IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 26, 2019, accepted October 18, 2019, date of publication November 6, 2019,
date of current version December 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2950355

Exploiting User Tagging for Web Service Co-Clustering

TINGTING LIANG[1], YISHAN CHEN[2], WEI GAO[2], MING CHEN[3,4], MEILIAN ZHENG[4,5], AND JIAN WU[2]

[1]School of Computer Science and Technology, Hangzhou Dianzi University, Hanzhou 310018, China
[2]College of Computer Science and Technology, Zhejiang University, Hanzhou 310027, China
[3]Hithink RoyalFlush Information Network Company, Ltd., Hangzhou 310023, China
[4]Zhejiang Hithink RoyalFlush Artificial Intelligence Research Institute, Hangzhou 310023, China
[5]School of Management, Zhejiang University of Technology, Hangzhou 310023, China

Corresponding author: Meilian Zheng (zmldlk@zjut.edu.cn)

This work was supported in part by the Subject of the Major Commissioned Project— Research on China's Image in the Big Data of Zhejiang Province's Social Science Planning Advantage Discipline —Evaluation and Research on the Present Situation of China's Image under Grant 16YSXK01ZD-2YB, in part by the Ministry of Education of China under Grant 2017PT18, in part by the Zhejiang University Education Foundation under Grant K18-511120-004, Grant K17-511120-017, and Grant K17-518051-021, in part by the Major Scientific Project of Zhejiang Laboratory under Grant 2018DG0ZX01, in part by the National Natural Science Foundation of China under Grant 61672453, and in part by the Key Laboratory of Medical Neurobiology of Zhejiang Province.

ABSTRACT We propose a novel Web services clustering framework by considering the word distribution of WSDL documents and tags. Typically, tags are annotated to Web services by users for organization. In this paper, four strategies are proposed to integrate the tagging data and WSDL documents in the process of service clustering. Tagging data is inherently uncontrolled, ambiguous, and overly personalized. Two tag recommendation approaches are proposed to improve the tagging data quality and service clustering performance. Comprehensive experiments demonstrate the effectiveness of the proposed framework using a real-world dataset.

INDEX TERMS Web service, WSDL documents clustering, co-clustering, tag recommendation.

I. INTRODUCTION

A service-oriented computing (SOC) paradigm and its realization through standardized Web service technologies provide a promising solution for the seamless integration of single-function applications to create new large-grained and value-added services. SOC has attracted industry attention and is applied in many domains, e.g., workflow management, finance, e-Business, and e-Science. With a growing number of Web services, discovering the user-required Web services is becoming an imperative task.

Generally, Web service discovery could be achieved by two main approaches: UDDI (*Universal Description Discovery and Integration*) and Web service search engines. As mentioned in [1], Web service discovery mechanisms are developed from the agent match-making paradigm and expanded to UDDI registry. Currently, the availability of Web services in UDDI decreases rapidly, since many Web service providers decided to publish services through their own websites instead of using public registries. The statistics presented in [2] shows that the percentage of invalid registered services in UDDI business registries is about 53%, while the percentage of valid and active Web services caught by service search engines is up to 92%. Compared with UDDI, using search engine to discover Web services is more effective.

Web service search is typically limited to keyword matching on names, locations, businesses, and buildings defined in the Web service description file [3]. A service would not be retrieved if the query does not contain at least one exact word in service name or description. For example, a service named "*Travel Service*" may not be returned from the query term "*Tourism*" provided by a user, though these two terms mean the same topic. As a consequence, clustering services into groups with similar functions based on WSDL documents became a widely accepted method to alleviate the drawbacks of traditional keyword-based service search engines [4], [5]. Previous work considered the similarity measurement between Web services by focusing on the feature extraction and similartity function definition. Work in [4] proposed techniques to automatically cluster WSDL documents into

The associate editor coordinating the review of this manuscript and approving it for publication was Ying Li.

functionally similar groups. Meanwhile, some related works proposed to extract five features from WSDL documents to cluster Web services [5].

Many existing methods for Web service clustering are limited by the one-way clustering, which solely clusters WSDL documents but ignores the distribution of words extracted from WSDL documents. The previous clustering approaches mainly considered the semantic similarities based on WSDL documents through the contents or titles. It should be noted that different service developers may make quite different descriptions to the same function in Web services, which would lead to the deviation of semantic similarity calculation. To overcome these drawbacks, we proposed an approach named WC*Cluster* to *co-cluster* or *simultaneously* cluster both documents and words, and modeled service clustering as a graph partitioning problem in our previous work [6]. The limitation of traditional Web service clustering approach could be largely relaxed by employing the co-clustering technique. However, the accuracy of these algorithms is still unsatisfactory, due to the limitation of service description data used by clustering.

Some real-world Web service search engines (i.e., Seekda![1] and Titan[2]) allow users to manually annotate Web services with tags to improve the organization of the stored content. Tags usually contain meaningful descriptions for Web services, and provide valuable information for service clustering. Figure 1 shows the tag number distribution of 1813 Web services crawled from Titan. The y-axis presents the number of tags assigned to the corresponding Web service. The largest number reaches 31, and for most Web services, about 77.3%, there are 1-4 labeled tags. Figure 1 shows that the scalability of tagging data for Web services is quite high.

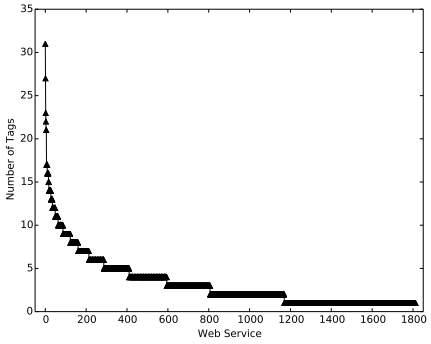


FIGURE 1. Distribution of the number of tags per web service in Titan.

It should be noted that the set of tags annotated to a Web service is an explicit set of keywords that users have found appropriate for categorizing the target Web service. Figure 2 shows the distribution of Web service tags over the most common *WordNet*[3] categories. Following this approach, we can

[1]Seekda: http://webservices.seekda.com/
[2]Titan: http://ccnt.zju.edu.cn:8080/
[3]WordNet: http://wordnet.princeton.edu/

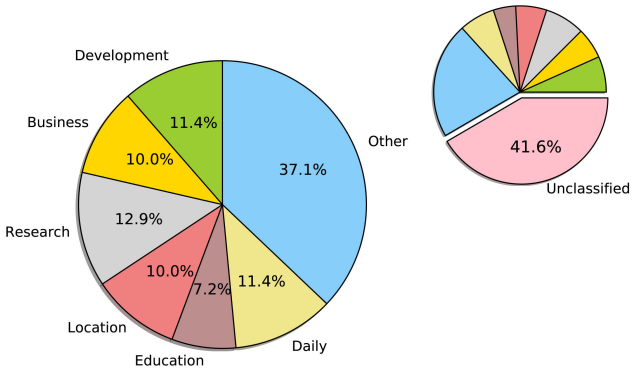


FIGURE 2. Most frequent WordNet categories for web service tags.

classify 58.4% of the tags in the collection, while leaving 41.6% of the tags unclassified, as depicted in the smaller pie chart of Fig.2. When focusing on the set of classified tags, it can be observed that *Research* are tagged most frequently (12.9%), followed by *Development* (11.4%), *Daily*(11.4%), *Location*(10%), *Business*(10%), and *Education*(7.2%). The category *Other* (37.1%) contains the set of tags that is classified by the WordNet broad categories, but does not belong to any of the aforementioned categories. Figure 2 demonstrates that users not only tag the function of the Web services (e.g., Business, Education, etc.), but also to a large extent provide a broader context (e.g., Location).

We use two Web services labeled *Tourism*, **LocationWS** and **xmlPostHotel**, as a simple example to illustrate the superiority of integrating tag information with service clustering. The terms with high frequency of occurrence extracted from the two WSDL documents respectively are ''location'', ''document'' and ''post'', ''hotel'', which are hardly relevant to each other. However both the services are tagged with a same term ''tourism'', and the strategies of combining tag information and WSDL document would assist the two services to be clustered in the same group.

We propose a novel clustering framework, named WTO, to integrate WSDL documents and service Tags for the CO-clustering approach. Based on the previously proposed clustering approach named WC*Cluster* [6], we view the problem of Web service clustering as a special text clustering problem and achieve an improvement by simultaneously clustering Web services and the words extracted from them. Inspired by the co-clustering algorithm [7], we consider the co-clustering problem as finding minimum cut vertex partitions in the bipartite graph between services and words. To further improve the performance of WC*Cluster*, we propose four integration strategies to leverage tags along with the words extracted from WSDL documents. Furthermore, two tag recommendation approaches are proposed to improve the tagging data quality and service clustering performance, by considering the inherently uncontrolled, ambiguous, and overly personalized tagging data.

In particular, the main contribution of our paper is summarized as follows:

1) We propose a novel framework, named WTO, which employs WC*Cluster* and explores WSDL documents & user tagging data for Web service clustering.
2) Four strategies are proposed to integrate WSDL documents and user tagging data in the process of service clustering.
3) Two tag recommendation approaches are proposed and applied to improve the performance of WTO.
4) We evaluate the performance of the proposed WTO by employing a real-world dataset crawled from Titan Web service search engine.

The remainder of the paper is structured as follows. Section II reviews the related work. Some preprocessing techniques, details of WC*Cluster* and four data integration strategies are introduced in Section III. Section IV discusses the process of tag recommendation, while Section V describes the experimental evaluation. Finally, Section VI concludes this paper.

II. RELATED WORK

The rapid development of service computing makes the service-oriented tasks draw much attention [8]–[10]. Web service discovery [11]–[16] is one of the most classical and imperative among these tasks. The techniques used for discovering semantic Web services and non-semantic Web services are quite different, as the former are described by OWL-S or WSMO while the latter use WSDL as description language [3], [17]–[19]. Study about the semantic Web service discovery is relatively mature. Benatallah *et al.* [20] presented a matchmaking algorithm which takes a service request and an ontology of service as input, then the algorithm finds a set of services whose descriptions contain as much common information with the ontology as possible. In [17], Klusch *et al.* proposed high-level match-making techniques and employed them for semantic Web services as structure annotations are available for service profiles.

We focus on non-semantic Web service discovery. In [21], Dong *et al.* first proposed a basic set of search functionalities that should be supported an effective Web service search engine, and then presented algorithms for supporting similarity search, which combine multiple sources of evidence to determine similarity between a pair of web-service operations. Nayak [3] described an approach based on the idea of finding search sessions which are similar to the one by user to locate a specific service and suggest words used in those sessions for the improvement of Web service discovery.

Many efforts have been devoted to cluster Web services to facilitate service discovery [4], [22]–[25]. Platzer *et al.* [24] proposed a statistical clustering approach that can improve an existing distributed vector space search engine for Web services based on the possibility of dynamically calculating clusters of similar services. In [22], Skoutas *et al.* proposed a methodology to rank the relevant services for a given request, and investigated methods for clustering the relevant services in a way that reveals and reflects the different trade-offs between the matched parameters. Liu and Wong [4] proposed techniques to gather, discover and integrate features related to a set of WSDL documents and cluster them into functionally similar homogeneous service communities. Analogously, Elgazzar *et al.* [5] presented an approach to improve service discovery of non-semantic Web services by clustering services into functionally similar groups through mining WSDL documents. In [23], Xia *et al.* developed a service clustering method named *vKmeans* which identifies core services of each category and clusters other services by the functional similarity.

In our previous work [6], the technique employed for WSDL documents clustering considers simultaneously clustering WSDL documents and the words extracted from them. There exists some early research about co-clustering. In [26], Cheng and Church presented an approach that allows automatic discovery of similarity based on a subset of attributes, simultaneous clusters both genes and conditions for knowledge discovery from expression data. Dhillon proposed co-clustering for the first time in [7], which regards the simultaneous clustering as a bipartite graph partitioning problem. The paper also showed that the solution of a real relaxation to the NP-complete graph bipartitioning problem are singular vectors of an appropriately scaled word-document matrix. Co-clustering algorithm has been widely applied in various fields. George and Merugu [27] designed incremental and parallel versions of co-clustering algorithm that involves simultaneous clustering of users and items, and apply it in building an efficient real-time CF framework. In [28], Liu and Shah utilized Maximization of Mutual Information co-clustering approach to discover clusters of intermediate concepts.

The performance of Web service clustering is limited by the singleness of WSDL documents, even though the co-clustering technique is useful. Recently tag information is becoming an essential component in many online systems including some Web service search engines. In [29], Ramage *et al.* proposed extensions of k-means and LDA using tagging data and demonstrated that tag information improves the performance of clustering algorithms. Cravino *et al.* [30] defined a distance metric based on a weighted cosine similarity that combines the textual features and the community structure of a network of tags to enhance the clustering of documents. In the field of Web service discovery, utilizing tag information to improve the performance gradually becomes feasible [31]–[34]. Chen *et al.* [35] proposed to utilize both WSDL documents and tags for web service clustering and adopted two tag recommendation strategies to improve the proposed approach. In [31], Chen *et al.* presented a hybrid mechanism using WSDL documents and tag network information to compute tag relevance scores by semantic computation and HITS model. Further, the mechanism was introduced into three applications of Web service mining. Wu *et al.* [32] presented a hybrid web service tag recommendation strategy to handle the clustering performance limitation. In [33], Lo *et al.* proposed a Web service tag learning system which

automatically learns high-quality tag annotations for service discovery.

We propose to utilize both WSDL documents and tag information to improve the accuracy of a co-clustering approach WC*Cluster*. Specifically, four strategies are proposed to integrate WSDL documents and tagging data in the process of co-clustering. Moreover, we employ a tag recommendation process to relax the limitation caused by the uncontrolled quality of tagging data.

III. TAGGED WEB SERVICES CO-CLUSTERING

We introduce the proposed WTO framework and the detailed co-clustering algorithm in Section III-A and Section III-B, respectively. The four data integration strategies for WC*Cluster* are introduced in Section III-C

A. WTO FRAMEWORK

We propose to extract terms that can represent the semantics or functionality of the Web services, and then use the co-clustering technique to simultaneously group the tagged Web services. Figure 3 illustrates the framework of WTO, which could be divided into four parts. From top to bottom, the first part is words and tags preprocessing including the stop words filtration, stemming, and TF-IDF (one kind of words weighting technique) calculation. The second part is a crucial component in WTO which introduces four integration strategies for word and tag combination. And the third part is the application of co-clustering algorithm based on the term-by-service matrix generated from the integration strategy. The last part is tag recommendation process which could be treated as an improvement for the data integration in the second part. This section introduces the first three parts of WTO framework, while the last part will be discussed in Section IV.

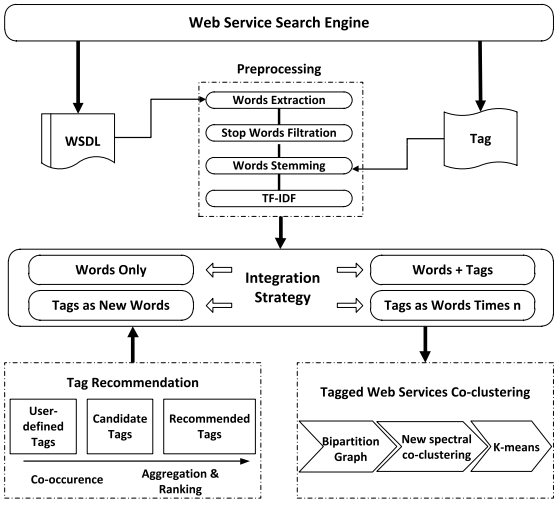


FIGURE 3. Framework of WTO.

B. WCCLUSTER

We first show the general result of data preprocessing which would be described in experiment part detailedly, then introduce the detailed process of WC*Cluster* approach. Figure 4 shows an example of preprocessing including stop words filtration and porter stemmer algorithm. The original words are finally turned into "request", "service", "security", "bind", "job", "title", as "Get" is filtered as a stop word, "Titles" and "Title" are transformed into "title" by porter stemmer algorithm.

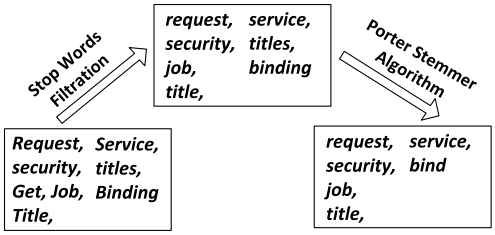


FIGURE 4. Example of preprocessing.

1) BIPARTITE GRAPH MODEL

Before we describe the major algorithm, we intend to introduce the bipartite graph model to represent a Web service collection. An undirected bipartite graph can be denoted by a triple $G = (\mathcal{S}, \mathcal{T}, E)$ where $\mathcal{S} = \{s_1, ..., s_n\}$ and $\mathcal{T} = \{t_1, ..., t_m\}$ represent two sets of vertices and E represents a set of edges $\{\{t_i, s_j\} : t_i \in \mathcal{T}, s_j \in \mathcal{S}\}$. In the scenario of Web service clustering, $\mathcal{S}$ is the set of Web services while $\mathcal{T}$ is the set of terms extracted from WSDL documents and tags of services. Figure 5 shows that if t_j is extracted from Web service s_i, the edge $\{t_i, s_j\}$ exists. Note that there are no edges between terms or between services.

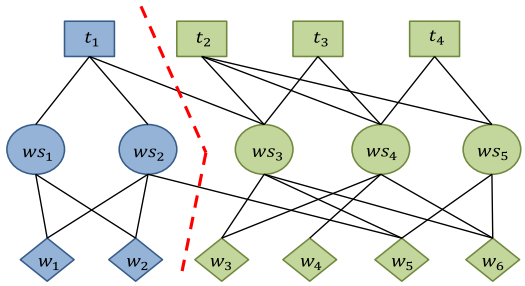


FIGURE 5. Two bipartite graphs of web services, words and tags.

Each edge represents a relation between a Web service and a term. In order to capture the strength of the relation, we set a positive weight for each edge. For instance, the term frequency could be used as the weight of the corresponding edge. The adjacency matrix $\boldsymbol{R}$ of the bipartite graph can be defined as follows,

$$R_{ij} = \begin{cases} E_{ij}, & \text{if there is an edge } \{t_i, s_j\} \\ 0, & \text{otherwise} \end{cases},$$

where E_{ij} denotes the weight of edge $\{t_i, s_j\}$. We can use the form of block matrix to represent matrix $\boldsymbol{R}$ as follows,

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{0} \end{bmatrix},$$

where $\boldsymbol{B}$ is $m \times n$ term-service matrix and B_{ij} denotes the edge weight E_{ij}. Here the vertices have been ordered so that the first m vertices indicate terms and the next n indicate services.

Given the vertex set mixed by $\mathcal{S}$ and $\mathcal{T}$, we can partition it into two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$. The cut between the two subsets plays an important role in WC*Cluster*, which is defined as

$$cut\left(\mathcal{V}_1, \mathcal{V}_2\right) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} R_{ij}, \tag{1}$$

which could be easily extended to k vertex subsets,

$$cut\left(\mathcal{V}_1, ...\mathcal{V}_k\right) = \sum_{i<j} cut\left(\mathcal{V}_i, \mathcal{V}_j\right).$$

A premise of WC*Cluster* algorithm is the observation of duality: *term clustering and Web service clustering can induce each other.* It could be easily verified that the best clustering of terms and services would correspond to a partitioning of graph such that the crossing edges between subsets have minimum weight.

$$cut\left(\mathcal{S}_1 \cup \mathcal{T}_1, ..., \mathcal{S}_k \cup \mathcal{T}_k\right) = \min_{\mathcal{V}_1, ..., \mathcal{V}_k} cut\left(\mathcal{V}_1, ..., \mathcal{V}_k\right),$$

where $\mathcal{V}_1, ..., \mathcal{V}_k$ is a k-partition of the bipartite graph.

2) SPECTRAL GRAPH BIPARTITIONING

There exist several effective heuristic methods for solving graph partition problem. One of them is spectral graph partitioning. Next we intend to specifically introduce the spectral graph partitioning in the WC*Cluster* approach. Given the undirected graph $G = (\mathcal{S}, \mathcal{T}, E)$ with n vertices in $\mathcal{S}$ and m vertices in $\mathcal{T}$, the *Laplacian matrix* $\boldsymbol{L} = \boldsymbol{L_G}$ of G can be defined as follows:

$$L_{ij} = \begin{cases} \sum_k E_{ik}, & i = j \\ -E_{ij}, & i \neq j \text{ and there is an edge } \{i, j\} \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where $\boldsymbol{L}$ is an $(m+n) \times (m+n)$ symmetric matrix with one row and column for each vertex.

As mentioned above, the optimal partition would be obtained by minimizing the cut value. In addition to small cut values reflecting the association between different partitions, the "balance" of clusters should be considered when solve a partition problem. Now we can provide a new objective function,

$$\mathcal{F}\left(\mathcal{V}_1, \mathcal{V}_2\right) = \frac{cut\left(\mathcal{V}_1, \mathcal{V}_2\right)}{w\left(\mathcal{V}_1\right)} + \frac{cut\left(\mathcal{V}_1, \mathcal{V}_2\right)}{w\left(\mathcal{V}_2\right)}, \tag{3}$$

where $w\left(\mathcal{V}_k\right)$ is the weight of a vertex subset $\mathcal{V}_k$, and $w\left(\mathcal{V}_k\right) = \sum_{i \in \mathcal{V}_k} w\left(i\right) = \sum_{i \in \mathcal{V}_k} W_{ii}$. W_{ii} is a diagonal element of the diagonal matrix $\boldsymbol{W}$, which represents an assigned positive weight of each vertex i. The smaller value of $\mathcal{F}\left(\mathcal{V}_1, \mathcal{V}_2\right)$ indicates more balanced partitioning when two different partitions with the same cut value are given. Therefore, a balanced partition with a small cut value can be obtained through the minimization of the new objective function $\mathcal{F}\left(\mathcal{V}_1, \mathcal{V}_2\right)$.

Given a graph G, let $\boldsymbol{L}$ and $\boldsymbol{W}$ be its *Laplacian* matrix and vertex weight matrix, respectively. Assume that we have a bipartitioning of $\mathcal{V}$ into $\mathcal{V}_1$ and $\mathcal{V}_2$ ($\mathcal{V} = \mathcal{S} \cup \mathcal{T}$). In addition, define a generalized partition vector $\boldsymbol{p}$ with elements

$$p_i = \begin{cases} +\sqrt{\dfrac{\mu_2}{\mu_1}}, & i \in \mathcal{V}_1 \\ -\sqrt{\dfrac{\mu_1}{\mu_2}}, & i \in \mathcal{V}_2 \end{cases}.$$

The generalized partition vector satisfies $\boldsymbol{p}^T \boldsymbol{We} = 0$ and $\boldsymbol{p}^T \boldsymbol{Wp} = w\left(\mathcal{V}\right)$, where $\mu_1 = w\left(\mathcal{V}_1\right)$ and $\mu_2 = w\left(\mathcal{V}_2\right)$.

Using above notations, and referring to some properties of the *Laplacian matrix* $\boldsymbol{L}$ and theorems presented in [7], we can achieve the following theorem.

THEOREM 1: Given the Laplacian matrix $\boldsymbol{L}$ of G, the vertex weight matrix $\boldsymbol{W}$ and a generalized partition vector $\boldsymbol{p}$, the Rayleigh Quotient is given by

$$\frac{\boldsymbol{p}^T \boldsymbol{Lp}}{\boldsymbol{p}^T \boldsymbol{Wp}} = \frac{cut\left(\mathcal{V}_1, \mathcal{V}_2\right)}{w\left(\mathcal{V}_1\right)} + \frac{cut\left(\mathcal{V}_1, \mathcal{V}_2\right)}{w\left(\mathcal{V}_2\right)}. \tag{4}$$

That is to say, the problem of finding the global minimum of objective function $\mathcal{F}\left(\mathcal{V}_1, \mathcal{V}_2\right)$ leads to the generalized partition vector $\boldsymbol{p}$. Since the problem is still NP-complete, it is a common practice to find a relaxation to the optimal generalized partition vector through the following theorem which is a standard result from linear algebra [36].

THEOREM 2: The problem

$$\min_{\boldsymbol{p} \neq 0} \frac{\boldsymbol{p}^T \boldsymbol{Lp}}{\boldsymbol{p}^T \boldsymbol{Wp}}, \quad \text{subject to } \boldsymbol{p}^T \boldsymbol{We} = 0$$

achieves the optimal solution when $\boldsymbol{p}$ is the eigenvector corresponding to the second smallest eigenvalue λ_2 of the generalized eigenvalue problem,

$$\boldsymbol{Lx} = \lambda \boldsymbol{Wx}. \tag{5}$$

3) ASSOCIATION WITH SVD

Next, we intend to give a exact definition of vertex weights. For convenience, we define the weight of each vertex equal to the sum of the weights of edges incident on it, i.e., $w\left(i\right) = \sum_k E_{ik}$, which is a cut objective called normalized-cut [37]. Based on this definition, the vertex weight matrix $\boldsymbol{W}$ equals the diagonal degree matrix $\boldsymbol{D}$.

According to THEOREM 2, it is clear that the key to get the minimum normalized cut is solving the second eigenvector of the generalized eigenvalue problem $\boldsymbol{Lx} = \lambda \boldsymbol{Wx}$. Next, we will illustrate how to find Web service and term clusters using our bipartite graph model. In our case,

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{D_1} & -\boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{D_2} \end{bmatrix}, \quad \textit{and } \boldsymbol{D} = \begin{bmatrix} \boldsymbol{D_1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D_2} \end{bmatrix},$$

where $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$ are diagonal degree matrices for term vertex set and Web service vertex set respectively, namely, $D_1(i,i) = \sum_j B_{ij}$, $D_2(j,j) = \sum_i B_{ij}$. The two matrices can be plugged into $\boldsymbol{Lx} = \lambda \boldsymbol{Wx}$,

$$\begin{bmatrix} \boldsymbol{D_1} & -\boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{D_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{D_1} & 0 \\ 0 & \boldsymbol{D_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}, \tag{6}$$

which can be rewritten in algebraic form on basis of the assumption that both $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$ are nonsingular. Through some algebraic calculation, we see the following two formulas,

$$\boldsymbol{D_1}^{-1/2}\boldsymbol{B}\boldsymbol{D_2}^{-1/2}\boldsymbol{v} = (1-\lambda)\boldsymbol{u},$$
$$\boldsymbol{D_2}^{-1/2}\boldsymbol{B}^T\boldsymbol{D_1}^{-1/2}\boldsymbol{u} = (1-\lambda)\boldsymbol{v},$$

where $\boldsymbol{u} = \boldsymbol{D_1}^{1/2}\boldsymbol{a}$ and $\boldsymbol{v} = \boldsymbol{D_2}^{1/2}\boldsymbol{b}$. It is obvious that the two formulas represent the process of singular value decomposition (SVD) of the normalized matrix $\boldsymbol{B_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{B}\boldsymbol{D_2}^{-1/2}$. Specifically, $\boldsymbol{u_2}$ and $\boldsymbol{v_2}$ are respectively the left and right singular vectors corresponding to the second largest singular value $\sigma_2 = 1 - \lambda_2$. It shows that the size of the *Laplacian matrix* $\boldsymbol{L}$ is much larger than matrix $\boldsymbol{B_n}$. For brief computation, we can take solving the left and right singular vectors corresponding to the second largest singular value of $\boldsymbol{B_n}$ as a replacement for the computation of the eigenvector of the second smallest eigenvalue of (6). Obviously, the left singular vector $\boldsymbol{u_2}$ offers us a bipartitioning of the terms and the right singular vector $\boldsymbol{v_2}$ offers the bipartitioning of Web services.

4) WC*CLUSTER* ALGORITHM

We discuss WC*Cluster* approach in the situation of multipartitioning as the bipartitioning is a special case where the number of term and Web service clusters is $k = 2$. Analogous to the fact that the second largest singular vectors contain bi-modal information, the $l = \lceil \log_2 k \rceil$ singular vectors $\boldsymbol{u}_2, \boldsymbol{u}_3, ..., \boldsymbol{u}_{l+1}$ and $\boldsymbol{v}_2, \boldsymbol{v}_3, ..., \boldsymbol{v}_{l+1}$ usually contain k-modal information about Web services. Given these vectors, the crucial task is to extract the ''best'' partition from them.

The optimal generalized partition vector for the multipartitioning problem must be k-valued. The classical k-means algorithm [38] can be utilized to find the best k-modal fit to the k l-dimensional points $\boldsymbol{c}_1, ..., \boldsymbol{c}_k$. From the previous section, we can form the l-dimensional eigenvector of $\boldsymbol{L}$ as follows,

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{D_1}^{-1/2}\boldsymbol{U} \\ \boldsymbol{D_2}^{-1/2}\boldsymbol{V} \end{bmatrix}, \tag{7}$$

where $\boldsymbol{U} = [\boldsymbol{u}_2, ..., \boldsymbol{u}_{l+1}]$ and $\boldsymbol{V} = [\boldsymbol{v}_2, ..., \boldsymbol{v}_{l+1}]$. Then the objective function to be minimized can be formed as

$$\sum_{j=1}^{k} \sum_{\boldsymbol{X}_2(i) \in \boldsymbol{c}_j} \left\| \boldsymbol{X}(i) - \boldsymbol{c}_j \right\|^2. \tag{8}$$

Combined the preprocessing of WSDL documents and tags, the WC*Cluster* algorithm can be described as **Algorithm 1**.

Algorithm 1 Framework of WC*Cluster*

Input:
A collection of WSDL documents and tags.
The number of web service and term clusters k.

Output:
The clustered $\boldsymbol{X}$.

1: Preprocess WSDL documents through extracting features, filtering stop words, stems extraction and preprocess tags.
2: Generate a $m \times n$ term-service matrix $\boldsymbol{B}_0$.
3: Get a new weighted matrix $\boldsymbol{B}$ by TF-IDF. Normalize $\boldsymbol{B}$ by $\boldsymbol{B_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{B}\boldsymbol{D_2}^{-1/2}$.
4: Compute $l = \lceil \log_2 k \rceil$ singular vectors of $\boldsymbol{B_n}$, $\boldsymbol{u}_2, \boldsymbol{u}_3, ..., \boldsymbol{u}_{l+1}$ and $\boldsymbol{v}_2, \boldsymbol{v}_3, ..., \boldsymbol{v}_{l+1}$, then form the matrix $\boldsymbol{X}$ mentioned in (7).
5: Run the k-means algorithm on matrix $\boldsymbol{X}$ to achieve the desired k clusters of web services and terms.
6: **return** $\boldsymbol{X}$ clustered by k-means algorithm.

The major computational cost of WC*Cluster* is generated in step 2 for generating $\boldsymbol{B}_0$ and step 4 for computing the left and right singular vectors. The computational cost of the former is $O(m^2 n)$ where m and n respectively represent the number of terms and Web services. The computation of SVD is a costly process and different algorithms of it need different cost, and the worst case is $O(N^3)$. The k-means algorithm in step 5 has the computational cost of $O(tk(m+n)l)$, where k is the number of clusters, t denotes the iterations and l is the number of singular vectors to be computed.

C. COMBINATION OF WORDS AND TAGS

A central problem in WTO is how to integrate tags with WSDL documents to improve the performance of Web service clustering. According to WC*Cluster*, this problem is equivalent to generating the weighted matrix $\boldsymbol{B}$ based on words and tags. Each column of $\boldsymbol{B}$ indicates a Web service represented as a vector in a real-valued space whose dimensionality is the size of terms. Thus the problem can be translated into how to model the Web services in the vector space. Now we introduce four strategies to model a service with a collection of words C_w and a collection of tags C_t as a vector V:

- ***Words Only.*** First, define V as $[w_1, w_2, ..., w_{|W|}]$ where w_i is the weight of term i, and $w_{|W|}$ means the number of words in C_w. The way to decide the weight can be based on some function of frequency of terms in C_w. This paper adopts TF-IDF technique which will be introduced in detail in Section V-C. Then V is l_2-normalized so that $\|V\|_2 = 1$.
- ***Tags as New Words.*** This strategy simply regards tags as additional words. V is defined as $[w_1, w_2, ..., w_{|W|}, w_{|W+1|}, ..., w_{|W|+|T|}]$ where w_i is the weight of word i for $i \leq |W|$ or the weight of tag $i - |W|$ for $i \geq |W|$. In this way, the same words respectively from

C_w and C_t should be treated as different terms. Then V is l_2-normalized.

- ***Tags as Words Times n.*** Combine the two collections, C_w and C_t, but each term in the tag collection C_t is treated as n terms. It means we use $C_w \cup (C_t \times n)$ instead of C_w used in *Words Only* model. For example, we extract the word "tourism" once from a WSDL document and the tag "tourism" from the corresponding Web service twice, then the Web service will be represented by the word "tourism" five times under the *Tag as Words Times 2* model.
- ***Words + Tags.*** Define V_w to be the words only vector mentioned in *Words Only* model. Similarly, we define the tags only vector as V_t. Then the *Words+Tags* vector can be placed as $V_{w+t} = \left[\sqrt{\frac{1}{2}}V_w, \sqrt{\frac{1}{2}}V_t\right]$. It could be interpreted as a concatenation of the two l_2-normalized vectors with equal weight.

Words Only is the equal of the proposed WC*Cluster* taking no account of tag information and could be used as baseline among the mentioned four strategies. The strategy of *Tags as New Words* regards tagging data as independent supplementary information. *Words+Tags* combines words and tags in the similar way compared with *Tags as New Words* except for the order of vector l_2-normalization and concatenation. Both the two strategies treat tags as separate information and weight them independently of any words. But for *Tags as Words Times n*, it mixes word and tag collections and emphasizes the weight of terms in tag set. Thus, it might be more sensitive to the selection of n and the quality of tagging data.

IV. TAG RECOMMENDATION

We propose to improve the quality of tagging data before data integration by considering the inherently uncontrolled, ambiguous, and overly personalized tagging data. It should be noted that the distribution of tags is not uniform as illustrated in Fig.1 Some Web services have more than 10 tags while some contain only 1 or 2 tags. The Web services with few tags may reduce the effect of WTO because of the limitation of additional information provided by few tags. To handle this problem, we propose to recommend some relevant tags to these web services for better clustering performance.

Figure 6 shows an overview of the tag recommendation process. It can be found that the process is mainly divided into two steps. In the first step, given a Web service with tags, we calculate the co-occurrence between user-defined tags and any other tags, then select the top-k co-occurring tags of each user-defined tag in order to generate an ordered list of m candidate tags. The number of k is set as 4 showed in Fig.6. There are three defined tags: *HR*, *business* and *unknown* and a list of 4 co-occurring tags of each is derived. For instance, the top-4 co-occurring tags of *HR* are *unknown*, *business*, *employee* and *management*. There are some normalization methods for co-occurrence calculation, and here we choose

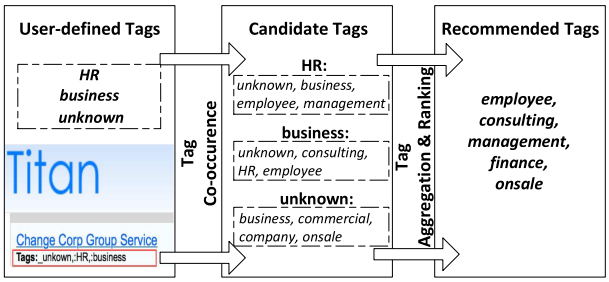


FIGURE 6. Overview of the tag recommendation process.

Jaccard coefficient method as follows:

$$J(t_i, t_j) = \frac{|t_i \cap t_j|}{|t_i \cup t_j|}, \tag{9}$$

where $|t_i \cup t_j|$ denotes the number of Web services that have t_i or t_j, and $|t_i \cap t_j|$ denotes the number of Web services that have both t_i and t_j. A list of candidate tags could be obtained for each user-defined tag after the co-occurrence calculation step.

The second step is a tag aggregation and ranking step in which we rank the candidate tags and select the top-n tags as the final recommended tags. In this paper, we present two aggregation methods based on *Vote* and *Sum* method.

- ***Vote.*** The voting method is utilized to compute a score for each candidate tag $c \in C$ (C is the set of all candidate tags). The value of $vote(u, c)$ between a candidate tag c and each user-defined tag $u \in U$ (U is the set of all the user-defined tags) is computed as follows:

$$vote(u, c) = \begin{cases} 1 & \text{if } c \in C_u \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

  where C_u refers to the set of candidate tags of user-defined tag u. A list of final recommended tags can be obtained by counting the voting results:

$$score(c) = \sum_{u \in U} vote(u, c) \tag{11}$$

- ***Sum.*** The summing method sums over the co-occurrence values between a candidate tag c and each user-defined tag u as the score of tag c:

$$score(c) = \sum_{u \in U} Co(u, c), \tag{12}$$

  where $Co(u, c)$ can be computed by (9).

The performance of these two tag recommendation approaches will be evaluated in the next section.

V. EXPERIMENT

We first introduce the evaluation metrics and the details of preprocessing. Then, we evaluate the performance of WC*Cluster* approch and WTO framework, including the impact of data integration strategies and tag recommendation. Further, the performance of tag recommendation is evaluated.

A. EXPERIMENT SETUP

To further demonstrate the adaptability of the proposed approach, the experiments are based on three datasets of online Web services, D_1, D_2 and D_3, gathered from real-world Web service search engine Titan. Dataset D_1 contains 114 Web services offering WSDL documents for WC*Cluster* evaluation which is to be discussed in Section V-D Dataset D_2 is a set of 120 tagged Web services, and is used to evaluate tagged Web services co-clustering, i.e., WTO. We make a manual classification of these Web services to easily evaluate the experiment results by comparing with the other algorithm. Section V-F employs dataset D_3 consisting of tag records of 1049 Web services to evaluate the performance of tag recommendation. Particularly, all the datasets can be downloaded from the provided link[4].

The preprocessing experiments in our paper are implemented with JDK 8.0.110, Eclipse 4.4.0, co-clustering algorithm is executed with Matlab7.13.0, and tag recommendation is implemented with Microsoft Visual Studio 2010. All these processes are conducted on a ASUS K40ID machine with an *2.20 GHz Intel Core 2 Duo T6670 CPU* and *2GB RAM*, running *Windows7 OS*.

B. EVALUATION MEASURES

1) PRECISION & RECALL

The evaluation of clustering is a tricky business. Fortunately, the Web services in the dataset used in this paper are categorized, so that we can compare the experimental results with true class labels. In this paper, we use two widely employed metrics, precision and recall, to evaluate the performance of WC*Cluster*. Precision is a measure of exactness while recall shows completeness [39]. The two metrics can be respectively described as follows,

$$\begin{aligned} Precision_{c_i} &= \frac{succ(c_i)}{succ(c_i) + mispl(c_i)} \\ Recall_{c_i} &= \frac{succ(c_i)}{succ(c_i) + missed(c_i)}, \end{aligned} \tag{13}$$

where c_i represents cluster i, $succ(c_i)$ denotes the number of Web services successfully placed in cluster c_i, $mispl(c_i)$ is the number of Web services incorrectly placed in cluster c_i, and $missed(c_i)$ represents the number of Web services that should be clustered in c_i but are incorrectly placed in other clusters.

2) F1 SCORE

An integrated metric is required when we compare four strategies for the integration of words and tags. F1 score could be treated as a balance between precision and recall. It should be noted that precision and recall here differ from the ones mentioned above, and we use precision* and recall* to denote them for distinction. Thus we choose F1 score as the metric to compare different data integration strategies. Precision* and recall* are computed over pairs of Web services for which two label assignments either agree or disagree. We consider that pairs of Web services are either the same class or different classes according to the manual classification predetermined. The proposed clustering algorithm predicts whether any given pair has the same or different clusters. Consider all of the pairs of Web services, we can conclude 4 cases:

1) **True Positives (TP)**: The clustering algorithm placed the pair of two Web services into the same cluster, and the predetermined manual classification has them in the same class.
2) **False Positives (FP)**: The clustering algorithm placed the pair of two Web services into the same cluster, but the predetermined manual classification has them in different classes.
3) **True Negatives (TN)**: The clustering algorithm placed the pair of two Web services into different clusters, and the predetermined manual classification has them in different classes.
4) **False Negatives (FN)**: The clustering algorithm placed the pair of two Web services into different clusters, but the predetermined manual classification has them in the same class.

We calculate precision* as $\frac{TP}{TP+FP}$, and calculate recall* as $\frac{TP}{TP+FN}$. Then F1 score could be obtained by $\frac{2\times precision*\times recall*}{precision*+recall*}$.

3) TAG RECOMMENDATION METRICS

According to [40] [41], it is reasonable for recommendation evaluation to consider the division of tags of each Web services into two sets: (1) the past tags, (2) the future tags. Thus, for a Web service, we generate the recommended tags based only on the past tags. For example, assume a Web service has been tagged with 6 tags and 4 of them are assigned to the past set, then we compute similarities only based on the past 4 tags to predict the rest 2 future tags. The evaluation of tag recommendation in this way is more indicative for real-world applications.

We adopt the following metrics to measure the effectiveness of tag recommendation based on the evaluation situation mentioned above.

1) *Top-k accuracy*: Percentage of Web services correctly tagged by at least one of the top-k recommended tags.
2) *Exact-k accuarcy*: Percentage of Web services correctly tagged by the kth recommended tag.
3) *Tag-precision*: The ratio of the number of relevant tags in the top-k list (i.e., those in the top-k list that belong in the future set of tags) to k.
4) *Tag-recall*: The ratio of the number of relevant tags in the top-k list to the total number of relevant tags (all tags in the future set).

C. PREPROCESSING

We first describe the detailed data preprocessing mentioned in Section III-B before the evaluation process.

[4]Dataset: http://zjujason.com/data.html

1) STOP WORDS REMOVAL

We find many words do not make any sense regarded as stop words (i.e., "be", "the", "above", "to") through the observation on WSDL documents. In WC*Cluster* approach, filtering stop words is to eliminate the noise of the parsed WSDL documents. As there is not a definite list of stop words for Web services, we employ a widely accepted stop words list [5] downloaded from the Internet.

2) STEMMING

Stems extraction is another problem to be solved except for the stop words. The Porter Stemmer algorithm is a process for removing the commoner morphological and inflexional endings from words in English. For example, "create" and "created" have the same meaning, while the computer will view them as two different words. The original stemming algorithm was proposed in [42]. In our experiment, we apply the Porter Stemmer algorithm of Java version.

3) TF-IDF

The TF-IDF value of each word in a document increases with the its frequency in the document, while offsets by its frequency in the whole collection. In this paper, we utilize TF-IDF to weight terms after the term-Web service frequency matrix is generated, and the new weighted matrix could be used for clustering.

D. PERFORMANCE OF WCCLUSTER

As mentioned, dataset D_1 is employed for evaluating the performance of WC*Cluster* in this section. D_1 consists of following categories: "Business", "Weather", "Bioinformatics", "Translation", "Music" and "HR". Since the performance of clustering different number of categories has distinction. We use two subsets of dataset D_1 in the size of 3 and 5 categories respectively.

1) CLUSTERING PERFORMANCE ON CATEGORIES3

In this subsection, we show partitioning result based on a dataset extracted from D_1 which we call "Categories3". Categories3 includes three categories: "Business", "Weather", and "Bioinformatics". As the true class label of each Web service is pre-known, a confusion matrix can be formed to show the co-clustering performance. Besides, the values of metrics (i.e., precision and recall) can be easily derived from the matrix.

Table 1 clearly demonstrates the effectiveness of WC*Cluster* on dataset Categories3. In the confusion matrix, it can be found that cluster $\mathcal{D}_1$ almost consists entirely of the "Business" category. Cluster $\mathcal{D}_2$ includes 20 Web services of which 19 are categorized as "Weather". And we find 10 out of 11 Web services in $\mathcal{D}_3$ belong to "Bioinformatics". It is worth mentioning that all the Web services categorized as "Bioinformatics" are placed in the same cluster. The precision and recall for each cluster are more than 90%, which indicates that WC*Cluster* works well on the collection of 3 categories.

WC*Cluster* can discover the structure in the sparse term-service matrix. Figure 7 shows the original term-service matrix and the realigned matrix generated by arranging rows and columns according to the cluster order to reveal the co-clusters. In Fig.7, WC*Cluster* displays the underlying sparsity structure of various co-clusters including 3 Web service

[5] http://www.ranks.nl/stopwords

TABLE 1. Clustering results for categories3.

	Business	Weather	Bioinformatics	Precision (%)	Recall (%)
$\mathcal{D}_1$	28	1	0	96.7	93.3
$\mathcal{D}_2$	1	19	0	95.0	95.0
$\mathcal{D}_3$	1	0	10	90.9	100.0

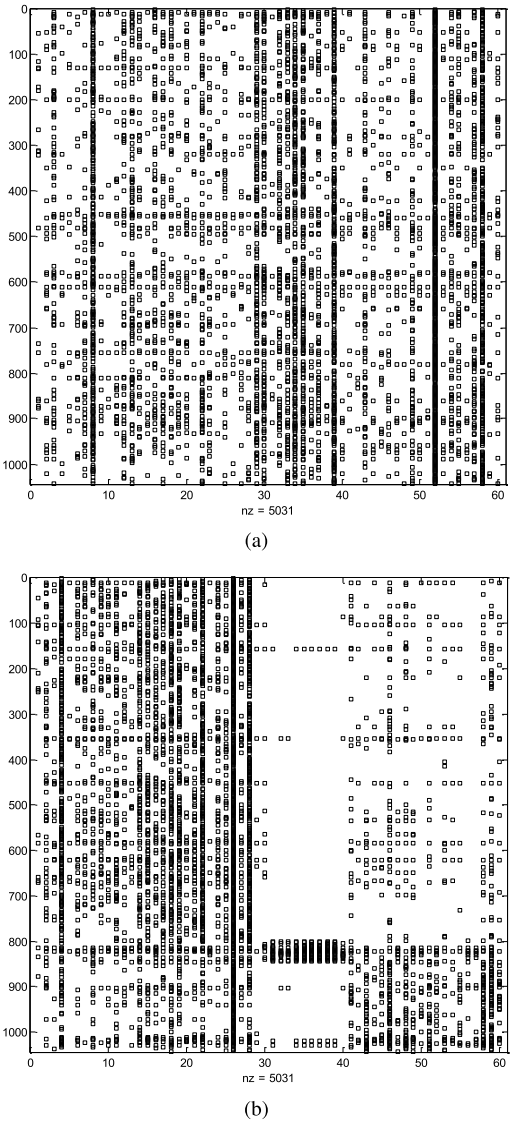


FIGURE 7. Sparsity structure of word-WSDL document co-occurrence matrix before (a) and after (b) co-clustering. The shaded regions represent the non-zero entries.

TABLE 2. Confusion matrix of the clustering result for categories5.

	Bioinformatics	Music	Translation	HR	Weather
$\mathcal{D}_1$	10	0	0	0	0
$\mathcal{D}_2$	0	10	1	0	2
$\mathcal{D}_3$	0	0	11	0	0
$\mathcal{D}_4$	0	0	0	24	0
$\mathcal{D}_5$	0	0	8	0	18

clusters and the corresponding 3 term clusters. According to some block diagonal sub-structure, we can find that some term clusters are highly indicative of individual Web service clusters. However, the dense sub-block at the bottom in the second panel shows that some clusters may have more uniform distribution over the Web service clusters.

2) CLUSTERING PERFORMANCE ON CATEGORIES5

We intend to show the performance of co-clustering based on the dataset Categories5 extracted from D_1, which contains 5 categories: "Bioinformatics", "Weather", "HR", "Translation", and "Music". Table 2 shows the partitioning results. It is found that $\mathcal{D}_1$, $\mathcal{D}_3$, and $\mathcal{D}_4$ are purely from the corresponding categories. The Web services belonging to "Bioinformatics", "Music", "HR" are exactly placed in the related clusters, as indicated by 100% recall values shown in Table 3. Cluster $\mathcal{D}_2$ and $\mathcal{D}_5$ are mixed by 2 or 3 original classes, which leads to the relatively low precision. It may attribute to the mutual correlation between two categories, "Translation" and "Weather". Another observation of the confusion matrix is that Web services supposed to belong to "Translation" are divided into 3 different clusters, because of the relatively uniform distribution of the terms extracted from Web services.

K-means method has been widely employed in Web service clustering task based on the semantic similarity between WSDL documents. Thus, we take it as baseline to make comparison with WC*Cluster*. Table 3 demonstrates the performance results of two methods based on five identified groups of Web services. WC*Cluster* shows its superiority with the higher precision and recall values for most identified categories. For example, WC*Cluster* improves precision by 30.8% and recall by 10% for category "Bioinformatics". It is noticed that the recall value for each cluster generated by WC*Cluster* is higher than that of the baseline.

E. PERFORMANCE OF WTO FRAMEWORK

We will evaluate the impact of different data integration strategies, the impact of tag recommendation process, and the overall performance of WTO framework in the following.

1) IMPACT OF DATA INTEGRATION STRATEGIES

We evaluate the impact of four proposed data integration strategies to the performance of service clustering based on precision*, recall*, and the comprehensive metric F1 score. Fig.8 shows the performance comparison results on 120 tagged Web services. As the *Words Only* strategy is equal to the WC*Cluster*, thus it is regarded as baseline. It could be observed that each strategy that takes tagging data into account performs better than baseline in all three metrics. Except for *Words Only* model, we can also find that the performance of *Tags as New Words* model is the best in all three metrics, while the rest three have similar performances. The above observations convincingly demonstrate that tagging data has positive influence on the performance of Web services clustering. In particular, *Tags as New Words* model can more effectively integrate tagging data as complementary information.

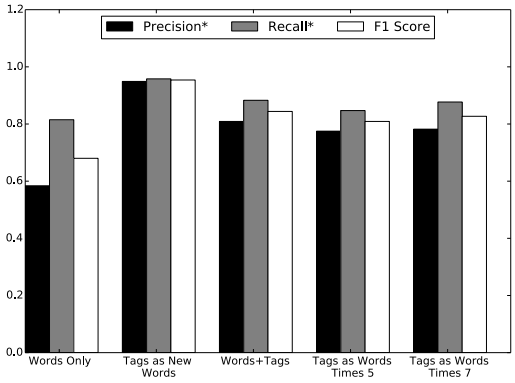


FIGURE 8. Impact of Data Integration Strategies.

2) IMPACT OF TAG RECOMMENDATION

We evaluate the impact of tag recommendation to the performance of service clustering. In particular, *Sum* tag recommendation approach is employed, and the number of recommended tags is set as 4. Table 4 shows the performance comparison of service clustering with and without

TABLE 3. Performance comparison between WC*Cluster* and *k*-means.

Cluster	WC*Cluster*		*k*-means	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)
Bioinformatics	**100.0**	**100.0**	69.2	90.0
Music	**76.9**	**100.0**	45.5	100.0
Translation	**100.0**	**55.0**	100.0	55.0
HR	**100.0**	**100.0**	100.0	83.3
Weather	69.2	**90.0**	100.0	90.0

TABLE 4. Impact of tag recommendation.

Strategy	Precision*		Recall*		F1 Score	
	Without	With	Without	With	Without	With
Tags as New Words	0.949	**0.965**	0.958	**0.978**	0.954	**0.971**
Words+Tags	0.809	**0.965**	0.883	**0.978**	0.844	**0.971**
Tags as Words Times 5	**0.775**	0.753	0.847	**0.863**	**0.809**	0.804
Tags as Words Times 7	0.782	**0.799**	0.877	**0.895**	0.827	**0.845**

TABLE 5. Performance evaluation of WTO framework.

Cluster	k-means		WC*Cluster*		WTO^1		WTO^2	
	Precision%	Recall%	Precision%	Recall%	Precision%	Recall%	Precision%	Recall%
Academic	75.0	52.2	32.5	56.5	95.8	100.0	95.8	100.0
HR	100.0	96.7	100.0	100.0	100.0	100.0	100.0	100.0
Tourism	33.3	6.5	100.0	9.7	96.8	96.8	96.9	100.0
University	91.7	42.3	73.5	96.2	96.2	96.2	100.0	96.2
Genetics	17.5	100.0	69.2	90.0	100.0	90.0	100.0	90.0

tag recommendation, by considering all data integration strategies. From Table 4, it could be observed the employment of tag recommendation improves the performance of service clustering in almost all cases in terms of precision*, recall* and F1 score. In particular, the degree of the improvement is quite different. For example, the precision* of *Tags as New Words* model increases by 0.016 while that of *Words+Tags* model increases by 0.156 so that the latter catches up with the former, which indicates that *Words+Tags* is more sensitive to the amount and quality of tagging data. F1 score reflects the overall effectiveness of a strategy. And it is distinct that *Tags as New Words* and *Words+Tags* outperform the rest models, which is basically in conformity with the results of co-clustering before tag recommendation. It is mainly led by the alternative information channel that could be counted separately and weighted independently of words provided by tagging data. As discussed above, the employment of tagging data improves the performance of Web service clustering, it is reasonable that the larger tag collections obtained by tag recommendation should be more beneficial to the clustering performance since more tag information has been received.

3) WTO PERFORMANCE

To evaluate the performance of the proposed WTO framework, we implement four Web service clustering approaches:

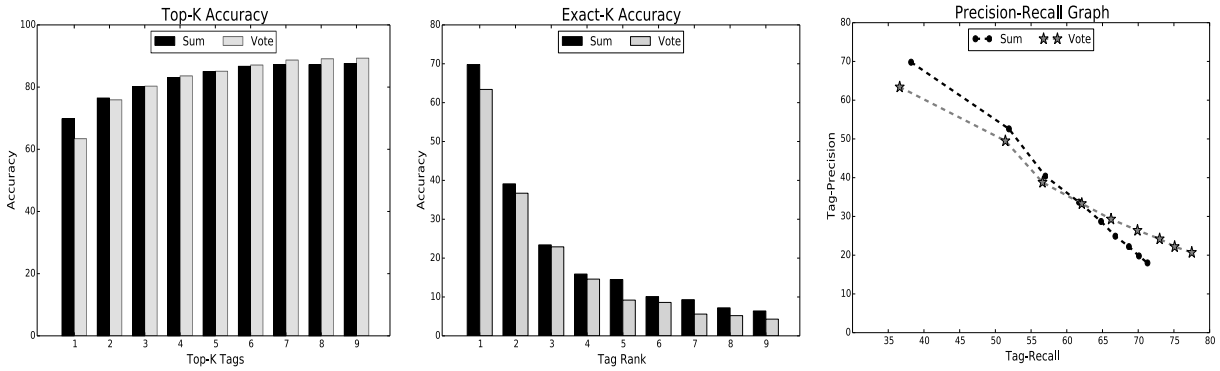
- ***k*-means.** The traditional clustering algorithm based on the semantic similarity between WSDL documents.
- **WC*Cluster*.** The co-clustering approach proposed in our early work.
- **WTO^1.** Based on WCCluster, integrating tagging data with WSDL documents by using *Tags as New Words* model to co-cluster Web services.
- **WTO^2.** In this approach, we first implement the tag recommendation process, and then cluster Web services by using WTO^1. In particular, *Sum* method is employed in this approach.

Table 5 reports the performance comparison of above 4 Web service clustering approaches. It can be observed that the approaches proposed in this paper (WTO^1, WTO^2) outperform the traditional k-means algorithm and WC*Cluster* in almost all cases. Using category "Academic" as an example, WTO (WTO^1, WTO^2) makes the improvement with 20.8% in precision and 47.8% in recall comparing with k-means, and for WC*Cluster*, the promotion of precision and recall are 63.3% and 43.5% respectively. As mentioned above, tags promise a reliable source of information on the similarity calculation between Web services. Utilizing these information improves the performance of Web service clustering. Both of WTO^1 and WTO^2 work well, since the worst precision and recall are able to reach 90%. Moreover, Table 5 shows that WTO^2 which contains tag recommendation is better than WTO^1 approach. It demonstrates that adding relevant tags to the original tag information can improve the performance of WTO framework.

F. PERFORMANCE OF TAG RECOMMENDATION

We select 1049 Web services which contain at least two tags to measure the performance of tag recommendation method. In particular, for each Web service, 50% tags are identified as the past tags, and the left 50% as the future tags. The experiment is processed based on two recommendation methods, *Sum* and *Vote*, and at most top 9 tags returned.

The comprehensive performance evaluation on the 1049 Web services data set is described in Fig.9. The performance in terms of top-k accuracy is showed in Fig.9(a), where we can observe that the *Sum* method makes 69.8% correct recommendation when only top-1 tag is returned, for which *Vote* method is 63.4%. The top-k accuracy is gradually improved with the increase of the number of recommended tags. The accuracy of top-9 tags for *Sum* is about 87.6% while that of *Vote* reaches 89.3%. Note that *Sum* has the higher accuracy in top-1 and top-2 situation, while *Vote* holds the better result since the number of recommended tags



(a) (b) (c)

FIGURE 9. Performance of tag recommendation on 1049 tagged web services.

increase to 3. The gap between *Sum* and *Vote* in terms of top-*k* accuracy is small as a whole. Fig.9(b) illustrates the performance comparison in terms of exact-*k* accuracy. It is showed that both *Sum* (69.8%) and *Vote* (63.4%) achieve their best performance at the exact top-1 tag, and the performance becomes worse with the decrease of *k*. Additionally, *Sum* outperforms *Vote* all the time in terms of exact-*k* accuracy.

In Fig.9(c), we depict a tag-precision versus tag-recall curve for two methods, *Sum* and *Vote*. It is distinct that tag-precision of each method falls as *k* increases. In contrast, tag-recall for both methods increases with the decrease of *k*. *Sum* reaches 69.8% tag-precision when a top-1 list of tags is recommended. *Vote* gets a tag-precision of 63.4% by comparison. It is analogous to top-*k* accuracy, *Vote* exceeds *Sum* in both tag-precision and tag-recall after *k* reaches 4. And *Vote* reaches a maximum tag-recall of 77.5%, while *Sum* achieves 71.3%. Fig.9(c) demonstrates that *Vote* outperforms *Sum* when the number of recommended tags is relatively large, such as 4 showed here.

VI. CONCLUSION

It has been widely accepted that clustering Web services into functional similar classes is an effective approach to improve Web service discovery. In this paper, we propose to utilize both WSDL documents and tagging data to improve the accuracy of Web service clustering. In particular, we implement the integration of WSDL documents and tagging data based on four distinct strategies in the proposed WTO framework. In addition, we present a tag recommendation process to relax the limitation of WTO generated by the uncontrolled quality of tagging data. To evaluate the performance of WTO, a real Web service dataset crawled from Titan is employed. We evaluate the impact of data integration strategies, tag recommendation methods, and the overall clustering performance of WTO framework, respectively. The experimental results demonstrate the effectiveness of the proposed WTO framework.

Efficiency issues will be considered in our future work. In particular, we will integrate our WTO approach into Hadoop and MapReduce framework to improve the efficiency for the case of massive and distributed Web services.

REFERENCES

[1] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Web service discovery mechanisms: Looking for a needle in a haystack," in *Proc. Int. Workshop Web Eng.*, vol. 38, 2004, p. 25.

[2] E. Al-Masri and Q. H. Mahmoud, "Investigating Web services on the world wide Web," in *Proc. 17th Int. Conf. World Wide Web*, Apr. 2008, pp. 795–804.

[3] R. Nayak, "Data mining in Web services discovery and monitoring," *Int. J. Web Services Res.*, vol. 5, no. 1, pp. 63–81, Jan. 2008.

[4] W. Liu and W. Wong, "Web service clustering using text mining techniques," *Int. J. Agent-Oriented Softw. Eng.*, vol. 3, no. 1, pp. 6–26, Feb. 2009.

[5] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering WSDL documents to bootstrap the discovery of Web services," in *Proc. Int. Conf. Web Services (ICWS)*, Jul. 2010, pp. 147–154.

[6] T. Liang, L. Chen, H. Ying, and J. Wu, "Co-clustering WSDL documents to bootstrap service discovery," in *Proc. 7th Int. Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2014, pp. 215–222.

[7] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2001, pp. 269–274.

[8] Y. Yin, J. Xia, Y. Li, Y. Xu, W. Xu, and L. Yu, "Group-wise itinerary planning in temporary mobile social network," *IEEE Access*, vol. 7, pp. 83682–83693, 2019.

[9] H. Gao, W. Huang, Y. Duan, X. Yang, and Q. Zou, "Research on cost-driven services composition in an uncertain environment," *J. Internet Technol.*, vol. 20, no. 3, pp. 755–769, 2019.

[10] H. Gao, H. Miao, L. Liu, J. Kai, and K. Zhao, "Automated quantitative verification for service-based system design: A visualization transform tool perspective," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 10, pp. 1369–1397, 2018.

[11] A. L. Lemos, F. Daniel, and B. Benatallah, "Web service composition: A survey of techniques and tools," *ACM Comput. Surv. (CSUR)*, vol. 48, no. 3, p. 33, Feb. 2016.

[12] L. Chen, Q. Yu, P. Yu, and J. Wu, "Ws-hfs: A heterogeneous feature selection framework for Web services mining," in *Proc. Int. Conf. Web Services*, Jul. 2015, pp. 193–200.

[13] Y. Zhong, Y. Fan, W. Tan, and J. Zhang, "Web service recommendation with reconstructed profile from mashup descriptions," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 468–478, Apr. 2016.

[14] A. Bukhari and X. Liu, "A Web service search engine for large-scale Web service discovery based on the probabilistic topic modeling and clustering," *Service Oriented Comput. Appl.*, vol. 12, no. 2, pp. 169–182, Jun. 2018.

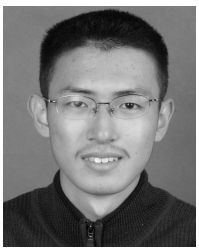
[15] Y. Yin, W. Zhang, Y. Xu, H. Zhang, Z. Mai, and L. Yu, "QoS prediction for mobile edge service recommendation with auto-encoder," *IEEE Access*, vol. 7, pp. 62312–62324, 2019.

[16] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," in *Mobile Networks and Applications*. New York, NY, USA: Springer, 2019, pp. 1–11.

[17] M. Klusch, B. Fries, and K. Sycara, "Automated semantic Web service discovery with OWLS-MX," in *Proc. 5th Int. Joint Conf. Auto. Agents-multiagent Syst.*, May 2006, pp. 915–922.
[18] Y. Wu, C.-G. Yan, Z. Ding, G.-P. Liu, P. Wang, C. Jiang, and M. Zhou, "A multilevel index model to expedite Web service discovery and composition in large-scale service repositories," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 330–342, May/Jun. 2016.
[19] Q. Yu and A. Bouguettaya, "Efficient service skyline computation for composite service selection," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 776–789, Apr. 2013.
[20] B. Benatallah, M.-S. Hacid, A. Leger, C. Rey, and F. Toumani, "On automating Web services discovery," *VLDB J.*, vol. 14, no. 1, pp. 84–96, 2005.
[21] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for Web services," in *Proc. 13th Int. Conf. Very Large Data Bases*, vol. 30, Sep. 2004, pp. 372–383.
[22] D. Skoutas, D. Sacharidis, A. Simitsis, and T. Sellis, "Ranking and clustering Web services using multicriteria dominance relationships," *Trans. Services Comput.*, vol. 3, no. 3, pp. 163–177, Jul./Sep. 2010.
[23] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu, "Category-aware API clustering and distributed recommendation for automatic mashup creation," *IEEE Trans. Services Comput.*, vol. 8, no. 5, pp. 674–687, Sep. 2015.
[24] C. Platzer, F. Rosenberg, and S. Dustdar, "Web service clustering using multidimensional angles as proximity measures," *Trans. Internet Technol. (TOIT)*, vol. 9, no. 3, p. 11, Jul. 2009.
[25] N. Zhang, J. Wang, K. He, Z. Li, and Y. Huang, "Mining and clustering service goals for restful service discovery," *Knowl. Inf. Syst.*, vol. 58, no. 3, pp. 669–700, Mar. 2019.
[26] Y. Cheng and G. M. Church, "Biclustering of expression data," in *Proc. ISMB*, vol. 8. 2000, pp. 93–103.
[27] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Proc. 5th Int. Conf. Data Mining*, Nov. 2005, p. 4–pp.
[28] J. Liu and M. Shah, "Scene modeling using co-clustering," in *Proc. 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–7.
[29] D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina, "Clustering the tagged Web," in *Proc. 2nd ACM Int. Conf. Web Search Data Mining*, Feb. 2009, pp. 54–63.
[30] N. Cravino, J. Devezas, and Á. Figueira, "Using the overlapping community structure of a network of tags to improve text clustering," in *Proc. 23rd ACM Conf. Hypertext Social Media*, Jun. 2012, pp. 239–244.
[31] L. Chen, J. Wu, Z. Zheng, M. R. Lyu, and Z. Wu, "Modeling and exploiting tag relevance for Web service mining," *Knowl. Inf. Syst.*, vol. 39, no. 1, pp. 153–173, Apr. 2014.
[32] J. Wu, L. Chen, Z. Zheng, M. R. Lyu, and Z. Wu, "Clustering Web services to facilitate service discovery," *Knowl. Inf. Syst.*, vol. 38, no. 1, pp. 207–229, 2014.
[33] W. Lo, J. Yin, and Z. Wu, "Accelerated sparse learning on tag annotation for Web service discovery," in *Proc. Int. Conf. Web Services (ICWS)*, Jul. 2015, pp. 265–272.
[34] B. Cao, X. F. Liu, J. Liu, and M. Tang, "Domain-aware Mashup service clustering based on LDA topic model from multiple data sources," *Inf. Softw. Technol.*, vol. 90, pp. 40–54, Oct. 2017.
[35] L. Chen, L. Hu, Z. Zheng, J. Wu, J. Yin, Y. Li, and S. Deng, "Wtcluster: Utilizing tags for Web services clustering," in *Service-Oriented Computing*. Cham, Switzerland: Springer, 2011, pp. 204–218.
[36] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD, USA: JHU Press, 2012, vol. 3.
[37] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[38] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[39] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel, "Performance measures for information extraction," in *Proc. DARPA Broadcast News Workshop*, Feb. 1999, pp. 249–252.
[40] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 116–142, 2015.
[41] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.
[42] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

TINGTING LIANG received the Ph.D. degree in computer science and technology from Zhejiang University. She is currently working with Hangzhou Dianzi University. Her current research interests include data mining, recommender systems, multiview learning, deep learning, and service oriented computing. Her articles have been published in some well-known conference proceedings and international journals, such as ICDM, ICSOC, ICWS, TSC, KBS, and WWWJ.

YISHAN CHEN received the B.S. degree from the College of Engineering, Nanjing Agricultural University, Nanjing, China, in 2017. She is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University. Her current research interests include cloud computing, service computing, edge computing, and big data.



WEI GAO received the B.S. degree in computer science from the Zhejiang University of Technology, in 2014. He is currently pursuing the Ph.D. degree with the College of Computer Science, Zhejiang University. His current research interests include service computing, recommender systems, and data mining.

MING CHEN received the Ph.D. degree in computer science and technology from Zhejiang University. He is currently a Chief Technology Officer with Hithink RoyalFlush Information Network Company, Ltd. and the Zhejiang Hithink RoyalFlush Artificial Intelligence Research Institute. He has published more than ten related academic articles and has rich practical experience in the application of AI in financial field and medical field. His current research interests include machine learning, big data, and computer vision.

MEILIAN ZHENG was born in Zhejiang, China, in 1972. She received the Ph.D. degree in business management from the School of Management, Zhejiang University, Hangzhou, China, in 2008. She is currently working with the Zhejiang University of Technology. Her current research interests include corporate financial and financial data analysis.

JIAN WU received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 1998 and 2004, respectively. He is currently a Full Professor with the College of Computer Science, Zhejiang University. He is also the Director of the Real Doctor AI Research Centre, Zhejiang University and the Vice-President of the National Research Institute of Big Data of Health and Medical Sciences, Zhejiang University. His current research interests include medical artificial intelligence, service computing, and data mining. He is a CFF member, CCF TCSC member, CCF TCAPP member, and a member of the 151 Talent Project of Zhejiang Province.