WT-LDA: User Tagging Augmented LDA for Web Service Clustering

Liang Chen[1], Yilun Wang[1], Qi Yu[2], and Jian Wu[1]

[1] Zhejiang University, China
[2] Rochester Institute of Technology, USA
{cliang, yilunwang, wujian2000}@zju.edu.cn,
qi.yu@rit.edu

Abstract. Clustering Web services that groups together services with similar functionalities helps improve both the accuracy and efficiency of the Web service search engines. An important limitation of existing Web service clustering approaches is that they solely focus on utilizing WSDL (Web Service Description Language) documents. There has been a recent trend of using user-contributed tagging data to improve the performance of service clustering. Nonetheless, these approaches fail to complete leverage the information carried by the tagging data and hence only trivially improve the clustering performance. In this paper, we propose a novel approach that seamlessly integrates tagging data and WSDL documents through augmented Latent Dirichlet Allocation (LDA). We also develop three strategies to pre-process tagging data before being integrated into the LDA framework for clustering. Comprehensive experiments based on real data and the implementation of a Web service search engine demonstrate the effectiveness of the proposed LDA-based service clustering approach.

1 Introduction

The explosive growth of Web services poses key challenges for Web service discovery. Existing service discovery approaches rely on either UDDI (*Universal Description Discovery and Integration*) based or Web service search engines to locate matching services. As many service providers choose to publish their Web services through their own websites instead of using public registries, the number of Web services in public UDDI registries decreases significantly. A recent study shows that more than 53% of the UDDI business registry registered services are invalid, whereas 92% of Web services cached by Web service search engines are valid and active [1]. Therefore, using search engines to search and discover Web services becomes more common and effective than UDDI service registries.

Existing Web service searching engines primarily focus on keyword-based matching on names, input/output parameters, and bindings defined in the Web service description file [1]. In this case, if a service description does not match the query term, it won't be discovered even though the service may provider the user desired functionality. As it is difficult for a casual user to choose keywords

that match the terms in a service description, keyword-based search usually suffers from low recall, where services containing synonyms or concepts at a higher (or lower) level of abstraction will not be discovered. As an example, a service named "Mobile Messaging Service" may not be returned for the query term "SMS" submitted by a user even though they describe the same concept. To handle this issue, service clustering has been recently exploited to improve the search quality [16–18]. By clustering Web services together, services in the same cluster are expected to provide similar functionalities so that they can be discovered together as a group. However, existing service clustering algorithms mainly rely on the WSDL descriptions of services, which usually contain very limited terms, some of which are even not proper words. Hence, these algorithm may lead to low clustering quality, which will negatively affect the accuracy of service discovery.

Recently, some real-world Web service search engines, such as *Seekda!*, have allowed users to manually annotate Web services using tags. Tags provide meaningful descriptions of objects and allow users to organize and index their contents. Tagging data has been proved to be very useful in many domains such as multimedia, information retrieval, data mining, and so on. Figure 1 shows two examples of Web service tags in *Seekda!* service search engine. *MeteorologyWS* in Fig. 1(a) is a weather forecasting Web service, which has two tags, *weather* and *waether*. However, there is no word weather in its service name or WSDL document. Thus, it is hard for this service to be clustered into the weather cluster. Further, this service will be hard to be retrieved without utilizing the tag information, if the query term is *weather*. Besides, the tag *waether* is also useful as some users may make a mistake in the typing process and use *waether* as the query term. As a service provider, one may have different naming convention and prefer to use *Meteorology* instead of *weather* in the generated WSDL file, as shown in Fig. 1(a). On the other hand, service users are likely to use the same tag to annotate services with similar functionality. Therefore, leverage the tagging information along with the WSDL can help improve the quality of service clustering.

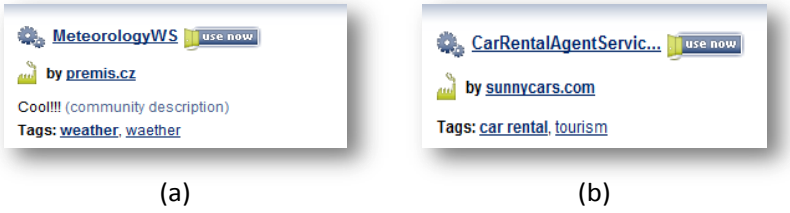

Fig. 1. Example of Web Service Tags

In this paper, we propose to augment the Latent Dirichlet Allocation (LDA) model, referred to as *WT-LDA*, to integrate WSDL documents and service tags

for service clustering. LDA has been demonstrated to be an effective tool in topic modeling and document clustering in the text domain. Specifically, WT-LDA models each service as a distribution of a set of topics and functionally similar services are expected to be represented by a similar set of topics. Service tags are also used to determine the topics of the services. We assess the effectiveness of the proposed *WT-LDA* via real-world Web services collected from Seekda!. Preliminary experimental results reveal that the performance of *WT-LDA* is affected by Web service with few tags or many meaningless tags. To tackle this issue, we propose three strategies to preprocess service tags before being used by *WT-LDA*. The experiment results in Section 5 demonstrate that our tag preprocessing strategies help improve the performance of *WT-LDA*. The major contributions of this paper can be summarized as follows:

1. We propose a novel Web service clustering approach *WT-LDA* based on probabilistic graphic model (LDA), in which both the WSDL documents and service tags are effectively utilized.
2. We propose three tag preprocessing strategies to improve the performance of *WT-LDA*.
3. We crawl 15,968 real Web services to evaluate the performance of *WT-LDA* and three tag preprocessing strategies.

The rest of this paper is organized as follows. Section 2 gives an overview of the related work on Web service discovery and clustering. Section 3 details the proposed *WT-LDA*. Section 4 presents the tag recommendation strategies that help improve the performance of *WT-LDA*. Section 5 shows the experimental results and Section 6 concludes the paper.

2 Related Work

With the wide adoption of service computing and cloud computing, Web service discovery becomes a popular research topic that attracts significant attention. Recently, Web service clustering [6, 10] has been demonstrated as an effective tool to boost the performance of Web service discovery. Most service clustering algorithms rely on the computation of similarity between services, which can be (1) semantic based and (2) non-semantic based. Ontology is utilized to compute the semantic similarity between Web services in many studies [2, 4, 11, 14]. Specifically, Cristina et al. [11] propose to use an ant-based method to cluster Web services based on semantic similarity. Sun et al. [14] propose to adopt Petri net as the modeling language for the specification of a service process model, and cluster services based on functional similarity and process similarity. In this paper, we focus on the clustering of non-semantic Web services as most services are described using the WSDL standard, which focuses on the syntactic description of services.

Several approaches have been developed for the calculation of non-semantic similarity between Web services [5, 8, 15]. Liu et al. propose to extract 4 features, i.e., *content, context, host name, and service name*, from the WSDL document

to cluster Web services [8, 15]. They take the process of clustering as the preprocessor to discovery, aiming to building a search engine that crawls and clusters non-semantic Web services. Khalid et al. also propose to extract features from WSDL documents to cluster Web services [5]. Different from the work in [8, 15], a set of different features, including *content, types, messages, ports, and service name* are extracted from the WSDL documents. Other clustering algorithms, such as Probabilistic Latent Semantic Analysis (PLSA), have also been applied to service discovery [9]. SVD based and matrix factorization based approaches are adopted to achieve the co-clustering of services and operations in [16, 18]. Co-clustering exploits the duality relationship between services and operations to achieve better clustering quality than one-side clustering.

Despite WSDL-based clustering is widely adopted, the clustering performance is rather limited as only WSDL documents are employed. With the development of Web service community, more and more tags are annotated to Web services by users. These tags can be employed to enhance the accuracy of service discovery. However, limited work has exploited tagging data for service discovery. In our preliminary work [3], we investigated the benefits of utilizing both WSDL documents and tagging data to cluster Web services. The findings motivate our present study. In this paper, we improve the performance of Web service clustering by introducing a novel LDA based approach to explore the knowledge behind WSDL & tags and by proposing three tag preprocessing strategies to improve the performance of service clustering.

3 WT-LDA based Service Clustering

In this section, we first describe the proposed architecture for Web service discovery framework in Section 3.1, and then introduce data preprocessing stage and the probabilistic graphic model of *WT-LDA* in Section 3.2 and Section 3.3, respectively.

3.1 Web Service Discovery Framework

Figure 2 shows the proposed architecture for Web service discovery framework, which consists of two major component: data preprocessing and service discovery. In the first component, both WSDL documents and tags of Web services are crawled from the Internet, which will be used for service clustering. Specifically, we use the meaningful words in WSDL document as the feature words to construct a probabilistic graphic model, i.e., *WT-LDA*. After we extract feature words and tags from Web services, the *WT-LDA* is used to cluster Web services. Since data preprocessing and service clustering are conducted offline, the efficiency of service discovery can be guaranteed. Hence, the focus will be placed on accuracy. In the second component, clustered result of *WT-LDA* will be used to improve the search result of a Web service search engine. When a query term is sent to the Web service search engine, it can return a more accurate search result by leveraging the clustered results.

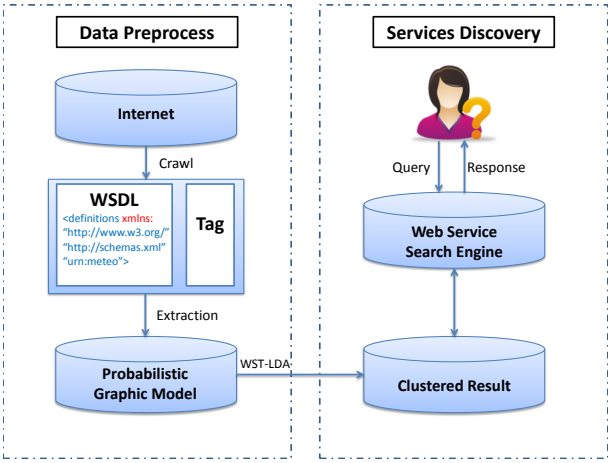

Fig. 2. Framework for Web service Discovery

3.2 Data Preprocessing

As discussed above, we extract the meaningful words from WSDL document as the feature words, and then jointly model these feature words and tags for the purpose of clustering of Web services. In this part, we describe the details in data preprocessing component.

1. **Building an original vector**. In this step, we perform tokenization over the entire WSDL document to produce the original content vector.
2. **Suffix Stripping**. Words with a common stem will usually have the same meaning, for example, *connect*, *connected*, *connecting*, *connection*, and *connections* all have the same stem *connect* [8]. We strip the suffix of all these words that have the same stem by using the Porter stemmer [12]. After the step of suffix stripping, a new content vector is produced.
3. **Pruning**. In this step, we remove two kinds of words from the content vector. The first kind is XML tags, such as *s:element*, *s:complexType*, and *wsdl:operation*, which are not meaningful for the comparison of content vectors. Content words are typically nouns, verbs or adjectives, and are often contrasted with function words which have little or no contribution to the meanings of texts. Therefore, the second kind of word to be removed is function word. Church *et al.* state that the function words could be distinguished from content words using a Poisson distribution to model word occurrence in documents [7]. Typically, a way to decide whether a word w in the content vector is a function word is by computing the degree of overestimation of the observed document frequency of the word w, denoted by n_w using Poisson distribution. The overestimation factor can be calculated as follows.

$$\Lambda_w = \frac{\hat{n_w}}{n_w}, \tag{1}$$

where $\hat{n_w}$ is the estimated document frequency of the word w. Specifically, the word with higher value of Λ_w has higher possibility to be a content word. In this paper, we set a threshold Λ_T for Λ_w, and take the words which have Λ_w higher than threshold as content words. The value of threshold Λ_T is set as follows:

$$\Lambda_T = \begin{cases} avg[\Lambda] & \text{if}(avg[\Lambda] > 1); \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where $avg[\Lambda]$ is the average value of the observed document frequency of all words considered. After the process of pruning, we can obtain a new content vector, in which both XML tags and function words are removed.

4. **Refining**. Words with very high occurrence frequencies are likely to be too general to discriminate between Web services. After the step of pruning, we implement a step of refining, in which words with too general meanings are removed. Clustering based approaches were adopted to handle this problem in some related work [5, 8]. In this paper, we choose a simple approach by computing the frequencies of words in all WSDL documents and setting a threshold to decide whether a word has to be removed.

After the above four steps, we obtain the meaningful words in a WSDL document.

3.3 WT-LDA

The WT-LDA model is based on Latent Dirichlet Allocation, which takes both the content of WSDL documents and the user-contributed tagging data into consideration. Main advantages of our proposed WT-LDA model are listed below:

1. It provides a generative probabilistic model of a WSDL corpus and a probabilistic view to extract latent variables from WSDL documents which can significant improve the clustering result of web services.
2. It measures the word co-occurrence from heterogeneous service descriptions of WSDL documents, infers the topic distribution of each WSDL document, and the result topic vectors can contribute to web services clustering.
3. It takes tagging data of a WSDL document into consideration while unique tag has its own distribution of topics. Tags with similar meaning or function have similar distribution of topics. So that the content of a WSDL document as well as tagging data can determine the clusters of web services.

In the model of WT-LDA, tag related to one document is chosen uniformly at random for each word in that document. Each tag has its own distinct contribution to the topic distribution of the documents. Thus, there is a topic distribution, which is drawn from Dirichlet hyper-parameter α corresponding to each tag of web service. The word distribution specific to each topic is drawn from the Dirichlet hyper-parameter β. Then, a topic is drawn from the topic distribution according to the chosen tag in the document, and the word is generated from that chosen topic. WT-LDA is a generative model of user-contributed tagging

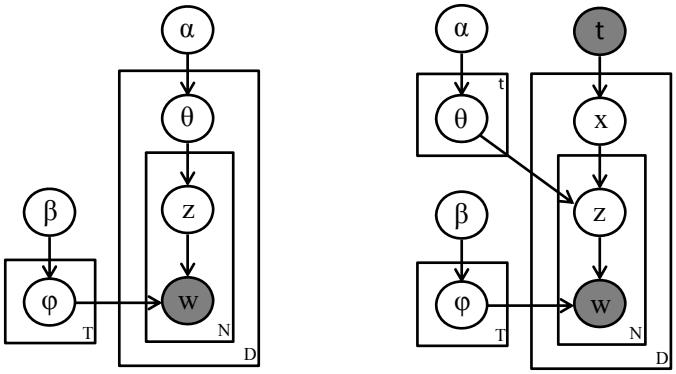

Fig. 3. Representation of LDA as a graphical model: the nodes denote random variables, while the edges indicate conditional dependencies. The shaded nodes are observed variables (words); the unshaded nodes are hidden variables (topics). The outer rectangles, or "plates", indicate repeated samples.

data and words in the WSDL documents. The generative process is described as follows:

1. Draw T multinomial ϕ_z from a Dirichlet prior β, one for each topic z
2. For each tag t_d in document d, draw a multinomial θ_{t_d} from a Dirichlet prior α; then for each word w_{di} in document d:
   (a) Draw a tag x_{di} uniformly from tags t_d in document d;
   (b) Draw a topic z_{di} from multinomial $\theta_{x_{di}}$;
   (c) Draw a word w_{di} from multinomial $\phi_{z_{di}}$;

The probabilistic graphical model corresponding to WT-LDA is shown in Fig. 3. As in the web service clustering model, each topic is associated with a distribution ϕ over words, drawn independently from a Dirichlet prior β. x indicates the tag responsible for a given word, chosen from t_d, and each tag has a distribution θ over topics generated from a Dirichlet prior α. The topic distribution of tags and the word distribution of topics work together to generate a topic z, and a word is drawn from the chosen topic.

As shown in the above process, the posterior distribution of topics depends on the information from both text and tag. WT-LDA parameterization is

$$\theta_{t_d}|\alpha \sim Dirichlet(\alpha) \tag{3}$$

$$\phi_z|\beta \sim Dirichlet(\beta) \tag{4}$$

$$z_{di}|\theta_d \sim Multinomial(\theta_d) \tag{5}$$

$$w_{di}|\phi_{z_{di}} \sim Multinomial(\phi_{z_{di}}) \tag{6}$$

$$x_{di}|t_d \sim Uniform(t_d) \tag{7}$$

We employ Gibbs sampling to perform approximate inference in *WT-LDA*. Note that Gibbs sampling provides a simple method for estimate the latent variable under Dirichlet priors and observed variable given by the content of the WSDL documents and the user-contributed tagging data of corresponding document. There are three latent variables in *WT-LDA* model-the word distribution of topic ϕ, the topic distribution of tag θ, the tag assignment of each word, and the topic assignment z of each word. Gibbs sampling construct a Markov chain that calculates the conditional distribution $P(z_{di} = j, x_{di} = k|w_i = m, z_{-di}, x_{-di}, t_d)$ where z_{-di} represents the topic assignments for all tokens except for w_{di}, x_{-di} represents the tag assigments for all tokens except for w_{di}, the conditional probability is shown below:

$$P(z_{di} = j, x_{di} = k|w_i = m, z_{-di}, x_{-di}, t_d) \propto \frac{m_{x_{di}z_{di}} + \alpha_{z_{di}}}{\sum_{v=1}^{V}(m_{x_{di}v} + \alpha_v)} * \frac{n_{z_{di}w_{di}} + \beta_{w_{di}}}{\sum_{v=1}^{V}(n_{z_{di}v} + \beta_v)} \tag{8}$$

where n_{zw} is the number of tokens of word w assigned to topic z, m_{xz} represent the number of tokens in tag x are assigned to topic z.

In Gibbs Sampling, we sample z_{di} and x_{di} by fixing z_{-di} and x_{-di}. The other two latent variables: the word distribution of topic ϕ and the topic distribution of tag θ are estimated from samples by:

$$\theta_{xz} = \frac{m_{x_{di}z_{di}} + \alpha_{z_{di}}}{\sum_{v=1}^{V}(m_{x_{di}v} + \alpha_v)} \tag{9}$$

$$\phi_{zw} = \frac{n_{z_{di}w_{di}} + \beta_{w_{di}}}{\sum_{v=1}^{V}(n_{z_{di}v} + \beta_v)} \tag{10}$$

For document d, we sum over all the θ_x where $x \in t_d$ to compute the topic distribution of document d. Therefore, we can cluster web services and get the detail information of each cluster by ϕ.

4 Tag Preprocessing Strategies

Some inherent properties of Web service tagging data, e.g., uneven tag distribution and noisy tags, impact the reliability of tagging data. In this section, we introduce three tag preprocessing strategies to make the tagging data more reliable and suitable for the WT-LDA framework for the purpose of improving service clustering accuracy.

4.1 Tag Recommendation

Through our observation, some Web services, especially those newly deployed ones, do not have tags. In this case, we have to assign some initial tags from the textual features of the services first by using approaches such as TF-IDF, and then use a tag recommendation approach to improve the quality of the tagging data. For the Web services with few tags, tag recommendation approaches could be directly employed. Typically, an initial set of tags $\mathcal{I}_s$ associated with a Web service s is provided to the recommendation method, which outputs a set of related tags $\mathcal{C}_s$, where $\mathcal{I}_s \bigcap \mathcal{C}_s = \emptyset$. Tag co-occurrence is a commonly used method for tag recommendation.

Fig. 4 shows an example of tag co-occurrence based recommendation framework, in which it first generates candidate tags based on original tags by using tag co-occurrence, and then obtains the recommended tags by using some tag ranking strategies, e..g, *Sum* and *Vote* [13]. Due to the space limitation, we do not give the details of tag recommendation. In this paper, tag co-occurrence and *Vote* ranking strategy are employed for recommendation.

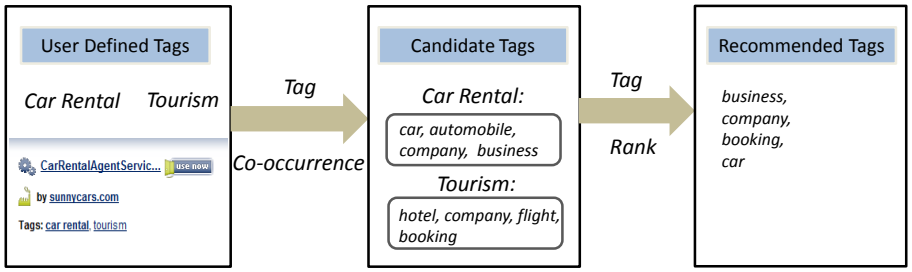

Fig. 4. An Example of Framework for Web Service Tag Recommendation

4.2 High-frequency Tags

In order to cluster Web services into the exact cluster they belong to, we expect tags of Web services provide accurate information of distinct property between different clusters and common property shared among the Web services in same cluster. As a result, we believe high-frequency tag, which means the unique tag of one Web service that occurs with a high frequency among all tags of the Web service, as an important evidence for web service clustering.

4.3 Tag Preprocessing Strategies

Based on the above analysis, we present three tag preprocessing strategies to improve the performance of *WT-LDA*

Table 1. Experimental Data Description

WSDL Document	185
Word	62,941
Token	1481
Tag	888
Recommended Tag	799

1. **Original tags**. In this strategy, only the original user-contributed tags are provided for *WT-LDA* based Web service clustering.
2. **Original tags + Recommended tags**. In this strategy, we mix original tags with recommended tags generated by the proposed approach in Section 4.1.
3. **High-frequency tag**. In this strategy, we select the high-frequency tag of each Web service for clustering.

5 Experiments

In this section, we first compare the performances of different Web service clustering approaches and then study the performances of tag preprocessing strategies .

5.1 Experiment Setup

To evaluate the performance of Web service clustering approaches and tag preprocessing strategies, we crawl 15,968 real Web services form the Web service search engine Seekda!. For each Web service, we get the data of service name, WSDL document, tags, and the name of service provider. We publicize the crawled dataset via http://www.zjujason.com.

As the manual creation of ground truth is an expensive process, we randomly select 185 Web services from the dataset we crawled to evaluate the performance of Web service clustering. We perform a manual classification of these 185 Web services to serve as the ground truth for the clustering approaches. Specifically, we distinguish the following categories: “Weather”, “Stock”, “SMS”, “Finance”, “Tourism”, and “University”. There are 28 Web services in ”Weather” category, 21 Web services in “Stock” category, 37 Web services in “SMS” category, 21 Web services in “Finance” category, 31 Web services in “Tourism” category, 27 Web services in “University” category. 20 Web services are randomly selected from other categories as noise in our experiment. Limited by space, we don’t show the detailed information of these Web services. The experimental data description is given in Table 1.

All experiments are implemented with JDK 1.7.0-10, Eclipse 3.6.0. They are conducted on a Dell OptiPlex 390 machine with an *3.10 GHZ Intel Core I3 CPU* and *2GB RAM*, running *Windows 7*.

5.2 Evaluation Metric

To evaluate the performance of Web service clustering, we introduce two metrics: Precision and Recall, which are widely adopted in the information retrieval community.

$$Precision_{c_i} = \frac{succ(c_i)}{succ(c_i) + mispl(c_i)} \tag{11}$$

$$Recall_{c_i} = \frac{succ(c_i)}{succ(c_i) + missed(c_i)}, \tag{12}$$

where $succ(c_i)$ is the number of services successfully placed into cluster c_i, $mispl(c_i)$ is the number of services that are incorrectly placed into cluster c_i, and $missed(c_i)$ is the number of services that should be placed into c_i but are placed into another cluster.

5.3 Performance of Web Service Clustering

In this section, we compare the performance of four Web service clustering approaches, including two state-of-art clustering approaches and two versions of the proposed *WT-LDA* approach. The details of these algorithms are given below:

1. ***WCluster***. In this approach, Web services are clustered according to the semantic WSDL-level similarity between Web services. This approach has been adopted in some related works [3, 5, 8].
2. ***WTCluster***. In this approach, both WSDL documents and the tagging data are employed to cluster the Web services according to the composite semantic similarity [3].
3. ***W-LDA***. In this approach, we extract feature words from WSDL documents and cluster Web service without any additional information using traditional LDA approach.
4. ***WT-LDA***. In this approach, we utilized both feature words from WSDL documents and the user-contributed tagging data, then cluster Web services using *WT-LDA* approach proposed in Section 3.

Fig. 5 shows the performance comparison of above 4 Web service cluster approaches. Empirically, we set $\alpha = 0.01$, $\beta = 0.01$ and run Gibbs-sampling for 1000 iterations in the proposed *WT-LDA* approach. It can be discovered that the proposed *WT-LDA* outperforms the other three approaches in most cases in terms of precision and recall, respectively. Further, it can be found the addition of tagging data improves the performance of service clustering, as *WT-LDA* outperforms *W-LDA*, and *WTCluster* outperforms *WCluster* in most cases.

Table 2 shows the average precision and recall values of the above four service clustering approaches. It can also be found that the proposed *WT-LDA* has the best performance in terms of average precision and average recall. Further, we can also find that *WTCluster* outperforms *WCluster*, and *WT-LDA* outperforms

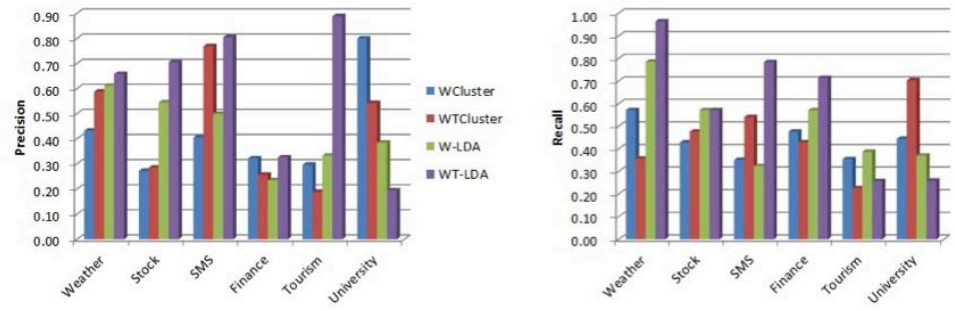

Fig. 5. Performance Comparison of Four Web Service Clustering Approaches

W-LDA. As we discussed above, the user-contributed tagging data of Web services contains a lot of information, such as service function, location, and other semantical information. Utilizing these information improves the performance of Web service clustering.

Table 2. Average Precision and Recall of Four Web Service Clustering Approaches

Clustering Approach	**Precision**	**Recall**
$WCluster$	0.4219	0.4378
$WTCluster$	0.4387	0.4553
$W-LDA$	0.4350	0.5017
$WT-LDA$	**0.5966**	**0.5919**

Fig. 6 shows the word distribution of two clusters, for each of which we pick words with top ten probability in each cluster and the size of word in Fig. 6 is corresponding to its probability. After observing the services in each cluster, we find the services in the first cluster are most about weather forecast, which matches the word distribution in the left figure. And the services in the second cluster are most about stock, which also matches the word distribution in the right figure. Thus, the proposed *WT-LDA* is quite effective. Compared with traditional unsupervised clustering approach, one advantage of *WT-LDA* is that user could directly know the main functionality of services in one cluster, instead of observing all services in the cluster.

5.4 Evaluation of Tag preprocessing Strategies

In this section, we compare the performance of the proposed three tag preprocessing strategies with the one without tagging data. The four clustering approaches for comparison are detailed as follows:

Fig. 6. Word Distribution of Two Clusters (Topics): the left cluster is about weather, while the right cluster is about stock

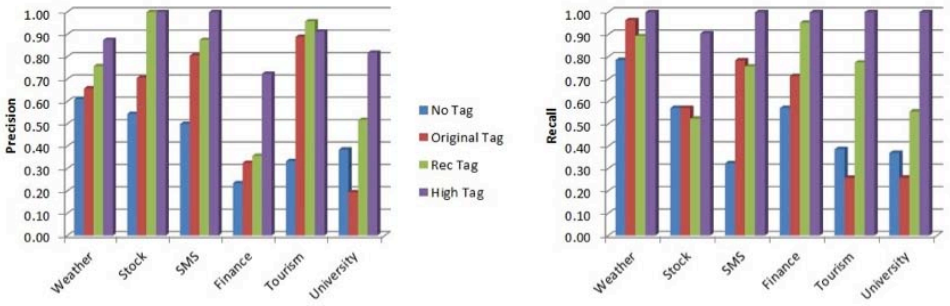

Fig. 7. Performance Comparison of Tag Preprocessing Strategies

1. ***No Tag***. In this approach, Web services are clustered according to word features from WSDL documents. We cluster Web services based on the traditional LDA approach.
2. ***Original Tag***. In this approach, we utilize both the WSDL documents and the original tags, and cluster the Web services using the proposed *WT-LDA*.
3. ***Rec Tag***. In this approach, we utilize the WSDL documents, the original tags, and the recommended tags. Then we cluster the Web services using the proposed *WT-LDA*.
4. ***High Tag***. In this approach, we utilize both the WSDL documents and the high-frequency tags, and cluster the Web services using *WT-LDA*.

Fig. 7 shows the performance comparison of the above four approaches. It can be found that the *High Tag* approach outperforms the other three ones in most cases in terms of precision and recall, respectively. This is because the high-frequency tags are helpful for distinguishing services, while the addition of low-frequency tags has a negative effect. From Fig. 7, it can also be found that the *Rec Tag* approach outperforms *Original Tag* approach in most cases, which means the recommended relevant tags improves the performance of service clustering.

Table 3. Average Precision and Recall of Four Tag Strategies

Clustering Approach	**Precision**	**Recall**
No Tag	0.4350	0.5017
Original Tag	0.5966	0.5919
Rec Tag	0.7442	0.7426
High Tag	**0.8882**	**0.9841**

Table 3 shows the average precision and recall values of the above four clustering approaches. It can be found that the *High Tag* has the best performance in terms of both average precision and average recall, while the performance of *No Tag* is the worst. This can be easily understood because *No Tag* does not utilize tagging data, which contains a lot of meaningful information. Similarly, it can be observed that *Rec Tag* outperforms *Original Tag*. Therefore, it can be found that the *High Tag* strategy is the best strategy for *WT-LDA* clustering.

6 Conclusion

In this paper, we propose a probabilistic graphical model based approach, referred to as *WT-LDA*, which explores the knowledge behind WSDL documents and user-contributed tagging data to cluster Web services. Three tag preprocessing strategies are also developed to improve the service clustering performance. Extensive experiments conducted over real Web services demonstrate the effectiveness of the proposed *WT-LDA* approach and tag preprocessing strategies. In our future work, we plan to use an online version of *WT-LDA* to improve the efficiency of Web service clustering, which allows the algorithm to scale to a massive number of services and service tags contributed by a large number of users.

7 Acknowledgements

This research was partially supported by the National Technology Support Program under the grant of 2011BAH16B04, the National Natural Science Foundation of China under the grant of No. 61173176, National High-Tech Research and Development Plan of China under the Grant No. 2012AA02A604 and No. 2013AA01A604.

References

1. Al-Masri, E., Mahmoud, Q.H.: Investigating web services on the world wide web. International World Wide Web Conference pp. 795–804 (2008)
2. Bianchini, D., Antonellis, V.D., Pernici, B., Plebani, P.: Ontology-based methodology for e-service discovery. ACM Journal of Information Systems 31(4), 361–380 (2006)

3. Chen, L., Hu, L., Zheng, Z., Wu, J.: Wtcluster: Utilizing tags for web services clustering. International Conference on Service Oriented Computing pp. 204–218 (2011)
4. Dasgupta, S., Bhat, S., Lee, Y.: Taxonomic clustering of web service for efficient discovery. Proceedings of International conference on Information and knowledge management pp. 1617–1620 (2010)
5. Elgazzar, K., Hassan, A.E., Martin, P.: Clustering wsdl documents to bootstrap the discovery of web services. International Conference on Web Services pp. 147–154 (2009)
6. Hao, Y., Junliang, C., Xiangwu, M., Bingyu, Q.: Dynamically traveling web service clustering based on spatial and temporal aspects. Lecture Notes in Computer Science 4802, 348–357 (2007)
7. K., C., W., G.: Inverse document frequency (idf): a measure of deviations from poisson. Proceedings of the ACL 3rd workshop on Very Large Corpora pp. 121–130 (1995)
8. Liu, W., Wong, W.: Web service clustering using text mining techniques. International Journal of Agent-Oriented Software Engineering 3(1), 6–26 (2009)
9. Ma, J., Zhang, Y., He, J.: Efficiently finding web services using a clustering semantic approach. In: CSSSIA '08: Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation. pp. 1–8. ACM, New York, NY, USA (2008)
10. Platzer, C., Rosenberg, F., Dustdar, S.: Web service clustering using multidimensional angles as proximity measures. ACM Transactions on Internet Technology 9(3), 1–26 (2009)
11. 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. Studies in Computational Intelligence 315, 23–33 (2010)
12. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
13. Sigurbjrnsson, B., van Zwol, R.: Flickr tag recommendation based on collective knowledge. Proceedings of the 17th international conference on World Wide Web pp. 327–336 (2008)
14. Sun, P., Jiang, C.: Using service clustering to facilitate process-oriented semantic web service discovery. Chinese Journal of Computers 31(8), 1340–1353 (2008)
15. WeiLiu, Wong, W.: Discovering homogeneous service communities through web service clustering. Service-Oriented Computing: Agents, Semantics, and Engineering pp. 69–82 (2008)
16. Yu, Q.: Place semantics into context: Service community discovery from the wsdl corpus. In: ICSOC. pp. 188–203 (2011)
17. Yu, Q.: Sparse functional representation for large-scale service clustering. In: ICSOC. pp. 468–483 (2012)
18. Yu, Q., Rege, M.: On service community learning: A co-clustering approach. In: ICWS. pp. 283–290 (2010)