Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

PICF-LDA: a topic enhanced LDA with probability incremental correction factor for Web API service clustering

Jiaji Shen, Wen Huang and Qiang Hu*

Abstract

Web API is a popular way to organize network services in cloud computing environment. However, it is a challenge to find an appropriate service for the requestor from massive Web API services. Service clustering can improve the efficiency of service discovery for its ability of reducing search space. Latent Dirichlet Allocation (LDA) is the most frequently used topic model in service clustering. To further improve the topic representation ability of LDA, we propose a new variant model of LDA with probability incremental correction factor (PICF-LDA) to generate the high-quality service representation vectors (SRVs) for Web API services. We first compute the words' topic contribution degree (TCD) in the service description text by its context weight and part-of-speech (POS) weight. Then the probability incremental correction factor (PICF) for a word is designed based on TCD and the word's maximum topic probability value. PICF is used to correct the probability distributions in SRVs. Experiments show that PICF-LDA has a better performance than LDA, the variant LDA models and other state-of-the-art topic models in service clustering.

Keywords: Web API, Cloud computing, Service clustering, LDA

Introduce

With the popularization of cloud computing technology, more and more enterprises have migrated their business systems to cloud service platforms [1]. Publishing cloud services is the main way for enterprises to encapsulate their business capability or products. Users can find suitable cloud services according to their needs [2]. As we all know, cloud computing can provide users with computing power, software and hardware resources. Therefore, it can greatly save the time and cost for the tenants to build new business systems [3].

Web API service is a common service publishing way in cloud computing environment [4]. Many enterprises have published a lot of web API services. For example, Google provides map service by multiple Web APIs, such as Static Maps API, Street View Image API, Distance Matrix API, Roads API, and Time Zone API. Given a road, we can embed its street view image with the speed limit into a web page by invoking Street View Image API and Roads API. Taking the website ProgrammableWeb as an example, we can find more than 25,000 web API services by the end of May 2022. It is convenient for us to build a new value-added application system by invoking these Web APIs. However, how to find an appropriate service for the users is becoming a challenge for the increasing number of Web API services published on the Internet [5].

Service clustering can group similar cloud services as a service cluster. It can reduce search space and improve the efficiency of service discovery [6]. Service clustering is widely used in service discovery. Early research on service clustering and discovery mainly focused on Web services described by WSDL. A WSDL document is a structured text with many tags. We can easily extract the feature information of Web services from the WSDL document to achieve service clustering [7].

*Correspondence: huqiang200280@163.com

College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, China

Currently, the function descriptions of Web API services are mostly organized by natural language [8]. The developers presented a short piece of text to describe the function, performance and usage of the Web API service. It is difficult to obtain the feature information about Web API services for the lack of tags in the service description texts [9]. To get the key feature of service description text, many researchers have applied topic models to generate topic vectors for Web API services [10–13]. These topic vectors are generated by service function description text. They are called service representation vectors (SRVs) in this paper. Service clustering for Web API services was carried out by computing the similarity of these SRVs.

LDA is easy to use and robust in topic modeling [14]. It is widely used in service clustering, text mining, sentiment analysis and other fields. In many clustering methods, LDA and its variant models are employed to generate SRVs. However, LDA doesn't consider the mutual position, semantic similarity and occurrence frequency of the words in the documents. In addition, some words with poor relevance to the topic will limit the quality of topic vectors generated by LDA [15]. Aiming to improve the express ability and semantic balance of topic modeling, we present a topic enhanced LDA with probability incremental correction factor for service clustering. The main contributions of this paper are as follows.

(1) Topic contribution degree (TCD), calculated by words' occurrence frequency, context similarity and part-of-speech, is proposed to evaluate the importance of a word in generating SRVs.
(2) We design a probability incremental correction factor (PICF) for each word based on TCD. The quality of SRVs generated based on LDA is improved by integrating PICF to correct the topic probability distribution.
(3) An extensive set of experiments are implemented to evaluate the performance of PICF-LDA. Experiment results show that the performance of PICF-LDA outperforms LDA, some variant LDA models and state-of-the-art topic models in service clustering.

The rest of this paper is organized as follows. Related work introduces the related works on service clustering and LDA model. How to compute topic contribution degree for the words in service description text is presented in Topic contribution degree. PICF-LDA elaborates the proposed PICF-LDA. Experiment verifies the effectiveness of PICF-LDA by service clustering experiments. Finally, conclusions and future work are presented in Conclusions.

Related work

Previous study on service clustering is mainly about the clustering methods of Web services which were described by Web service description language (WSDL). There are many tags in the WSDL documents, such as type, operation, input and output [16]. The feature information used in Web service clustering can be easily extracted from WSDL documents by these tags.

Kumara proposed a new approach to cluster Web services by mining WSDL documents and generating an ontology via hidden semantic patterns within the complex terms in service features to measure similarity [17]. Wu clustered Web services by utilizing both WSDL documents and tags to handle the clustering performance limitation caused by uneven tag distribution and noisy tags. He employed tag co-occurrence, tag mining, and semantic relevance measurement for tag recommendation [7]. Agarwal proposed an approach based on Length Feature Weight. It is used to vectorize the pre-processed WSDL files after preprocessing the WSDL documents. Experiments have proved that the proposed method outperforms the clustering done by using TF-IDF method for vector space representation of web services [18].

With the increasing number of cloud services described by natural language, it is difficult to obtain the service features from their description texts. So topic models are widely used in current service clustering. They are employed to extract topic features from the description texts of the cloud services. These topic features are used to perform service clustering. The topic models applied in service clustering mainly include LSA, LDA, BTM, HDP, and DMM [19]. Among the above models, LDA and its variants are the most widely used. Many researchers have managed to improve the performance of traditional LDA model. For example, Chen proposed WT-LDA which seamlessly integrates tagging data and WSDL documents through augmented LDA [20]. A semantic Web service discovery based on LDA clustering that learns the OWL-S Web service documents was presented by Zhao. It can make the documents more abundant of the semantic information [21]. Shi put forward WE-LDA which used word2vec to obtain word vectors and cluster words into word clusters by K-means++. These word clusters were incorporated to the semi-supervised training process of LDA [22].

Bukhari proposed Web service search engine (WSSE) by extracting topics from Web service descriptions based on LDA. WSSE is based on the probabilistic topic

modeling and clustering techniques that are integrated to support each other by discovering the semantic meaning of Web services and reducing the search space [23]. Zhao employed Word2Vec to adapt the representation of services, and learned a list of similar words in service corpus. Moreover, He integrated TF-IDF into the similarity calculation to filter noise words. This method can enhance LDA with the similar words finding and filtering strategy for service clustering [24]. As following work, Zhao proposed a model named as HRT-LDA. The effects of different tags on clustering performance are considered in HRT-LDA. A tag filtering and appending strategy based on transfer learning, Word2vec, TF-IDF and semantic computing is integrated into LDA. Experiments shows that HRT-LDA outperforms the state-of-art LDA in service clustering [25].

Web service structure was modeled as Weighted Directed Acyclic Graph (WDAG) by Baskara. Then Biterm Topic Model was employed to mine the topic on the WDAG for high precision service similarity calculation [26]. To improve topic modeling accuracy, an SP-BTM that only chooses the words with specific parts-of speech to form biterms was proposed by Hu and Liu [27]. After using the HDP model to solve service vectors' dimension problems, Cao adopted SOM neural network to cluster these service vectors [28]. Fletcher deployed the HDP technique to extract topics from service description and user requirements to enhance the discovery of services [29].

Some clustering methods in other fields also have enlightening significance for us to improve the quality of service clustering. For example, Li proposed an adaptive time interval clustering algorithm based on density grid. The algorithm can perform adaptive time-interval clustering according to the size of the real-time ship trajectory data stream, so that a ship's hot zone information can be found efficiently and in real-time [30]. Zhao presented an efficient framework to cluster previous summaries with new data. It significantly outperforms the existing incremental face clustering methods [31]. Xue developed a novel density-based clustering approach for incomplete data based on Bayesian theory, which conducts imputation and clustering concurrently and makes use of intermediate clustering results. Experimental results show the effectiveness of the proposed method [32].

A data-driven clustering recommendation method, called DDCR, is proposed to recommend hierarchical clustering or spectral clustering for scRNA-seq data. They perform DDCR on two typical single cell clustering methods, SC3 and RAFSIL, and the results show that DDCR recommends a more suitable downstream clustering method for different scRNA-seq datasets and obtains more robust and accurate results [33]. Hu presented a two-level weighting strategy to measure the importance of views and features. A collaborative working mechanism is introduced to balance the within-view clustering quality and the cross-view clustering consistency [34]. Xiong proposed a semantic clustering news keyword extraction algorithm based on TextRank. It uses the word vectors and k-means clustering to obtain semantic clustering. The proposed algorithm has greater precision, recall, and F1 value than the traditional TextRank and Term Frequency-Inverse Document Frequency (TF-IDF) algorithms [35].

Topic contribution degree

To make up for the deficiency of LDA in considering the words' mutual position, semantic similarity and occurrence frequency, we propose the concept of TCD to express the importance of a word in generating the SRVs. TCD calculates the words' weights in generating SRVs from three aspects: word context similarity, word frequency and part-of-speech. Relevant definitions about cloud service and TCD are presented as follows.

Definition 1. (Web API service) A Web API service is a 5-tuple $s=(Id, n, t, d, c)$, where

(1) *Id* is the ID number of the service;
(2) *n* is the name of the service;
(3) *t* is the set of service tags;
(4) *d* is the service description text;
(5) *c* is the category of the service;

Definition 2. (service representation vector) Given a Web API service *s*, if $srv=(k_1, k_2, \ldots, k_i, \ldots, k_n)$ is the topic vector generated by a topic model based on *s.d*, then *srv* is called the service representation vector of *s*.

Definition 3. (semantic similarity of words)w_i and w_j are two words in a piece of text T, V_{w_i} and V_{w_j} are the word vectors of w_i and w_j respectively, the semantic similarity of words w_i and w_j is defined as $SemSim(w_i, w_j) = \frac{V_{w_i} \bullet V_{w_j}}{|V_{w_i}| \times |V_{w_j}|}$.

Definition 4. (context fitness of word) *d* is *a* document including *m* different words. w_i is a word in *d*. The context fitness of word w_i in *d* is defined as $Context\_Fitness(w_i, d) = \sum_k SemSim(w_i, w_k)/(m-1)$. k=1,2, …, m and $k! = i$.

Definition 5. (TF-IDF of word) *d* is a document in the document set *D*. w_i is a word in *d*. The TF-IDF of word w_i in *d*.is defined as $TF\text{-}IDF(w_i, d, D) = TFw_i * IDFw_i$.

Here, $TFw_i = Nw_i/Nw$. Nw_i and Nw are the number of w_i in *d* and the total word number of *d*, respectively. $\mathrm{IDF}_{w_i} = lg\frac{|D|}{|\{j:w_i \in d_j\}|}$, $|D|$ is the number of documents in *D* and *j* is the number of documents including the word w_i.

Definition 6. (context weight of word) d is a document in the document set D. w_i is a word in d. The context weight of word w_i in d.is defined as $CW(w_i, d) = Context\_Fitness(w_i, d)^* TF\text{-}IDF(w_i, d, D)$.

Definition 7. (POS weight) d is a document in the document set D. For each word w in d, the POS weight, denoted as $PW(w, d)$, is a weight value assigned on w by its part-of-speech in d.

A word can better reflect the text topic if it appears frequently or has a high semantic similarity with other words. We know from the Definition 6 that the context weight of a word reflects the importance of a word in the text from the perspectives of word frequency and semantic similarity. It can be used to evaluate the importance of a word in generating SRVs.

In the service description, nouns are usually used to describe the function and operation objects of a cloud service. Verbs are mostly used to describe the operations or tasks contained in this service. Adjectives are commonly adopted to evaluate the quality of the cloud service. By introducing the part-of-speech of words, we further distinguish the importance of various words in the service description text when they are used to generate SRVs. The SRVs can be further optimized once the words in the service description are given different part-of-speech weights.

To comprehensively consider the influence of context weight and POS weight on generating SRVs, we present a concept of topic contribution degree (TCD).

Definition 8 (topic contribution degree) The topic contribution degree of a word w in document d is defined as $TCD(w, d) = CW(w, d)^* PW(w, d)$.

How to compute the TCD for words in the service description text is presented in Algorithm 1. Two empty set *Corpus_w* and *TCD_S* are firstly initialized in line (1). *Corpus_w* is used to store the service description texts for the cloud services in S. *TCD_S* is the set of TCD value for the words in service description texts. All the words in the services' description texts are added to *Corpus_w* by the codes from line (2) to line (4). Then, Word2vec is employed to train the vectors for each word in *Corpus_w*. These vectors will be used to calculate the semantic similarity between the words in context weight.

We compute TCD for each word form line (6) to line (12). The word's context weight is obtained by its context fitness and TF-IDF in line (8). The tool NLTK. pos_tag and stanfordcorenlp are used to determine the part-of-speech for every word in a service description text. The TCD value is computed based on the word's context weight and POS weight in line (9). It should be noted that the POS weight is set as a super parameter. We take the quality of SRVs as the optimization goal to evaluate POS weight for the words with different part-of-speech by adjusting parameters. The evaluated TCD value will be added to the set *TCD_ S*. The algorithm will finally return the set *TCD_S*.

Algorithm 1 TCD_calculation
Input: the set of Web API services S;
Output: the value of TCD for each word w in the service s of S;
(1) $Corpus\_w = TCD\_S = \emptyset$;
(2) ***for*** each $s \in S$
(3) $Corpus\_w = Corpus\_w \cup s.d$;
(4) ***endfor***
(5) train the vector $V(w)$ for each word w in $Corpus\_w$ by Word2vec;
(6) ***for*** each cloud service $s \in S$
(7) ***for*** each word w in $s.d$
(8) $CW(w, s.d) = Context\_Fitness(w, s.d)^* TF\text{-}IDF(w, s.d, Corpus\_w)$;
(9) $TCD(w, s.d) = CW(w, s.d)^* PW(w, s.d)$
(10) $TCD\_S = TCD\_S \cup TCD(w, s.d)$
(11) ***endfor***
(12) ***endfor***
(13) ***return***(TCD_S);

PICF-LDA

LDA is a topic model which can learn the hidden topic information of the existing documents and return topic vectors in the form of probability distribution. LDA is an efficient tool in topic modeling and text clustering domain. The graphical model of LDA is shown in Fig. 1. Here, n is the nth word in document d. α and η are topic parameter and proportions parameter; K is the number of topics; θ_d is per-document topic proportions; $Z_{d,n}$ represents doucument-word matrix while $W_{d,n}$ represents words' probability distribution.

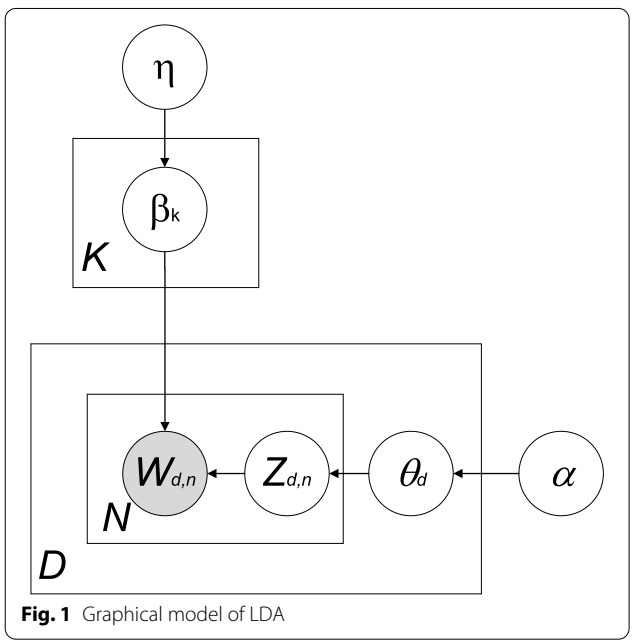

Fig. 1 Graphical model of LDA

LDA assumes that the prior distribution of document topics is a Dirichlet distribution. For any document d, its topic distribution is $\theta_d = Dirichlet(\vec{\alpha})$.

Here, $\vec{\alpha}$ is a hyper-parameter vector which have K dimensions. Then, the topic assignment of the n th word in document d can be calculated as $z_{dn} = multi(\theta_d)$. Similarly, LDA assumes that the prior distribution of words in the topic is Dirichlet distribution, that is, for any topic K, its word distribution is $\beta_k = Dirichlet(\vec{\eta})$.

Finally, the observed word probability distribution of w_{dn} is $w_{dn} = multi(\beta_{zdn})$. As shown in formula (1), the joint distribution of all the visible variables and the hidden variables in the LDA model can be approximated by Gibbs sampling.

$$p(w_d, z_d, \theta_d, \beta_k|\alpha, \beta) = \prod_{n=1}^{N} p(\theta_d|\alpha)p(z_{dn}|\theta_d)p(\beta_k|\beta)p(w_{dn}|\beta_k) \quad (1)$$

we propose a new variant model of LDA with probability incremental correction factor (PICF-LDA) to generate high-quality SRVs. The graphical model of PICF-LDA is shown in Fig. 2. An incremental correction factor is presented to correct the probability distribution value of SRVs. The incremental correction factor for word n in document d is represented as $PICF(d, n)$. The value of $PICF(d, n)$ is assigned as the product of $TCD(d, n)$ and $argmax(d, n)$. Here, $argmax(W_{d,n})$ refers to the maximum probability value of all the topics for n in the word-topic distribution matrix $W_{d,n}$.

PICF-LDA is an improved LDA model. It enhances the quality of topic vectors by utilizing PICF to correct the topic probability distribution. Algorithm 2 shows how to generate SRVs by PICF-LDA. An empty set *EQ_srv* is initialized in line (1). It is used to store the enhanced-quality SRVs for cloud service s in S. Then, the preprocessed web service description texts are sent to the LDA model to obtain $srv(s)$, $W_{d,n}$ and θ_d in line (3). Here $srv(s)$ is the SRV for cloud service s, $W_{d,n}$ is the word-topic matrix and θ_d is document probability distribution.

The correction of SRVs is organized in line (6) to line (12). Each word in the service description text needs to be processed as follows when PICF-LDA performs probability correction.

For the word n in $s.d$, we first find its the maximum topic probability distribution $argmax(W_{d,n})$ and its related topic (denoted as *kmax-topic*). Then the correction factor for the word n is presented as $PICF(d, n)$ in line (9). Here, we introduce λ to balance the order of magnitude for *PICF*. Finally, the value of $\theta_{k_{max\_topic},d}$ in $srv(s)$ will be updated by $\theta_{k_{max\_topic},d} * PICF(d, n)$. Algorithm 2 returns the enhanced-quality SRVs in line (13).

Algorithm 2 srv_PICF-LDA
Input: the set of Web service S;
Output: the SRVs for cloud services in S generated by PICF-LDA;
(1) $EQ\_srv = \emptyset$;
(2) ***for*** each cloud service s in S
(3) feed s.d to LDA for 15 iterations and obtain $srv(s)$, $W_{d,n}$ and θ_d;
(4) $EQ\_srv = EQ\_srv \cup \{srv(s)\}$
(5) ***endfor***
(6) ***for*** each $srv(s)$ in EQ_srv
(7) ***for*** each word n in $s.d$
(8) $k_{max\_topic} = argmax(W_{d,n})$;
(9) $PICF(d, n) = 1 + W_{d,n}[k_{max\_topic}] * TCD(d, n) * \lambda$;
(10) update $srv(s)$ by $\theta_{k_{max\_topic},d} = \theta_{k_{max\_topic},d} * \mathrm{PICF}(d, n)$;
(11) ***endfor***
(12) ***endfor***
(13) ***return***(*EQ_srv*);

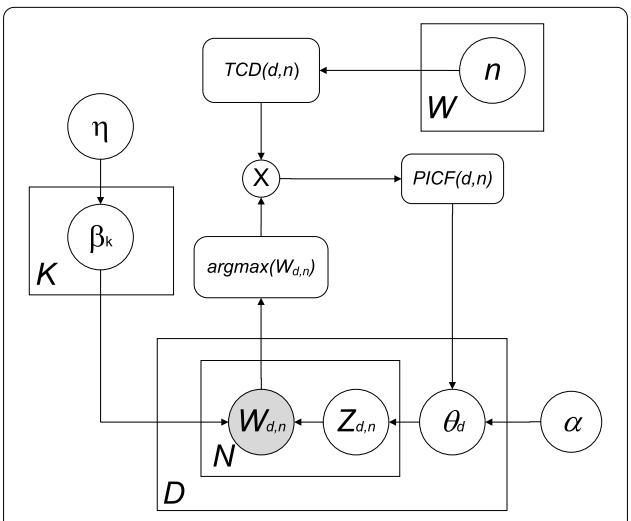

Fig. 2 Graphical model of PICF-LDA

Experiment

In this section, we carry out a series of clustering experiments to verify whether the quality of SRVs generated by PICF-LDA is better than LDA and other topic models. The pipeline process for cloud service or Web API clustering is shown in Fig. 3. All the experiments were conducted on a PC with Intel i7-8750 h and 16 GB RAM under Win10.

We first crawled the Web API services from cloud platforms, such as ProgrammableWeb and Casicloud. After preprocessing, the description information about these API services are transformed into service clustering texts. Then PICF-LDA is employed to generate SRVs based on the service clustering texts. Finally, K-means + + algorithm is used to cluster these Web API services based on the SRVs generated by PICF-LDA. The performance of the proposed PICF-LDA is compared with LDA, the variant LDA models and other state-of-the-art topic models from three evaluation metrics.

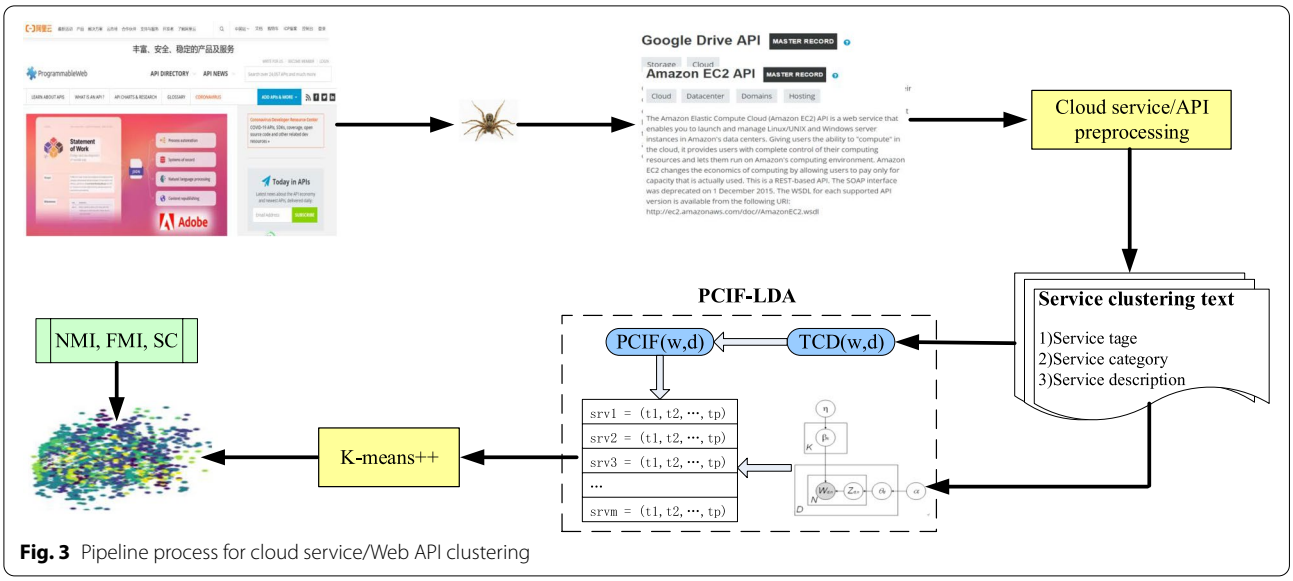

Fig. 3 Pipeline process for cloud service/Web API clustering

Tweet Archivist API (service name)

Search Social (service tag)

(service description)

Tweet Archivist is a service that lets users search Twitter for Tweets by sender, recipient, object of reference, or contents. Users may then create an archive based on that search which they can analyze, export, and share. Users may choose to keep the search private or share the results with friends, colleagues, or the world. Users may create a maximum of three active archives. The Tweet Archivist API enables users to visualize the data collected in their archives. Available visualizations include volume over time, top users, Tweet vs. Retweet, top words, top URLs, and source of Tweets.

Category Search (service category)

{ tweet archivist service let user search twitter tweet sender recipient object reference content user create archive based search analyze export share user choose keep search private share result friend colleague world user create maximum active archive tweet archivist api enables user visualize data collected archive available visualization include volume time top user tweet retweet top word top url source tweet }

Fig. 4 An example of Web API service

Dataset

An example of Web API service crawled from ProgrammableWeb is provided in the left part of Fig. 4. The main information that we can get about a Web API service includes: service name, service tag, service category and service description text. Following steps are used to process the service description texts.

(1) Text splitting: The words in the service description are separated by spaces.
(2) Removal of irrelevant characters and stopwords: In this step, firstly, irrelevant characters like punctuations marks, URLs, newline, special symbols, and quotes are removed from descriptions because they don't play any role for service clustering. After that, unnecessary words like 'a', 'an', 'the', 'what' etc. are removed.
(3) Lowercase conversion: All the words are converted into lowercase.
(4) Lemmatization: Lemmatization is used to convert a word with tense or voice changes into its root form.
(5) Addition of service tag and category: Words in service tag and category are added into the service description text.

After the above five steps, we have generated new description texts that can be used for service clustering. The text on the left side of Fig. 4 is the service description text processed by the above steps for "Tweet Archivist API".

We crawled 22,832 Web API services from ProgrammableWeb. To ensure the quality of services participating in service clustering, the Web API service has been removed once its service description is less than 15 words. Meanwhile, categories with less than 20 services were also eliminated. The final data set in our experiment contains 19,241 Web API services. They are preprocessed

Table 1 Outline of datasets

DataSet	Service components	Number of services
DS1	Top 20	9394
DS2	Top 50	14,768
DS3	Top 80	17,516
DS4	Top 132	19,421

to generate the new service description texts for service clustering experiments. We rank service categories according to the number of Web services they include. The top 20, 50, 80 and 132 Web API service categories are chosen as the classification benchmark category. They are named DS1, DS2, DS3and DS4, respectively. We will carry out the experiments on these data sets with different granularity. Table1 presents the outline of datasets.

Evaluation Metrics

Let $X = \{x_1, x_2, ..., x_k\}$ and $Y = \{y_1, y_2, ..., y_k\}$ be the predicted clustering labels and the real category labels, respectively. The following evaluation metrics are employed to observe the performance of different topic models.

Normalized Mutual Information (NMI)

NMI is used to evaluate the degree of consistency between two samples. It is the normalization of mutual information (MI) score. The calculation method is shown in formula (2).

$$MNI(X,Y) = \frac{MI(X,Y)}{F(H(X),H(Y))} \tag{2}$$

$MI(X,Y)$ is the mutual information of X and Y. It reflects the correlation degree between X and Y. $H(x)$ and $H(y)$ represent the entropies of X and Y respectively. F is the normalized function. The range of NMI value is [0,1]. The higher score means the better clustering quality in view of NMI.

Fowlkes-Mallows scores(FMI)

FMI is defined as the geometric mean of paired precision and recall rate. The calculation method is shown in formula (3).

$$FMI(X,Y) = \frac{TP(X,Y)}{\sqrt{(TP(X,Y)+FP(X,Y))(TP(X,Y)+FN(X,Y))}} \tag{3}$$

where, TP is the number of true positive (the number of positive samples predicted as positive class), FP is the number of false positive (the number of negative samples predicted as positive class) and FN is the number of false negative (the number of positive samples predicted as negative class).

The range of FMI value is [-1,1]. The higher score means the better clustering quality in view of FMI.

Silhouette Coefficient (SC)

For a sample x, let a be the average distance from other samples in the same category, and b be the average

Table 2 Influence of POS

PW_n	PW_a	PW_v	SC	NMI	FMI	CS	PW_n	PW_a	PW_v	SC	NMI	FMI	CS
0.1	0.1	0.8	0.5504	0.2962	0.2401	1.0867	0.3	0.4	0.3	0.5425	0.2959	0.2400	1.0784
0.1	0.2	0.7	0.5497	0.2966	0.2405	1.0868	0.3	0.5	0.2	0.5399	0.2947	0.2402	1.0748
0.1	0.3	0.6	0.5501	0.2961	0.2402	1.0864	0.3	0.6	0.1	0.5455	0.3030	0.2410	1.0895
0.1	**0.4**	**0.5**	0.5633	0.3060	0.2412	1.1105	0.4	0.1	0.5	0.5358	0.2946	0.2399	1.0703
0.1	0.5	0.4	0.5491	0.2954	0.2403	1.0848	0.4	0.2	0.4	0.5359	0.2974	0.2402	1.0735
0.1	0.6	0.3	0.5544	0.3054	0.2418	1.1016	0.4	0.3	0.3	0.5382	0.3019	0.2411	1.0812
0.1	0.7	0.2	0.5469	0.2944	0.2404	1.0817	0.4	0.4	0.2	0.5376	0.2967	0.2400	1.0743
0.1	0.8	0.1	0.5509	0.2973	0.2404	1.0886	0.4	0.5	0.1	0.5395	0.3027	0.2412	1.0834
0.2	0.1	0.7	0.5436	0.2943	0.2402	1.0781	0.5	0.1	0.4	0.5298	0.2972	0.2402	1.0672
0.2	0.2	0.6	0.5436	0.2953	0.2403	1.0792	0.5	0.2	0.3	0.5285	0.2948	0.2400	1.0633
0.2	0.3	0.5	0.5505	0.3053	0.2417	1.0975	0.5	0.3	0.2	0.5300	0.2973	0.2401	1.0674
0.2	0.4	0.4	0.5438	0.2947	0.2403	1.0788	0.5	0.4	0.1	0.5267	0.2939	0.2402	1.0608
0.2	0.5	0.3	0.5511	0.3036	0.2412	1.0959	0.6	0.1	0.3	0.5233	0.2957	0.2401	1.0591
0.2	0.6	0.2	0.5404	0.2865	0.2388	1.0657	0.6	0.2	0.2	0.5238	0.2957	0.2399	1.0594
0.2	0.7	0.1	0.5517	0.3049	0.2415	1.0981	0.6	0.3	0.1	0.5243	0.2968	0.2402	1.0613
0.3	0.1	0.6	0.5390	0.2936	0.2401	1.0727	0.7	0.1	0.2	0.5168	0.2958	0.2400	1.0526
0.3	0.2	0.5	0.5414	0.2944	0.2397	1.0755	0.7	0.2	0.1	0.5146	0.2955	0.2402	1.0503
0.3	0.3	0.4	0.5420	0.2954	0.2399	1.0773	0.8	0.1	0.1	0.5109	0.2967	0.2400	1.0476

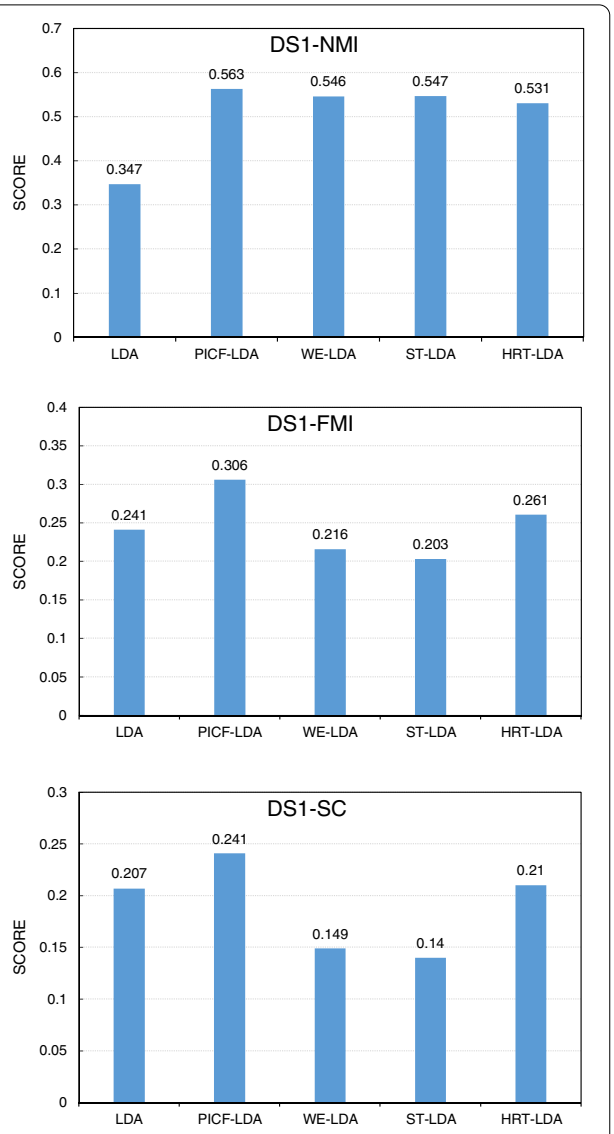

Fig. 5 Performance comparison between PICF-LDA and variant LDA models in DS1

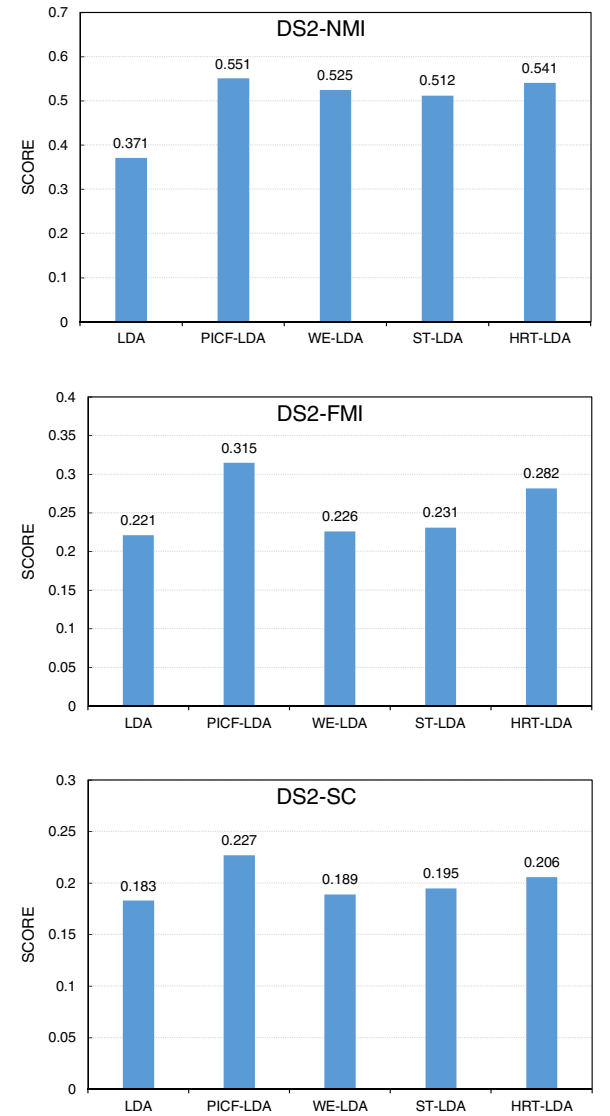

Fig. 6 Performance comparison between PICF-LDA and variant LDA models in DS2

distance from the nearest samples in different categories. The Silhouette Coefficient of x is given by formula (4).

$$SC(x) = \frac{b - a}{\max(a, b)} \tag{4}$$

The range of SC value is [-1,1]. The higher score means the better clustering quality in view of SC.

Baseline models

To verify the performance of PICF-LDA, we have chosen the LDA and the following three variant LDA models as baseline models to verify the performance of our method on DS1 to DS4. Meanwhile, the topic models HDP, BTM, DMM and LSA were employed to perform service clustering. The service clustering quality was also evaluated between our method and these state-of-the-art topic models.

(1) LDA-K: This method uses the traditional LDA and K-means++to perform service clustering.
(2) WE-LDA [21]: In WE-LDA, the word vectors obtained by Word2vec are clustered into word clusters by K-means++algorithm and these word

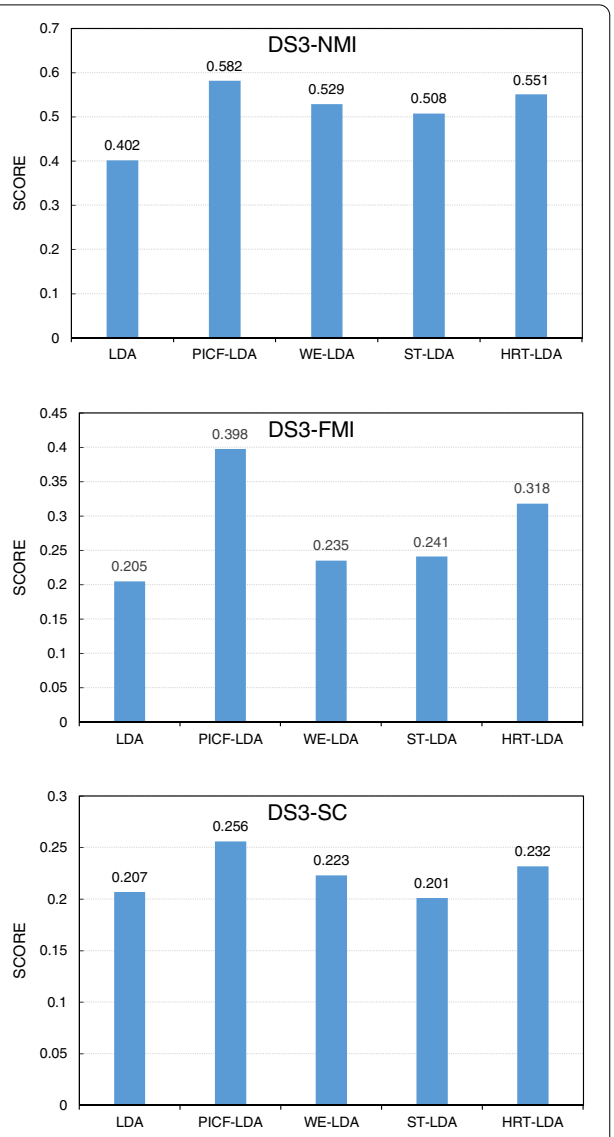

Fig. 7 Performance comparison between PICF-LDA and variant LDA models in DS3

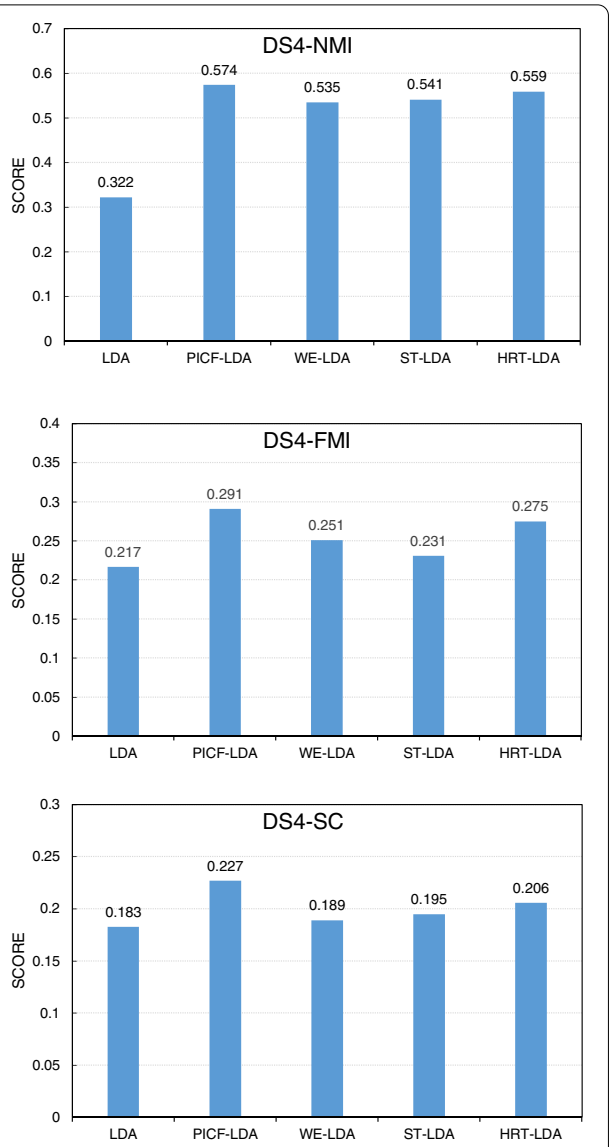

Fig. 8 Performance comparison between PICF-LDA and variant LDA models in DS4

Table 3 Performance improvement of PCIF-LDA and variant LDA models

Model	NMI	FMI	SC
LDA	38.2%	37.0%	22.1%
WE-LDA	6.3%	23.8%	17.3%
ST-LDA	7.7%	28.2%	17.7%
HRT-LDA	4.0%	14.3%	8.4%

clusters are incorporated to semi-supervise the LDA training process, which can elicit better distributed representations of Web services.

(3) ST-LDA [23]: In ST-LDA, Word2vec is adopted to adapt the representation of services, and learn a list of similar words in service corpus. TF-IDF is integrated into similarity calculation to filter noise words for LDA.
(4) HRT-LDA [24]: In HRT-LDA, the effects of different tags on clustering performance are considered. A tag filtering and appending strategy based on transfer learning, Word2vec, TF-IDF and semantic computing is integrated into LDA.

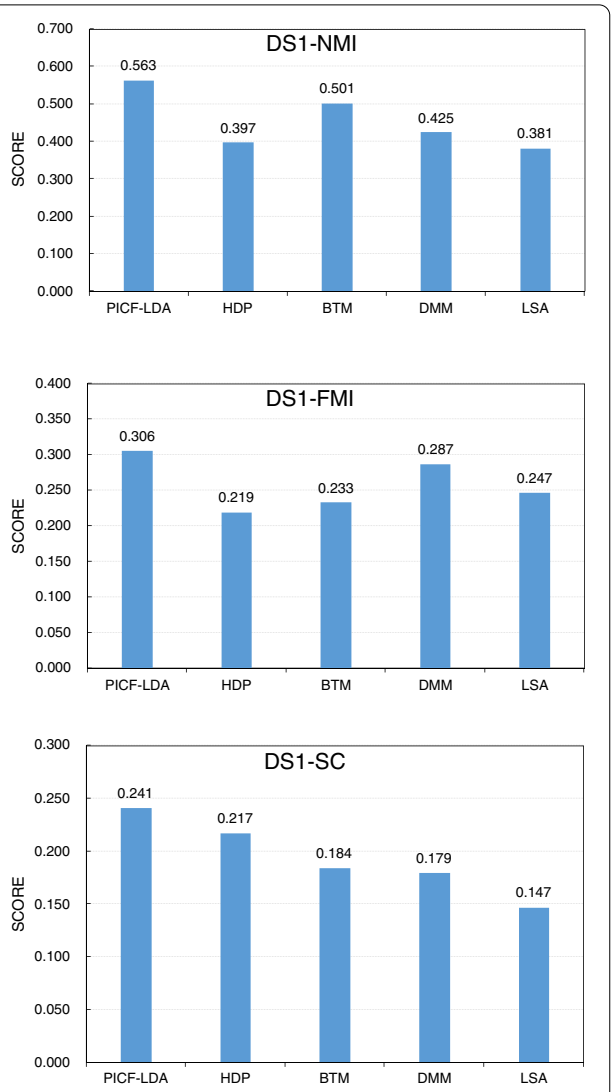

Fig. 9 Performance comparison between PICF-LDA and non-LDA models in DS1

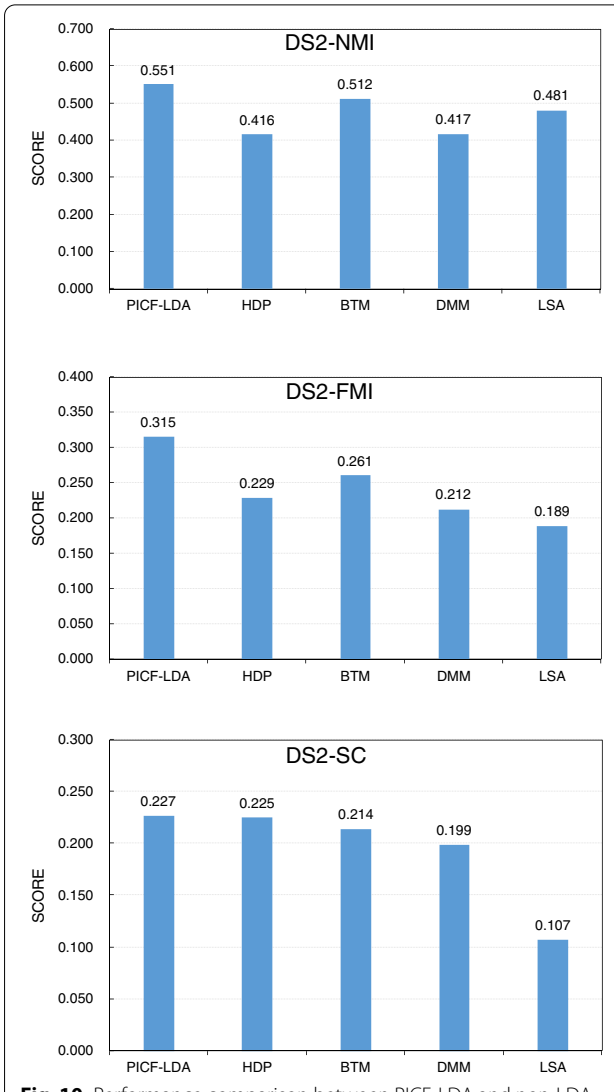

Fig. 10 Performance comparison between PICF-LDA and non-LDA models in DS2

Result and comparison

Compared with the traditional LDA model, the probability increment correction factor is added into PCIF-LDA. PCIF consists of three parts: context weight, TF-IDF and POS weight. The words' context weights and TF-IDFs can be calculated by algorithm 1. POS weight is a super parameter, which needs to be set by parameter adjustment.

We take DS1 as an example to set POS weights in this section. All the words in service description texts are divided into nouns, verbs, adjectives and adverbs according to their part-of-speeches. The adjectives and adverbs were given the same POS weight in this experiment. We use *PWn*, *PWv* and *PWa* to denote the POS weights for the nouns, verbs, adjectives and adverbs, respectively. The evaluation metrics SC, NMI and FIM were investigated during the adjustment of POS weights.

To comprehensively find the optimal POS weight in view of CS, NMI and FMI, we use CS (comprehensive score) to sum the value of three evaluation metrics. Table 2 provides the scores of various evaluation metrics with different POS weight in our experiment. We can see that highest quality of SRVs appears in the POS weight (0.1, 0.4, 0.5). That is the PICF-LDA shows the best performance when the weights of nouns, adjectives and verbs are set as 0.1, 0.4 and 0.5 in DS1.

After determining the POS weights for each dataset, we verify the quality of SRVs by service clustering experiments on the dataset DS1 to DS4. The comparison

Table 4 Performance improvement of PCIF-LDA and non-LDA models

Model	NMI	FMI	SC
HDP	21.9%	24.0%	5.6%
BTM	17.8%	23.6%	11.9%
DMM	32.8%	23.0%	10.3%
LSA	31.1%	33.1%	39.2%

models are LDA, WE-LDA, ST-LDA and HRT-LDA. K-means++algorithm is employed to perform service clustering.

We compare the performance between PICF-LDA and variant LDA models. The scores of three evaluation metrics for different datasets were presented

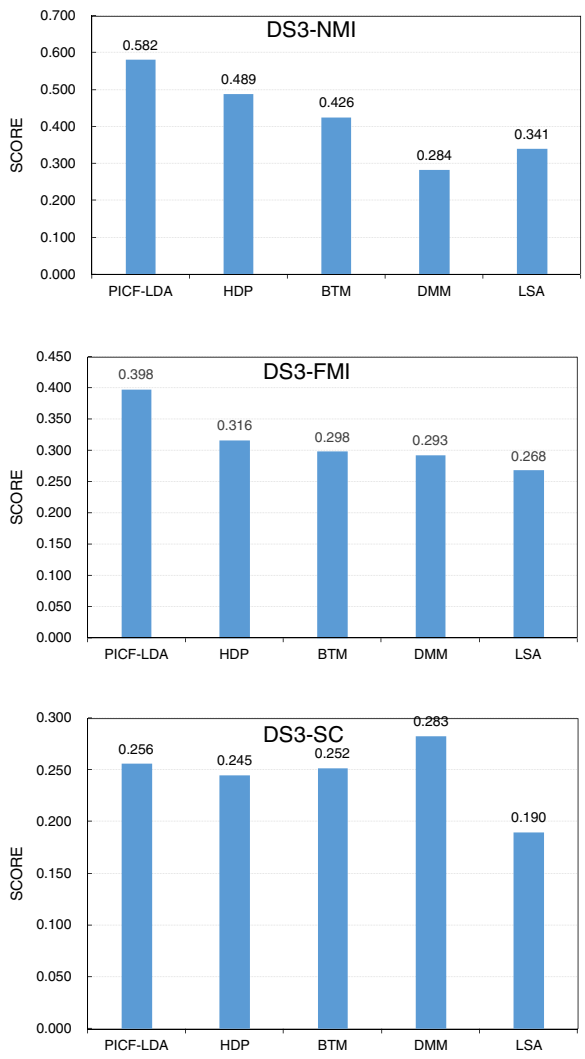

Fig. 11 Performance comparison between PICF-LDA and non-LDA models in DS3

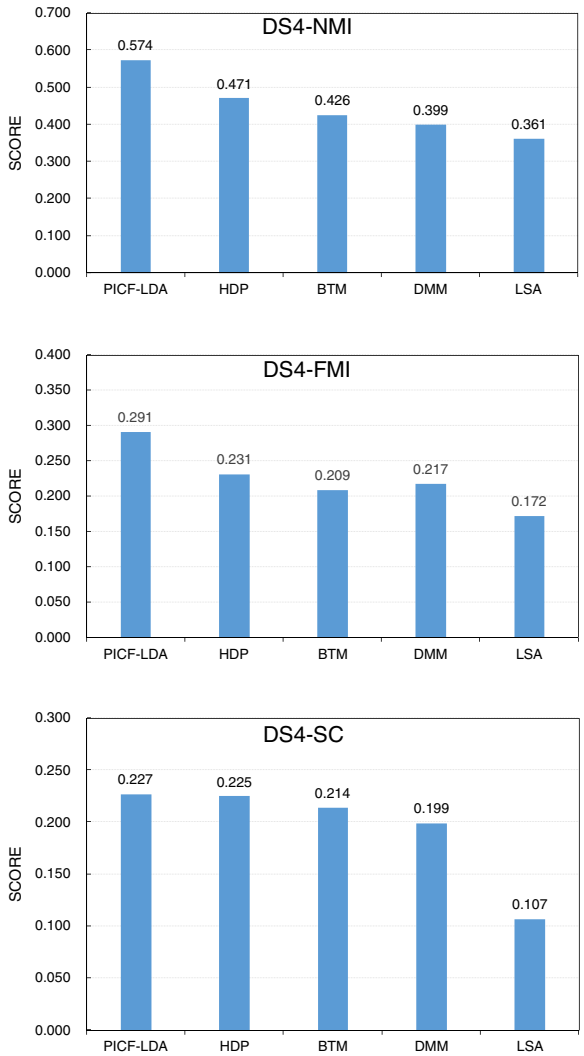

Fig. 12 Performance comparison between PICF-LDA and non-LDA models in DS4

in Figs. 5, 6, 7 and 8. We can see that PCIF-LDA has achieved better performance than other models in every evaluation metric and dataset. It proves that the introduction of PCIF has improved the quality of SRVs. Compared with the given models, the performance improvement data for each evaluation metric of PCIF-LDA was shown in Tab. 3. PCIF-LDA improves the topic extraction performance of the traditional LDA model by more than 22%. Compared with the other three state-of-art variant models, PCIF-LDA has also enhanced the scores of valuation metrics by 4%-28.2%. Therefore, PCIF-LDA is effective and advanced to extract the topic information for service clustering.

Besides LDA, HDP, BTM, DMM and LSA are also the commonly used topic models for service clustering. We

compared the performance between PICF-LDA and these state-of-the-art topic models. The score of NMI, FMI and SC for different datasets were presented in Figs. 9, 10, 11 and 12. We can see that PCIF-LDA has achieved better performance than these topic models in every evaluation metric and dataset. Table 4 shows the promotion proportion of different evaluation metrics in four data sets. The scores of NMI, FMI and SC has been increased by 17.8% to 32.8%, 23.6% to 33.1%, and 5.6% to 39.2%, respectively.

Conclusions

Although there are many new topic models, LDA is still widely used in service clustering for its ease of use and robustness. The researchers have also proposed a series of improved models for LDA. These variant models perform well in topic extraction. To further improve the performance of LDA in service clustering, we proposed an improved LDA model which is named as PICF-LDA. TCD is first presented to help determine the contribution of words in topic extraction. Then PICF is designed to correct the probability distribution of SRVs. PICF-LDA can generate high-quality SRVs for the cloud services and improve the quality of service clustering. We verify that the quality of SRVs generated by PICF-LDA is better than LDA and its variant models by experiments. Meanwhile, PICF-LDA also has a better the topic extraction performance than state-of-the-art topic models in service clustering.

In future work, we will investigate how to improve the performance of PICF-LDA from the perspective of feature word extraction. We will also apply the PCIF to other topic models to verify the universal effectiveness of the proposed method.

Abbreviations

LDA: Latent Dirichlet Allocation; POS: Part-of-Speech; TF: Term Frequency; IDF: Inverse Document Frequency; SRV: Service Representation Vector; TCD: Topic Contribution Degree; PICF: Probability Incremental Correction Factor.

Acknowledgements

The authors will be grateful to the editor and anonymous referees for their valuable comments and suggestions.

Authors' contributions

Jiaji Shen conducted the evaluation experiments and wrote the manuscript. Wen Huang collected the data and designed the program. Qiang Hu designed this study and reviewed the manuscript. The author(s) read and approved the final manuscript.

Funding

This work is supported the Natural science foundation of China under Grant 61973180, the Natural science foundation of Shandong Province under Grant ZR2019MF033 and ZR2021MF092, the key research program of Shandong Province (Soft Sciences) under Grant 2021RKY02037, and the foundation of Yun'nan Educational Committee under Grant 2022J0635.

Availability of data and materials

The data used to support the findings of this study are available from the corresponding author upon request.

Declarations

Competing interests

The authors declare no conflict of interest.

Received: 3 August 2021 Accepted: 20 June 2022
Published online: 18 July 2022

References

1. Rashid A, Chaturvedi A (2019) Cloud computing characteristics and services: a brief review. Int J Comput Sci Eng 7(2):421–426
2. Movahedi Z, Defude B (2021) An efficient population-based multi-objective task scheduling approach in fog computing systems. J Cloud Comput 10:53
3. Silva J, Marques ER, Lopes L, Silva F (2021) Energy-aware adaptive offloading of soft real-time jobs in mobile edge clouds. J Cloud Comput 10:51
4. Qi L, He Q, Chen F, Zhang X, Dou W, Ni Q (2020) Data-driven web APIs recommendation for building web applications. IEEE Trans Big Data. https://doi.org/10.1109/TBDATA.2020.2975587(earlyaccess)
5. Tsagkaropoulos A, Verginadis Y, Papageorgiou N, Paraskevopoulos F, Apostolou D, Mentzas G (2021) Severity: a QoS-aware approach to cloud application elasticity. J Cloud Comput 10:38
6. Cao B, Liu XF, Rahman MM, Li B, Liu J, Tang M (2017) Integrated content and network-based service clustering and web apis recommendation for mashup development. IEEE Trans Serv Comput 13(1):99–113
7. Wu J, Chen L, Zheng Z, Lyu MR, Wu Z (2014) Clustering web services to facilitate service discovery. Knowl Inf Syst 38(1):207–229
8. Zhang N, Wang J, He K, Li Z, Huang Y (2019) Mining and clustering service goals for RESTful service discovery. Knowl Inf Syst 58(3):669–700
9. Agarwal N, Sikka G, Awasthi LK (2020) Web service clustering approaches to enhance service discovery: a review. The International Conference on Recent Innovations in Computing. Springer, Singapore, pp 23–35
10. Cao B, Liu J, Wen Y, Li H, Xiao Q, Chen J (2019) QoS-aware service recommendation based on relational topic model and factorization machines for IoT Mashup applications. J Parallel Distrib Comput 132:177–189
11. Yang D, He D (2021) Web service clustering method based on word vector and biterm topic model. 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA). IEEE, Surabaya, pp 299–304
12. Jiang Y, Tao D, Liu Y, Sun J, Ling H (2019) Cloud service recommendation based on unstructured textual information. Futur Gener Comput Syst 97:387–396
13. Agarwal N, Sikka G, Awasthi LK (2020) Evaluation of web service clustering using Dirichlet Multinomial Mixture model based approach for dimensionality reduction in service representation. Inf Process Manage 57(4):102238
14. Cao B, Liu XF, Liu J, Tang M (2017) Domain-aware Mashup service clustering based on LDA topic model from multiple data sources. Inf Softw Technol 90:40–54
15. Jelodar H, Wang Y, Yuan C, Feng X, Jiang X, Li Y, Zhao L (2019) Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. Multimed Tools Appl 78(11):15169–15211
16. Liang T, Chen L, Ying H, Wu J (2014) Co-clustering WSDL documents to bootstrap service discovery. 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications. pp 215–222
17. Kumara BTGS, Paik I, Chen W (2013) Web-service clustering with a hybrid of ontology learning and information-retrieval-based term similarity. 2013 IEEE 20th International Conference on Web Services. pp 340–347. https://doi.org/10.1109/ICWS.2013.53
18. Agarwal N, Sikka G, Awasthi LK (2020) Enhancing web service clustering using length feature weight method for service description document vector space representation. Expert Syst Appl 161(2020):113682
19. Hu Q, Shen J, Wang K, et al (2022) A Web service clustering method based on topic enhanced Gibbs sampling algorithm for the Dirichlet Multinomial Mixture model and service collaboration graph[J]. Inf Sci 586:239-260.

20. Chen L, Wang Y, Yu Q, Zheng Z, Wu J (2013) WT-LDA: user tagging augmented LDA for web service clustering. International conference on service-oriented computing. Springer, Berlin, Heidelberg, pp 162–176
21. Zhao H, Chen J, Xu L (2019) Semantic web service discovery based on LDA clustering. International Conference on Web Information Systems and Applications. Springer, Cham, pp 239–250
22. Shi M, Liu J, Zhou D, Tang M, Cao B (2017) WE-LDA: a word embeddings augmented LDA model for web services clustering. 2017 ieee international conference on web services. pp 9–16
23. Bukhari A, Liu X (2018) A Web service search engine for large-scale web service discovery based on the probabilistic topic modeling and clustering. SOCA 12(2):169–182
24. Zhao Y, He K, Qiao Y (2018) ST-LDA: high quality similar words augmented LDA for service clustering. International conference on algorithms` and architectures for parallel processing. Springer, Cham, pp 46–59
25. Zhao Y, Qiao Y, He K (2019) A novel tagging augmented LDA model for clustering. Int J Web Serv Res 16(3):59–77
26. Baskara AR, Sarno R (2017) Web service discovery using combined biterm topic model and WDAG similarity. 2017 11th International Conference on Information & Communication Technology and System (ICTS). IEEE, Chengdu, pp 235–240
27. Hu R, Liu J, Wen Y (2020) SP-BTM: A Specific Part-of-speech BTM for Service Clustering. 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking. IEEE, Exeter, pp 1050–1057
28. Cao BQ, Xiao QX, Zhang XP, Liu JX (2019) An API service recommendation method via combining self-organization map-based functionality clustering and deep factorization machine-based quality prediction. Chin J Comput 6(42):1367–1383
29. Fletcher KK (2018) A quality-based web api selection for mashup development using affinity propagation. International conference on services computing. Springer, Cham, pp 153–165
30. Li J, Jiao H, Wang J, Liu Z, Wu J (2020) Online real-time trajectory analysis based on adaptive time interval clustering algorithm. Big Data Mining Analytics 3(2):131–142
31. Zhao X, Wang Z, Gao L, Li Y, Wang S (2021) Incremental face clustering with optimal summary learning via graph convolutional network. Tsinghua Sci Technol 26(4):536–547
32. Xue Z, Wang H (2021) Effective density-based clustering algorithms for incomplete data. Big Data Mining Analytics 4(3):183–194
33. Tian Y, Zheng R, Liang Z, Li S, Wu FX, Li M (2021) A data-driven clustering recommendation method for single-cell RNA-sequencing data. Tsinghua Sci Technol 26(5):772–789
34. Hu J, Pan Y, Li T, Yang Y (2020) TW-Co-MFC: Two-level weighted collaborative fuzzy clustering based on maximum entropy for multi-view data. Tsinghua Sci Technol 26(2):185–198
35. Xiong A, Liu D, Tian H, Liu Z, Yu P, Kadoch M (2021) News keyword extraction algorithm based on semantic clustering and word graph model. Tsinghua Sci Technol 26(6):886–893

Publisher's Note