Co-clustering WSDL Documents to Bootstrap Service Discovery

Tingting Liang, Liang Chen, Haochao Ying, Jian Wu
College of Computer Science & Technology
Zhejiang University
Hangzhou, China,310027
Email: {liangtt,cliang,haochaoying,wujian2000}@zju.edu.cn

***Abstract*—With the increasing popularity of web service, it is indispensable to efficiently locate the desired service. Utilizing WSDL documents to cluster web services into functionally similar service groups is becoming mainstream in recent years. However, most existing algorithms cluster WSDL documents solely and ignore the distribution of words rather than cluster them simultaneously. Different from the traditional clustering algorithms that are on one-way clustering, this paper proposes a novel approach named WC*Cluster* to simultaneously cluster WSDL documents and the words extracted from them to improve the accuracy of clustering. WC*Cluster* poses co-clustering as a bipartite graph partitioning problem, and uses a spectral graph algorithm in which proper singular vectors are utilized as a real relaxation to the NP-complete graph partitioning problem. To evaluate the proposed approach, we design comprehensive experiments based on a real-world data set, and the results demonstrate the effectiveness of WC*Cluster*.**

Keywords-Web service, WSDL documents clustering, bipartite graph partitioning, co-clustering

I. Introduction

Service-oriented Computing (SOC) which is a computing paradigm driven by Service-oriented Architecture (SOA) realizes through standardized web services technologies. A service in SOA is defined by a Web interface which supports interpretable operations between different software applications, which is matched with the definition made by W3C: a Web service is a software system designed to support interoperable machine-to-machine interaction over a network.[1] Currently, web services are the most promising SOC-based technology. They utilize the Internet as the communication medium and open Internet-based standards which contain the Web Service Description Language (WSDL) for services definition, the Simple Object Access Protocol (SOAP) for message transmission and the Business Process Execution Language for Web Services (BPEL4WS)[2].

Web service discovery is usually applied in the scenario that a requester wishes to initiate an interaction with a provider, but it does not know what provider agent it wished to engage, then the requester may need to "discover" a proper candidate. Discovery is "the act of locating a machine-processable description of a web service that may have been previously unknown and that meets certain functional criteria." [3] With the explosive growth of the number of registered web services, discovering the appropriate web services for users is becoming considerably important. As mentioned by Garofalakis et al. [4], web service discovery mechanisms are originated from the agent match-making paradigm through a middle agent and expand to apply UDDI registry offering the requirements. Then other trends for discovering services such as web service search engine emerged. Since Al-Masri et al. [5] point out that more than 53% of the UDDI Business Registry (UBR) registered services are invalid, while 92% of web services cached by search engines are valid and active, which indicates an increasing trend of adopting a service-engine based model for web service discovery.

Using search engines to search for web services easily results in bottleneck during the process, especially for non-semantic Web service, because of the typical limitation of keyword matching. In order to address the drawbacks of traditional web service search engines, the approach of clustering services into similar functional groups based on the WSDL documents is widely used. A WSDL is an XML format file for describing web services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information [6]. Except the operating message, we also can find description text from the interface names. Figure 1 is a simple example of WSDL file. We can extract words reflecting the web service function to a certain extend. For example, from the WSDL file of figure 1, words "Temperature", "Pressure" , "Sky" and "Visibility" et al. can be extracted, and these words imply this web service is about weather.

In previous work, researchers mainly considered the similarity measure between web services and paid attention to define similarity functions and calculate the similarity based on the features extracted from the WSDL documents. In [7], Liu et al. proposed techniques to automatically cluster WSDL documents on the web into functionally similar service groups. Eglazzar et al. [8]. proposed a novel approach to cluster web services based on function similarities by mining WSDL documents.

The common premise of existing algorithms for Web service clustering is that most of them are on one-way clustering which means just clustering WSDL documents and ignoring the words extracted from WSDL documents. And these traditional algorithms mainly considered the semantic similarity of WSDL documents through the words or titles. However, different service developers may make quite different descrip-

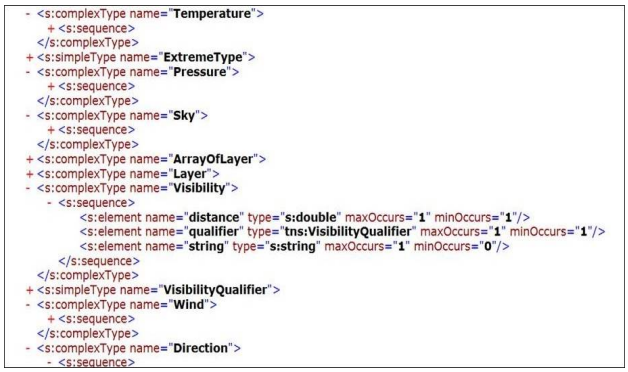

Fig. 1. An Example of WSDL Document

tions of the same function in web services, which would lead to deviation of semantic similarity calculation. To relax these drawbacks, we propose to *co-cluster* or *simultaneously* cluster both documents and words and pose clustering as graph partitioning problem since it does not need to take semantic similarity into account. According to some text clustering methods, following the assumption that words which typically occur together should be associated with similar concepts, words may be clustered on the basis of the documents in which they co-occur. As co-clustering is an exciting algorithm which is more informative, has less parameters and is able to effectively intertwine row and column information, it is gradually desirable to simultaneously cluster documents and words by exploiting the clear duality that word clustering induces document clustering while document clustering induces word clustering.

In this paper, we view the problem of web service clustering as a special text clustering problem and achieve an improvement of web service discovery by proposing a novel approach named WC*Cluster* to simultaneously cluster WSDL documents and the words extracted from them. Inspired by the co-clustering algorithm proposed by Dhillion [9], we regard the co-clustering problem as finding minimum cut vertex partitions in bipartite graph between WSDL documents and words. Since it is impractical to achieve a globally optimal solution to such a graph partitioning problem, we utilize the second left and right singular vectors of a suitably normalized word-document matrix to get an optimal solution.

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

1. We propose a novel approach called WC*Cluster* which can simultaneously cluster WSDL documents and words.
2. We evaluate the performance of the proposed WC*Cluster* by employing a real data set crawled from Titan [1]web service search engine.

The remainder of the paper is structured as follows: Section II reviews the related work. Some preprocessing techniques and detailed description of WC*Cluster* are introduced in Section III. Section IV describes our experiment and Section V concludes the paper.

[1]Titan: http://ccnt.zju.edu.cn:8080/

II. Related Work

Due to the rapid development of service computing, web service discovery has gradually gained much attention. There exist quite different techniques to search for the semantic Web services and non-semantic web services as the former are described by OWL-S or WSMO while the latter use WSDL documents as a description language. The research in semantic web services discovery has been relatively mature. Klusch et al. [10] presented high-level match-making techniques and utilize it for semantic web services since more structured annotations are available for service profiles. In [11], B. Benatallah et al. grounded the discovery process on a matchmaking between available web service description and a requester query and proposed a novel approach to automate web service discovery.

In this paper, we focus on the discovery of non-semantic web services. Nayak [12] proposed an approach which is based on the idea of finding search sessions that are similar to the one by user to locate a particular service and then suggest words used in those sessions, to improve web service discovery. Xin et al. [13] thought of the algorithms underlying the Woogle search engine which supports similarity search for web services, and described techniques to support the similarity search. Recent years, many efforts have been made to clustering web services for service discovery. As there are no queries to match against, the clustering of web service files differs from the traditional web service discovery problem. In [7], Liu and Wong proposed clustering WSDL documents into functionally similar homogeneous service groups is a bootstrapping step for a service engine creation and service discovery reduction. Similarly, Elgazzar et al. [8] applied text mining techniques to extract features from WSDL documents in order to cluster Web services and improved service discovery of non-semantic Web services. Chen et al. [14][15] proposed to utilize both of WSDL documents and tag information for web service mining. Wu et al. [16] presented a hybrid web service tag recommendation strategy to handle the clustering performance limitation.

Different from the one-sided clustering algorithms applied for web services discovery in previous literature, the technique used for WSDL documents clustering in this paper considers clustering WSDL documents and the words extracted from them simultaneously. There was some early work about co-clustering, Cheng et al. [17] introduced “biclustering” of both gene and conditions to knowledge discovery from expression data. Co-clustering is formally proposed for the first time by Dhillon in [9], which poses the simultaneous clustering problem as a bipartite graph partitioning problem and shows that the solution of a real relaxation to the NP-complete graph bipartitioning problem are singular vectors of an appropriately scaled word-document matrix.

In our paper, it is the first time to utilize co-clustering algorithm in the field of web services discovery. Inspired by Dhillon, we apply co-clustering to web services discovery and

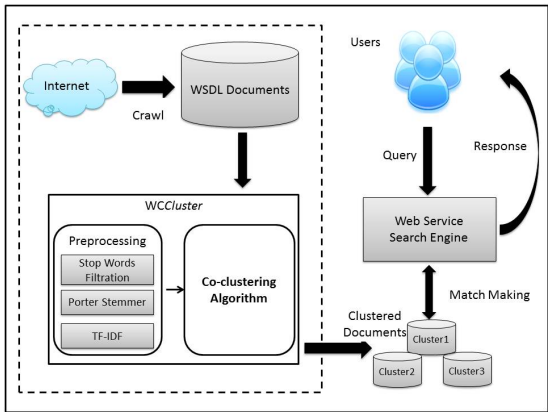

Fig. 2. Framework of Web Service Discovery

pose the services clustering problem as a WSDL documents-words bipartite graph partitioning problem.

III. WSDL Document Co-clustering

In this section, we first introduce the basic web service discovery framework in Section A. Thereafter, the preprocessing for co-clustering of web services is illustrated in Section B. And in the last section, we propose the detailed co-clustering algorithm for searching web service.

A. Web Service Discovery Framework

Inspired by the information available in web services description documents, our approach extracts features that can represent the semantic and function of the web services from WSDL documents, and then uses the co-clustering technique to simultaneously group the features and WSDL documents. Fig 2 illustrates the framework for web service discovery raised in this paper. Our method can be divided into two parts: One part is WSDL documents preprocessing including the stop words filtration, Porter Stemmer algorithm for extracting stems, and TF-idf, one kind of words weighting technique. The second part is the major component in our approach. Given the preprocessed WSDL documents, we employ our presented WC*Cluster* to co-cluster web services. After the clustered documents are generated, they can be used to match semantically the query given by users who want to obtain the desirable web services in search engine. And then the search engine returns the most relevant web services.

B. Preprocessing

1) Stop Words Filtration: After extract words from the WSDL documents, we find there are many words do not make sense, such as "be", "the", "above", "to", etc. All these words are regarded as stop words, which also called common words. Most search engines do not consider extremely common words so as to save disk space or to speed up search results. In WC*Cluster* approach, filtering stop words is to eliminate the noise of the parsed WSDL documents. There is not one definite list of stop words, we use the most prevailing list downloaded from Internet in section IV.

2) Porter Stemmer Algorithm: Except for the stop words, stems extraction is another problem should be solved. 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, but the computer will view them as two different words, and stemming algorithm can analyse the stem "creat". The original stemming algorithm was proposed in [18], then it was developed as Snowball [19]. In our experiment, we apply the Porter Stemmer algorithm of java version.

3) TF-IDF: TF-IDF is a numerical statistic that is intended to reflect the importance of a word to a document in a collection. The TF-IDF value of each word in a document increases with the frequency of it appears in the document, but is offset by the its frequency in the whole collection. The algorithm is often used as a weighting factor in text mining and other fields. We utilize TF-IDF to weight words after the word-WSDL document frequency matrix is generated, and the new weighted matrix can be used to co-clustering.

C. WCCluster

As mentioned above, WC*Cluster* is an novel approach that simultaneously cluster WSDL documents preprocessed and the features extracted from them. This section we discuss the detailed process of WC*Cluster* approach.

1) Bipartite Graph Model: Before we describe the major algorithm WC*Cluster*, it is essential to introduce our bipartite graph model for representing a WSDL document collection. An undirected bipartite graph can be denoted by a triple $G = (\mathcal{D}, \mathcal{W}, E)$ where $\mathcal{D} = \{d_1, ..., d_n\}$ and $\mathcal{W} = \{w_1, ..., w_n\}$ are two sets of vertices and E is a set of edges $\{\{d_i, w_j\} : d_i \in \mathcal{D}, w_j \in \mathcal{W}\}$. Consider our case of web service clustering, $\mathcal{D}$ is the set of WSDL documents while $\mathcal{W}$ is the set of words extracted from them. If w_j is extracted from WSDL document d_i, then edge $\{d_i, w_j\}$ exists. What should be noticed is that there are no edges between words or between WSDL documents.

An edge stands for an association between a word and a document. It is valid to capture the strength of the association through setting positive weights for the edges. Making edge-weights equal term frequencies is one possibility which is applied in our experiment. Assume the weight of edge $\{d_i, w_j\}$ is E_{ij}, then the adjacency matrix $\boldsymbol{R}$ of our bipartite graph is defined by

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

The matrix $\boldsymbol{R}$ also can be written in the form of block matrix as

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

where $\boldsymbol{A}$ is $m \times n$ word-document matrix and A_{ij} denotes the edge weight E_{ij}. Note that the vertices have been ordered so that the first m vertices index the words and the rest n index the WSDL documents.

Given a partitioning of the vertex sets mixed by $\mathcal{D}$ and $\mathcal{W}$ into two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, the cut between them plays an important role in WC*Cluster* approach. The cut 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 is 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 precondition of our WC*Cluster* algorithm is the observation of duality: Word clustering induces WSDL documents clustering while the WSDL documents clustering induces word clustering. It is easy to verify that the best word and document clustering would correspond to a partitioning of graph such that the crossing edges between subsets have minimum weight. It means the best clustering is achieved when

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

where $\mathcal{V}_1, ...\mathcal{V}_k$ is any *k*-partitioning of the bipartite graph.

2) Spectral Graph Bipartitioning: Many effective heuristic methods exist for solving the problem of graph partitioning. One of them is spectral graph partitioning. Now we concretely introduce the spectral graph partitioning in our WC*Cluster* approach. Given the undirected graph $G = (\mathcal{D}, \mathcal{W}, E)$ in which $\mathcal{D}$ and $\mathcal{W}$ have *n* vertices and *m* vertices respectively, then the *Laplacian matrix* $\boldsymbol{L} = \boldsymbol{L_G}$ of G can be defined as follows:

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

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

As presented in the section of Bipartite Graph Model, the optimal partition would be obtained by minimizing the cut value. In addition to small cut values which reflect the association between different partitions, the "balance" of clusters should be considered when solve a partitioning problem in practice. Now we provide a new objective function.

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

where $weight\left(\mathcal{V}_k\right)$ represents the weight of a vertices subset $\mathcal{V}_k$, and $weight\left(\mathcal{V}_k\right) = \sum_{i \in \mathcal{V}_k} weight\left(i\right) = \sum_{i \in \mathcal{V}_k} W_{ii}$. We assign a positive weight for each vertex i denoted by $weight\left(i\right)$, and let $\boldsymbol{L}$ be the diagonal matrix of such weights. For $\mathcal{F}\left(\mathcal{V}_1, \mathcal{V}_2\right)$, the smaller value of it indicates more balanced partitioning when two different partitions with the same cut value are given. Therefore, we can minimize the new objective function $\mathcal{F}\left(\mathcal{V}_1, \mathcal{V}_2\right)$ to get a balanced partition with a small cut value.

Suppose a graph G, and let $\boldsymbol{L}$ and $\boldsymbol{W}$ respectively be its *Laplacian* and vertex weight matrices. Assume we have a bipartitioning of $\mathcal{V}$ into $\mathcal{V}_1$ and $\mathcal{V}_2$ $(\mathcal{V} = \mathcal{D} \cup \mathcal{W})$. In order to lead to a critical theorem, let us define a generalized partition vector $\boldsymbol{q}$ with elements

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

satisfies $\boldsymbol{q}^T \boldsymbol{W} \boldsymbol{e} = 0$, and $\boldsymbol{q}^T \boldsymbol{W} \boldsymbol{q} = weight\left(\mathcal{V}\right)$, where $\mu_1 = weight\left(\mathcal{V}_1\right)$ and $\mu_2 = weight\left(\mathcal{V}_2\right)$.

Using the notation above, and referring to some properties of the *Laplacian matrix* $\boldsymbol{L}$ and theorems mentioned in [9], 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 $\boldsymbol{q}$, the Rayleigh Quotient*

$$\frac{\boldsymbol{q}^T \boldsymbol{L} \boldsymbol{q}}{\boldsymbol{q}^T \boldsymbol{W} \boldsymbol{q}} = \frac{cut\left(\mathcal{V}_1, \mathcal{V}_2\right)}{weight\left(\mathcal{V}_1\right)} + \frac{cut\left(\mathcal{V}_1, \mathcal{V}_2\right)}{weight\left(\mathcal{V}_2\right)}. \tag{4}$$

That is to say, the problem of finding the global minimum of (3) may be lead to the generalized partition vector $\boldsymbol{q}$. Since the problems is still NP-complete, it is probable to find a relaxation to the optimal generalized partition vector through the following critical theorem which is a standard conclusion from linear algebra[20].

THEOREM 2. *The problem*

$$\min_{\boldsymbol{q} \neq \boldsymbol{0}} \frac{\boldsymbol{q}^T \boldsymbol{L} \boldsymbol{q}}{\boldsymbol{q}^T \boldsymbol{W} \boldsymbol{q}}, \quad \textit{subject to} \quad \boldsymbol{q}^T \boldsymbol{W} \boldsymbol{e} = 0$$

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

$$\boldsymbol{L} \boldsymbol{x} = \lambda \boldsymbol{W} \boldsymbol{x}. \tag{5}$$

3) Association with SVD: Up to now we have not made the decision about the 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., $weight\left(i\right) = \sum_k E_{ik}$, which is a cut objective called normalized-cut. Note that the vertex weight matrix $\boldsymbol{W}$ equals the diagonal degree matrix $\boldsymbol{D}$.

According to THEOREM 2, we know that the key to get the minimum normalized cut is solve the second eigenvector of the generalized eigenvalue problem $\boldsymbol{L} \boldsymbol{x} = \lambda \boldsymbol{W} \boldsymbol{x}$. Next, we will illustrate WC*Cluster* to find WSDL documents and words clusters using our bipartite graph model. In our case,

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

which can be plugged into $\boldsymbol{L} \boldsymbol{x} = \lambda \boldsymbol{W} \boldsymbol{x}$,

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

where $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$ are diagonal degree matrices for word vertex set and WSDL document vertex set respectively. Equation (6) can be rewritten in algebraic form on basis of

the assumption that both $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$ are nonsingular. And through some algebraic calculation, we see two formulas

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

where $\boldsymbol{u} = \boldsymbol{D_1}^{1/2}\boldsymbol{a}$ and $\boldsymbol{v} = \boldsymbol{D_2}^{1/2}\boldsymbol{b}$. It is clearly that the two formulas represent the process of singular value decomposition (SVD) of the normalized matrix $\boldsymbol{A_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{A}\boldsymbol{D_2}^{-1/2}$. In detail, $\boldsymbol{u_2}$ 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 $\boldsymbol{A_n}$, For brief computation, we can replace the computation of the eigenvector of the second smallest eigenvalue of (6) of solving the left and right singular vectors corresponding to the second largest singular value of $\boldsymbol{A_n}$. Obviously, the left singular vector $\boldsymbol{u_2}$ offers us a bipartitioning of the words extracted and the right singular vector $\boldsymbol{v_2}$ offers the bipartitioning of WSDL documents.

4) WC*Cluster Algorithm:* Generally, we discuss our WC*Cluster* approach in the situation of multipartitioning since the bipartitioning is a special case where the number of word and WSDL document clusters $k = 2$. Similar with the fact that the second largest singular vectors contain bi-model 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 WSDL documents. Given these vectors, the main task is to extract the "best" partition from them.

The optimal generalized partition vector for multipartitioning problem must be k-valued. We use the classical k-means algorithm to look for the best k-modal fit to the k l-dimensional points $\boldsymbol{p}_1, ..., \boldsymbol{p}_k$. From the previous section, we can form the l-dimensional eigenvector of $\boldsymbol{L}$

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

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

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

Combined with the techniques in WSDL document preprocessing, the WC*Cluster* algorithm is described as follows.

The major computational cost of WC*Cluster* is Step 2 for generating $\boldsymbol{A}_0$ 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 words and WSDL documents. 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 Step 5 of k-means algorithm has the computational cost of $O(tk(m+n)l)$, k is the number of clusters, t denotes the iterations and l is the number of singular vectors to be computed.

Algorithm 1 Framework of WC*Cluster*.

Input:
A collection of WSDL documents.
The number of WSDL document and word clusters k.
Output:
The Clustered $\boldsymbol{X}$.

1: Preprocess WSDL documents through extracting features, filtering stop words, stems extraction.
2: Generate the $m \times n$ word-WSDL document matrix $\boldsymbol{A}_0$.
3: Get the new weighted matrix $\boldsymbol{A}$ by TF-IDF. Normalize $\boldsymbol{A}$ by $\boldsymbol{A_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{A}\boldsymbol{D_2}^{-1/2}$.
4: Compute $l = \lceil \log_2 k \rceil$ singular vectors of $\boldsymbol{A_n}$, $\boldsymbol{u}_2, \boldsymbol{u}_3, ..., \boldsymbol{u}_{l+1}$ 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 WSDL documents and words.
6: **return** $\boldsymbol{X}$ Clustered by k-means algorithm.

IV. Experiment

In this section, we apply WC*Cluster* to two different WSDL document collections and evaluate them in terms of precision and recall. In addition, the experiment learns the hidden sparsity structure of word-WSDL document matrix through WC*Cluster*.

A. Experiment Setup

Our experiments are based on the 114 WSDL documents of online web services gathered from real-world Web service search engine Titan. We make a manual classification of these WSDL documents to easily evaluate the experiment results by comparing with the other algorithm. Our data sets consist of following categories: "Business", "Weather", "Bioinformatics", "Translation", "Music" and "HR" as shown in Table I with the WSDL document URLs.

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. 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

Clustering results evaluation is a tricky business. Fortunately, the WSDL documents are already categorized so that we can compare the clusters with the true class labels. For this situation, we use two evaluate metrics, precision and recall, to evaluate the performance of WC*Cluster*. As introduced in [21], precision reflects the exactness while recall is a measure of completeness. Precision and Recall can be respectively described as:

$$\begin{aligned} Precision_{c_i} &= \frac{succ(c_i)}{succ(c_i) + mispl(c_i)} \\ Recall_{c_i} &= \frac{succ(c_i)}{succ(c_i) + missed(c_i)}, \end{aligned} \tag{9}$$

TABLE I
THE MANUALLY LABELED CATEGORIES FOR CLUSTERING VERIFICATION

Category	WSDL URL	
Bioinformatics (10)	http://staff.vbi.vt.edu/pathport/services/wsdls/beta/blastlocalgt.wsdl.	http://gbio-pbil.ibcp.fr/ws/ClustalwWS.wsdl.
	http://gbio-pbil.ibcp.fr/ws/CoilsWS.wsdl.	http://gbio-pbil.ibcp.fr/ws/DscWS.wsdl.
	http://gbio-pbil.ibcp.fr/ws/PredatorWS.wsdl.	http://gbio-pbil.ibcp.fr/ws/GorIWS.wsdl.
	http://staff.vbi.vt.edu/pathport/services/wsdls/pathport/watergt.wsdl.	http://gbio-pbil.ibcp.fr/ws/SSearchWS.wsdl.
	http://staff.vbi.vt.edu/pathport/services/wsdls/beta/water.wsdl.	http://gbio-pbil.ibcp.fr/ws/FastaWS.wsdl.
Business (30)	http://schemas.monster.com/BGW3.5/WSDL/MonsterBusinessGateway.wsdl.	http://ws.strikeiron.com/DnBBusinessProspectLinkage2?WSDL.
	http://ser02.2sms.com/WebServices/1.0/SMSService.asmx?WSDL.	http://ws.strikeiron.com/CorteraBusinessVitals?WSDL.
	http://www.feedoo.com/database/BusinessRulesWebService.asmx?WSDL.	http://www.xignite.com/xnews.asmx?WSDL.
	http://www.explore.fr/businessimmo/Webservices/WsBusinessImmo.asmx?wsdl.	http://ws.strikeiron.com/ReversePhoneBizIntel?WSDL.
	http://2sms.com/WebServices/1.4/SMSService.asmx?WSDL.	http://www.2sms.com/WebServices/1.1/SMSService.asmx?WSDL.
	http://asp.uniserv-online.de/axis/services/InternationalPostalValidation?wsdl.	http://www.xignite.com/xreleases.asmx?WSDL.
	http://www.investorsinpeople.co.uk/_vti_bin/BusinessDataCatalog.asmx?wsdl.	http://www.xignite.com/xwatchlists.asmx?WSDL.
	http://iconx.biz/IconWebServices/icontest.asmx?WSDL.	http://www.xignite.com/xinsider.asmx?WSDL.
	https://webservices.primerchants.com/CheckVerifyWS/CheckVerifyWS.asmx?wsdl.	http://portal3.telecomh2o.com/portalcustomer.asmx?WSDL.
	http://www.selectrucks.com/DealerContactService.asmx?WSDL.	http://www.kintera.com/services/Sphere.asmx?WSDL.
	http://www.greeninsurancecompany.co.uk/quoteservice/quoterequest.asmx?WSDL.	http://ws.cdyne.com/cdyne411ws/cdyne411.asmx?wsdl.
	https://www.contact-everyone.fr.orange-business.com/ContactEveryone/services/WSReceiveMO?wsdl.	http://www.xignite.com/xholdings.asmx?WSDL.
	https://www.api-contact-everyone.fr.orange-business.com/ContactEveryone/services/MultiDiffusionWS?wsdl.	https://www.insightgateways.com/natprop.cfc?wsdl.
	https://webservices.optimalpayments.com/directdebitWS/DirectDebitService/v1?wsdl.	https://208.57.4.35:8443/axis/services/Reseller?wsdl.
	https://webservices.optimalpayments.com/Teller/tp/TransactionProcessing.jws?WSDL.	http://developer.ebay.com/webservices/393/ebaySvc.wsdl.
Music (10)	http://ws.contentlib.mweb.co.th/ContentLibrary.WS/WSMusic.asmx?wsdl.	http://radio3.cbc.ca/services/MusicWebService.asmx?WSDL.
	http://www.birdwellmusic.com/BirdwellMusicServices.asmx?wsdl.	http://www.skst.com.cn/service.php?wsdl.
	http://www.flash-db.com/services/ws/flashCDDB.wsdl.	http://hunderttausend.de/shared/webservice/Amazon.asmx?WSDL.
	http://www.warwick.ac.uk/services/musiccentre/contact/society_login.php?wsdl.	http://popflycreator.com/Services/IDEService.asmx?WSDL.
	http://webservices.amazon.de/AWSECommerceService/DE/AWSECommerceService.wsdl.	http://mien.ch/music/webservice.php?wsdl.
Translation (20)	http://office2003research.translate.ru/ResearchService/TranslateSvc.asmx?wsdl.	http://intellekt.ws/translatr/translatr.asmx?wsdl.
	http://langrid.nict.go.jp/langrid-1.2/wsdl/GoogleTranslate.	http://www.webservicex.net/translateservice.asmx?WSDL.
	http://office2003research.translate.ru/ResearchService/ITranslateSvc.asmx?wsdl.	http://langrid.nict.go.jp/langrid-1.2/wsdl/YakushiteNet.
	http://langrid.nict.go.jp/langrid-1.2/wsdl/TranslationCombinedWithBilingualDictionaryWithLongestMatchSearch.	http://langrid.nict.go.jp/langrid-1.2/wsdl/AbstractTranslation.
	http://ubu.math.helsinki.fi:8080/NLG_service/services/NLGTranslator?wsdl.	http://su.nottingham.ac.uk/fugby/translate.wsdl.
	http://langrid.nict.go.jp/langrid-1.2/wsdl/TranslationCombinedWithBilingualDictionary.	http://langrid.nict.go.jp/langrid-1.2/wsdl/AbstractTranslation2.
	http://langrid.ai.soc.i.kyoto-u.ac.jp/langrid-service_1_2/services/JServer?wsdl.	http://langrid.nict.go.jp/langrid-1.2/wsdl/ThreeHopTranslationEnJa.
	http://biomoby.org/services/wsdl/genome.imim.es/translateGeneIDGFFPredictions.	http://langrid.nict.go.jp/langrid-1.2/wsdl/KyotoUCLWT.
	http://langrid.ai.soc.i.kyoto-u.ac.jp/langrid-service_1_2/services/CLWT?wsdl.	http://langrid.nict.go.jp/langrid-1.2/wsdl/JServerJTCForMembers.
	http://office2003research.translate.ru/ResearchService/SimpleTranslateSvc.asmx?wsdl.	http://www.aliparvaresh.com/webservices/onlinedic.asmx?wsdl.
Weather (20)	http://www.webservicex.net/usweather.asmx?wsdl.	http://www.webservicex.com/globalweather.asmx?WSDL.
	http://www.webservicex.net/WeatherForecast.asmx?wsdl.	http://ws.srlink.com/geoservices/globalweatherservice.asmx?wsdl.
	http://www.pathfinder-xml.com/development/WSDL/WeatherStationService.wsdl.	http://das.ae/webservices/WorldWeather.asmx?WSDL.
	http://www.pathfinder-xml.com/development/WSDL/WeatherForecastService.wsdl.	http://websky.kma.go.kr/services/WeatherChartService?wsdl.
	http://ws.strikeiron.com/MalaikaConsultants/USWeatherForecast?WSDL.	http://lostsprings.com/weather/WeatherService.asmx?WSDL.
	http://hs19.iccs.bas.bg/work/soap/Weather.php?wsdl.	http://rightactionscript.com/webservices/nusoap/server.php?wsdl.
	http://www.nws.noaa.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl.	http://api.wxbug.net/weatherservice.asmx?wsdl.
	http://bearmini.net/LivedoorWeatherHacks.asmx?WSDL.	http://www.premis.cz/PremisWS/MeteorologyWS.asmx?WSDL.
	http://www.pathfinder-xml.com/development/WSDL/MetarService.wsdl.	http://webservice.jaron.cn/weather.asmx?WSDL.
	http://www.weather-maker.com/(wugtm4ji53etem45qngnor45)/webservices/getweather.asmx?wsdl.	http://websky.kma.go.kr/services/DustStormService?wsdl.
HR (24)	http://developer.employease.com/wsdl/AddEmployeeEarnings.wsdl.	http://developer.employease.com/wsdl/AddEmployeeJobTitle.wsdl.
	http://developer.employease.com/wsdl/ChangeEmployeeCorpGroup.wsdl.	http://developer.employease.com/wsdl/AddJobTitle.wsdl.
	http://developer.employease.com/wsdl/ChangeEmployeeStatusClassification.wsdl.	http://developer.employease.com/wsdl/AddEmployeeReportsTo.wsdl.
	http://developer.employease.com/wsdl/ChangeEmployeeEarnings.wsdl.	http://developer.employease.com/wsdl/ChangeEmployeeJobTitle.wsdl.
	http://developer.employease.com/wsdl/ChangeEmployeePersonalInfo.wsdl.	http://developer.employease.com/wsdl/ChangeEmployeePayGroup.wsdl.
	http://developer.employease.com/wsdl/ChangeEmployeeReportsTo.wsdl.	http://developer.employease.com/wsdl/ChangeCorpGroup.wsdl.
	http://developer.employease.com/wsdl/ChangeEmployeeWorkInfo.wsdl.	http://developer.employease.com/wsdl/GetCorpGroup.wsdl.
	http://developer.employease.com/wsdl/GetEmployeeCorpGroup.wsdl.	http://developer.employease.com/wsdl/GetEmployeeEarnings.wsdl.
	http://developer.employease.com/wsdl/GetEmployeeJobTitle.wsdl.	http://developer.employease.com/wsdl/GetEmployeePayGroup.wsdl.
	http://developer.employease.com/wsdl/GetEmployeeReportsTo.wsdl.	http://developer.employease.com/wsdl/GetEmployeeStatus.wsdl.
	http://developer.employease.com/wsdl/GetEmployeeWorkInfo.wsdl.	http://developer.employease.com/wsdl/GetJobTitle.wsdl.
	http://developer.employease.com/wsdl/ReturnFromLeaveEmployee.wsdl.	http://developer.employease.com/wsdl/RetireEmployee.wsdl.

where c_i denotes the cluster i, $succ(c_i)$ is the number of web services successfully placed in the cluster c_i, $mispl(c_i)$ is the number of web services incorrectly clustered in 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.

TABLE II
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

C. Results and Discussion

Since the performance of clustering different number of categories has distinction. We use two collections of WSDL documents in the size of 3 and 5 categories respectively.

1) Clustering for 3 Categories: In this part, we present partitioning result on a data set named "Categories3" including the "Business", "Weather" and "Bioinformatics" categories. As we manually know the "true" category label for each WSDL document, a confusion matrix can be formed to show the performance of WC*Cluster*. In addition, the metrics, precision and recall, can be easily derived from the matrix.

Table II demonstrates the effectiveness of applying WC*Cluster* to the Categories3 data set. It can be observed in the confusion matrix that cluster $\mathcal{D}_1$ almost consists entirely of the "Business" category. And 19 of the 20 WSDL documents in $\mathcal{D}_2$ are from "Weather" category, while cluster $\mathcal{D}_3$ contains 11 WSDL documents in which there are 10 of "Bioinformatics". It is worth mentioning that all of the documents in "Bioinformatics" are clustered in one cluster. The two performance measures precision and recall for each clusters reach to more than 90%, which means WC*Cluster* works well on the collection of 3 categories.

Meanwhile, WC*Cluster* can discover the structure in the sparse word-WSDL document matrix. It is showed in Fig.3 that the original word-WSDL document matrix and the realigned matrix acquired by arranging rows and columns based on the cluster order to reveal the co-clusters. In Fig.3, WC*Cluster* reveals the underlying sparsity structure of various co-clusters containing 3 WSDL document clusters and the corresponding 3 word clusters. According to some block diagonal substructrue, it can be learned that some word clusters are highly

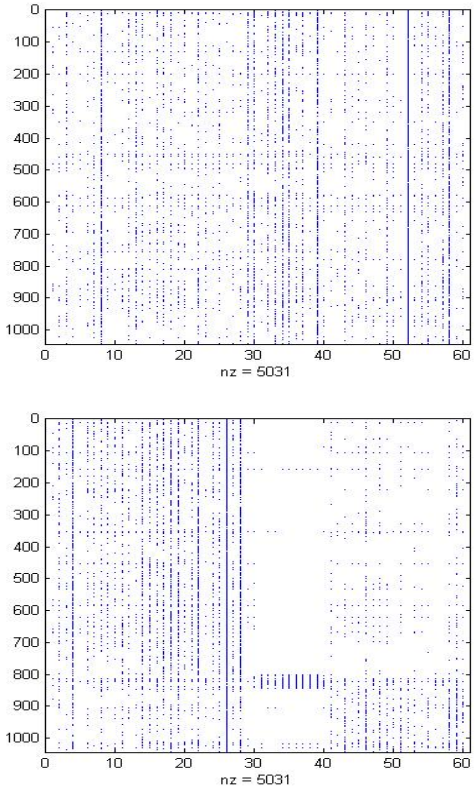

Fig. 3. Sparsity structure of word-WSDL document co-occurrence matrix before(above) and after(below) co-clustering. The shaded regions stand for the non-zero entries

indicative of individual WSDL document clusters. Whereas the dense sub-block at the bottom in the second panel shows that some clusters may have more uniform distribution over the WSDL document clusters.

2) Clustering for 5 Categories: This section we show the partitioning performance of co-clustering WSDL documents from 5 Categories: "Bioinformatics", "Translation" ,"Weather", "HR" and "Music". The number of this collection amounts to 84 and we call it "Categories5". Table III gives the multi-partitioning result obtained by using $l = \lceil \log_2 k \rceil = 3$ singular vectors. It is observed that $\mathcal{D}_1$, $\mathcal{D}_3$ and $\mathcal{D}_4$ are "purely" from the corresponding categories respectively. The web services of "Bioinformatics", "Music" , "HR" are successfully placed in clusters, as indicated by 100% recall value showed in Table IV. We notice from the confusion matrix that clusters $\mathcal{D}_2$ and $\mathcal{D}_5$ are mixed with 2 or 3 original classes, which leads to the relatively low precision values. We find that the low precision is mainly because of the mutual correlation between two categories, "Translation" and "Weather". The web services that are supposed to belong to "Translation" group are divided into 3 clusters and it may caused by the relatively average frequency of the words extracted from the WSDL documents.

K-means has been widely used in web services clustering based on the semantic similarity between WSDL documents. Table IV compares the performance of WC*Cluster* and k-means algorithm based on five manually identified groups of web services. It is evidently that WC*Cluster* has higher precision and recall values for most identified categories. For example, WC*Cluster* makes the improvement to the precision for the class "Bioinformatics" by 30.8% and to the recall by 10%. Particularly, the recall values for each cluster of WC*Cluster* are higher than that of k-means algorithm. In a word, our proposed approach is superior to the contrast algorithm on the whole.

TABLE III
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

TABLE IV
PERFORMANCE MEASURES COMPARED WITH 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

V. CONCLUSION

Clustering web services into functional similar classes has been demonstrated to be an effective way to bootstrap web service discovery. Different from the previous approaches which solely consider the semantic similarity between WSDL documents, in this paper, we propose an approach WC*Cluster* to co-cluster WSDL documents in order to bootstrap web services discovery. The novel idea of WC*Cluster* is to pose a WSDL documents collection as a bipartite graph, using which we model the co-clustering problem as a graph partitioning problem. The proposed approach solves the partitioning problem based on a spectral graph algorithm that uses left and right singular vectors of the word-WSDL document matrix to obtain a good partitioning.

In addition, WC*Cluster* works well on real data set as showed in the experimental results. As future work, we plan to utilize the tag information of web services to improve the performance of WC*Cluster*.

ACKNOWLEDGMENT

This research was partially supported by the National Technology Support Program under grant of 2011BAH16B04, the National Natural Science Foundation of China under grant of 61173176, Science and Technology Program of Zhejiang Province under grant of 2013C01073, National High- Tech Research and Development Plan of China under Grant No. 2013AA01A604, the scientific research project of Central South University under the grant No.904010001.

REFERENCES

[1] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web services architecture, w3c working group note, 11 february 2004," *World Wide Web Consortium, article available from: http://www. w3. org/TR/ws-arch*, 2004.

[2] M. Parazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[3] H. Haas and A. Brown, "Web services glossary," *W3C Working Group Note (11 February 2004)*, 2004.

[4] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Web service discovery mechanisms: Looking for a needle in a haystack," in *International Workshop on Web Engineering*, vol. 38, 2004.

[5] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 795–804.

[6] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana *et al.*, "Web services description language (wsdl) 1.1," 2001.

[7] W. Liu and W. Wong, "Web service clustering using text mining techniques," *International Journal of Agent-Oriented Software Engineering*, vol. 3, no. 1, pp. 6–26, 2009.

[8] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering wsdl documents to bootstrap the discovery of web services," in *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 147–154.

[9] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 269–274.

[10] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with owls-mx," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 2006, pp. 915–922.

[11] B. Benatallah, M.-S. Hacid, A. Leger, C. Rey, and F. Toumani, "On automating web services discovery," *The VLDB Journal*, vol. 14, no. 1, pp. 84–96, 2005.

[12] R. Nayak, "Data mining in web services discovery and monitoring," *International Journal of Web Services Research (IJWSR)*, vol. 5, no. 1, pp. 63–81, 2008.

[13] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 372–383.

[14] L. Chen, L. Hu, Z. Zheng, J. Wu, J. Yin, Y. Li, and S. Deng, "Wt-cluster: Utilizing tags for web services clustering," in *Service-Oriented Computing*. Springer, 2011, pp. 204–218.

[15] L. Chen, J. Wu, Z. Zheng, M. R. Lyu, and Z. Wu, "Modeling and exploiting tag relevance for web service mining," *Knowledge and information systems*, vol. 39, no. 1, pp. 153–173, 2014.

[16] J. Wu, L. Chen, Z. Zheng, M. R. Lyu, and Z. Wu, "Clustering web services to facilitate service discovery," *Knowledge and information systems*, vol. 38, no. 1, pp. 207–229, 2014.

[17] Y. Cheng and G. M. Church, "Biclustering of expression data." in *Ismb*, vol. 8, 2000, pp. 93–103.

[18] M. F. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.

[19] M. Porter, "Snowball: A language for stemming algorithms," 2001.

[20] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[21] J. Makhoul, F. Kubala, R. Schwartz, R. Weischedel *et al.*, "Performance measures for information extraction," in *Proceedings of DARPA broadcast news workshop*, 1999, pp. 249–252.