv. Agents World Wide Web **7**(2), 121–133 (2009)
13. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. In: Proceedings of International Conference Research on Computational Linguistics (ROCLING X), pp. 1–15 (1997)
14. Blei, D., Ng, A., Jordan, M.: Generative probabilistic model for collections of discrete data such as text corpora. J. Mach. Learn. Res. **3**, 993–1022 (2002)

15. Lamberti, P.W., Majtey, A.P., Borras, A., et al.: Metric character of the quantum Jensen-Shannon divergence. Phys. Rev. A **77**(5), 1–8 (2008)
16. Niwattanakul, S., Singthongchai, J., Naenudorn, E., et al.: Using of Jaccard coefficient for keywords similarity. In: Proceedings of the International MultiConference of Engineers and Computer Science, pp. 1–6 (2013)
17. Skoutas, D., Sacharidis, D., Simitsis, A., et al.: Ranking and clustering web services using multicriteria dominance relationships. IEEE Trans. Serv. Comput. **3**(3), 163–177 (2010)
18. Zhang, J., Xu, L., Li, Y.: Classifying Python code comments based on supervised learning. In: Meng, X., Li, R., Wang, K., Niu, B., Wang, X., Zhao, G. (eds.) WISA 2018. LNCS, vol. 11242, pp. 39–47. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02934-0_4