Sparse Functional Representation for Large-Scale Service Clustering

Qi Yu

College of Computing and Information Science,
Rochester Institute of Technology
qi.yu@rit.edu

Abstract. Service clustering provides an effective means to discover hidden service communities that group services with relevant functionalities. However, the ever increasing number of Web services poses key challenges for building large-scale service communities. In this paper, we address the scalability issue in service clustering, aiming to discover service communities over very large-scale services. A key observation is that service descriptions are usually represented by long but very sparse term vectors as each service is only described by a limited number of terms. This inspires us to seek a new service representation that is economical to store, efficient to process, and intuitive to interpret. This new representation enables service clustering to scale to massive number of services. More specifically, a set of anchor services are identified that allow to represent each service as a linear combination of a small number of anchor services. In this way, the large number of services are encoded with a much more compact anchor service space. We conduct extensive experiments on real-world service data to assess both the effectiveness and efficiency of the proposed approach. Results on a dataset with over 3,700 Web services clearly demonstrate the good scalability of sparse functional representation.

1 Introduction

Service oriented computing holds tremendous promise by exploiting Web services as an efficient vehicle to deliver and access various functionalities over the Web. The past few years have witnessed a fast boost of Web services due to the wide adoption of service-oriented computing in both industry and government. The proliferating services have formed a functionality-centric repository, through which key computing resources can be conveniently accessed via the standard Web service interface. However, the ever increasing number of Web service poses key challenges to discover services with user required functionalities. A rigorous and systematic methodology is in demand for efficiently and accurately searching user desired services from a large and diverse service repository.

Universal Description Discovery and Integration (UDDI) provides a standard registry service to publish and discover Web services. To make a service searchable, the service provider needs to first publish its service in the UDDI registry. Nonetheless, as service providers are autonomous in nature, it is infeasible to enforce them to publish their services in the registry. In fact, most service vendors

C. Liu et al. (Eds.): ICSOC 2012, LNCS 7636, pp. 468–483, 2012.

choose to directly advertise their services via their own web sites. Furthermore, when change occurs to a published service, the service entry in the UDDI may need to be updated to ensure consistency. This gives rise to additional maintenance cost for service providers. Recent statistics show that more than 50% services in the public UDDI registries are invalid.

Service search engines have gained increasing popularity by automatically collecting service descriptions using crawlers. Service descriptions are then indexed and matched against user's searching keywords. One key impediment towards the wide adoption of service search engines has been the poor search quality resulted from simple keyword matching. While keyword matching may perform reasonably well on regular Web pages, it suffers from service descriptions, which are usually generated from application programs using Web service deployment tools. Many service descriptions are comprised of very limited number of terms, most of which are not proper words. Therefore, there is a low chance for a service description to match a searching keyword even though the service may provide the exact user-desired functionality.

Clustering techniques have been adopted to improve the quality of service discovery [5,16,3]. Service clustering computes the similarity among services to group together relevant services into homogeneous service communities. Clustering enables services to be discovered by exploiting the proximity to other services. Consider two similar services, S_1 and S_2, where S_1 contains the searching keyword while S_2 does not. Through service clustering, both S_1 and S_2 will be returned as they are deemed to provide similar functionality desired by the user. In this way, the search quality can be dramatically improved. Furthermore, service discovery can be directed to only relevant service communities so that more efficient performance is achieved.

As the number of services keeps increasing, building service communities over large-scale Web services arises as a central challenge. Following traditional document clustering, each service description $\mathbf{s}_i$ is denoted by a term vector, in which $\mathbf{s}_i(j)$ is set to the normalized frequency (or other metrics such as TF/IDF) of $\mathbf{t}_j$ if $\mathbf{t}_j \in \mathbf{s}_i$ and 0 otherwise. The length of $\mathbf{s}_i$ is equal to the size of the term dictionary, which consists of the distinct terms over all service descriptions. Most service descriptions are generated from program source codes, where various naming conventions may be used by different developers. This results in a large number of distinct terms especially when scaling to a massive number of services. For example, in one of the real service dataset used in our experiments [18], we extract around 17,000 distinct terms from over 3,700 service descriptions. However, each service description only consists of 20 distinct terms on average. Therefore, the term vector $\mathbf{s}_i$ will be very large and extremely sparse (density is around 0.1% in our dataset) when dealing with large-scale services.

Simple clustering algorithms, such as K-means, scales well with the number of services. The similarity between two term vectors is usually computed based on the number of terms that co-occur in these two vectors. However, directly applying these algorithms to large-scale service clustering usually leads to poor clustering quality because the term vectors for service descriptions are extremely sparse and hence less likely to share common terms. Advanced algorithms, such as matrix factorization based ones (e.g., SVD co-clustering [17] and NMTF [4]), have been demonstrated to be more effective in dealing with limitations of service descriptions and generate high-quality service communities. However, it remains unclear how these algorithms can handle extremely large and sparse term vectors. In addition, the high computational cost also prohibits them from scaling to a massive number of services.

In this paper, we address the scalability issue in service clustering, aiming to discover service communities over very large-scale services. The central idea is that instead of using a large and highly diverse dictionary of terms, we seek a much more succinct representation of service descriptions. Inspired by recent works on sparse coding [8,19], we devise a novel strategy to learn a set of "anchor" services, which form a new dictionary to encode the service descriptions. This allows each service to be represented as a linear combination of a small number of anchor services. In general, the number of anchor services is smaller than the number of distinct terms with several orders of magnitude. Hence, the large number of services are encoded with a much more compact dictionary of anchor services. The new representation is essentially a projection onto the anchor service space. Similarity between services is determined based on how they are related to a small number of anchor services. Simple clustering algorithms, like K-means, can then be applied to this compact representation to efficiently and accurately cluster large-scale services. We demonstrate the effectiveness of the proposed algorithm via extensive experiments on two real-world service datasets.

The remainder of the paper is organized as follows. We discuss some related works in Section 2, which provide a background overview of the proposed approach. We present the details of sparse functional representation in Section 3. We use a concrete example to explain how sparse functional representation works and provide intuitive justifications of its effectiveness. We also propose a novel clustering scheme that integrates information from both the anchor service space and the term vector space. We apply sparse functional representation to two real-world service datasets and assess its effectiveness in Section 4. We conclude the paper and provide some future directions in Section 5.

2 Related Work

Service clustering and related technologies have been increasingly adopted to facilitate service discovery [5] or other key tasks in service computing, such as service composition [10] and service ontology construction [13].

Clustering has been a central technique to improve the accuracy of service search engines. Woogle, a Web service search engine, performs term clustering to generate a set of high-level concepts, which are then used to facilitate the matching between users' queries and the service operations [5]. Similarly, term clustering is also used in [10] to facilitate service discovery and composition. Basic K-means algorithms are usually used and the similarity between terms are evaluated based on their co-occurrence in the service descriptions. Quality

Threshold (QT) clustering is employed to cluster Web services in [6] to bootstrap the discovery of Web services. WSDL descriptions are carefully parsed and important components are extracted, which include content, types, messages, ports, and name of the Web service. Weights are assigned to each component when similarity between two services is evaluated. More complicated algorithms, such as Probabilistic Latent Semantic Analysis (PLSA), have also been applied to service clustering and discovery [12]. A SVD based algorithm is adopted to achieve the co-clustering of services and operations in [17]. Co-clustering exploits the duality relationship between services and operations to achieve better clustering quality than one-side clustering.

Given the limitations of the WSDL service descriptions, some recent proposals seek to explore external information sources, such as Wordnet [1] and Google, to improve service clustering and discovery [9,1]. In [16], matrix factorization and the semantic extensions of service descriptions have been integrated for service community discovery. The integration has the effect of placing the extended semantics into the context of the service, which more effectively leverages the extended semantics to benefit community discovery. As Web services usually consist of both WSDL and free text descriptors, novel approaches have been developed in [13] to integrate both types of descriptors for effective bootstrapping of service ontologies. Another important piece of information that is complementary to service descriptions is the service tags that users use to annotate services. A novel approach, referred to as WTCluster, is developed in [3] that exploits both WSDL documents and service tags for Web service clustering.

3 Sparse Functional Representation for Service Clustering

Sparse functional representation aims to seek a compact dictionary of anchor services to succinctly represent large-scale services. Consider a set of services $\mathcal{S} = \{\mathbf{s}_1, ..., \mathbf{s}_m\}$, where each $\mathbf{s}_i \in \mathbb{R}^n$ is denoted as a term vector and n is the size of the term dictionary that is comprised of all distinct terms extracted from the services in $\mathcal{S}$. By mapping terms into rows and services into columns, $\mathcal{S}$ is conveniently represented by a two dimensional service matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$. Each entry $\mathbf{X}_{ij} \in \mathbf{X}$ is set to the normalized frequency of term $\mathbf{t}_i$ in service $\mathbf{s}_j$ and zero if $\mathbf{t}_i \notin \mathbf{s}_j$. Table 1 provides a quick reference to a set of symbols that are commonly used in the paper.

As discussed in Section 1, due to the diverse naming conventions used in service descriptions, the size of the term dictionary increases dramatically with the number of services in $\mathcal{S}$. This will result in a huge and extremely sparse matrix $\mathbf{X}$. For example, in our experiments, a $16,884 \times 3,738$ matrix is constructed from a real-world service dataset with $3,738$ services. $\mathbf{X}$ consists of approximately 6.3×10^8 entries, among which only 0.1% are nonzero, implying a 99.9% sparsity ratio. This poses a set of key challenges for clustering large-scale services. First,

[1] `http://wordnet.princeton.edu/`

Table 1. Symbols and Descriptions

Symbol	Description
$\mathcal{S}$	set of services
$\mathbf{s}_j, \mathbf{t}_i$	the j^{th} service and i^{th} term
$\mathbf{X}, \mathbf{A}, \mathbf{W}, \mathbf{Z}$	matrices
$\mathbf{X}'$	the transpose of matrix $\mathbf{X}$
$\mathbf{X}_{ij}$	the element at the i^{th} row and j^{th} column of matrix $\mathbf{X}$
$\mathbf{x}_j$	the j-th column vector of matrix $\mathbf{X}$
$\mathbf{x}_j(i)$	the i-th element of $\mathbf{x}_j$
$\mathbf{x}_i^T$	the i-th row vector of matrix $\mathbf{X}$

scalability arises as a significant challenge for storing and processing a large service matrix whose size grows quickly with the number of services. Second, the highly sparse term vectors are a key impediment for applying many clustering algorithms to generate high-quality clusters as sparse vectors are less likely to share common terms.

3.1 Sparse Functional Representation

The above observation implies that a large and diverse term dictionary does not provide a suitable representation for large-scale service clustering. Instead, concepts with coarser granularity may be more instrumental to produce a compact and cohesive service representation. Hence, we aim to seek a new service representation that is economical to store, efficient to process, and intuitive to interpret. This new representation will enable service clustering to scale to massive number of services. Inspired by recent advances in sparse coding [8,19], we devise a novel sparse functional representation (SFR) strategy to discover a set of so called "anchor services". The anchor services are expected to capture the high-level functionalities of services while significantly compressing the original term vector space. More specifically, SFR seeks a matrix $\mathbf{A} = \{\mathbf{a_1}, ..., \mathbf{a_k}\}$, where each $\mathbf{a}_i \in \mathbb{R}^n$ denotes an anchor service and is a linear combination of a set of term vectors (or columns of $\mathbf{X}$):

$$\mathbf{A} = \{\mathbf{a}_1, ..., \mathbf{a}_k\} = \mathbf{X}\mathbf{W} \tag{1}$$

$$\mathbf{a}_i = \mathbf{X}\mathbf{w}_i = \sum_{j=1}^{m} \mathbf{W}_{ij}\mathbf{x}_j, \forall i = 1, ..., k \tag{2}$$

where $\mathbf{W} = \{\mathbf{w}_1, ..., \mathbf{w}_k\} \in \mathbb{R}^{m \times k}$ is a weight matrix. Each entry $\mathbf{W}_{ij}$ denotes how much service $\mathbf{s}_j$ contributes to anchor service $\mathbf{a}_i$. It is worth to note that $\mathbf{W}_{ij}$ may take a negative value, meaning that $\mathbf{s}_j$ related information is removed from $\mathbf{a}_i$. Hence, a negative entry in $\mathbf{W}$ serves as the "de-noise" purpose to generate an anchor service with purer functionality or concept.

In practice, we have $k \ll n$. Hence, the anchor services provide a compact way to represent large-scale services. The desired anchor services are expected to capture high-level concepts or functionalities of the service space. Thus, we should be able to recover the original services by using the anchor services. Meanwhile, as most services are designed with specific purposes, it is common for a single service to provide focused and limited functionalities. In another word, a service is expected to be only related to a small subset of anchor services that cover its functionalities. Therefore, a desired anchor service set should optimize the following objection function:

$$\min_{\mathbf{A}, z_i \geq 0} J_0 = \sum_{i=1}^{m} \|\mathbf{x}_i - \mathbf{A}\mathbf{z}_i\|^2 + \lambda ||\mathbf{z}_i||_0 \tag{3}$$
$$\text{subject to } ||\mathbf{a}_j||^2 \leq c, \forall j = 1, ..., k$$

where $\mathbf{z}_i \in \mathbb{R}_+^k$ is the coefficient vector with $\mathbf{z}_i(j)$ signifying the correlation between $\mathbf{x}_i$ and anchor service $\mathbf{a}_j$. $||\mathbf{z}_i||_0$ is the L_0 norm of $\mathbf{z}_i$ that counts the number of nonzero elements in $\mathbf{s}_i$. Since each service is expected to correlate with only a small subset of anchor services, $\mathbf{z}_i$ with many nonzero elements will be penalized and λ is the penalty parameter. Therefore, the second term of Eq. (3) corresponds to a sparsity constraint on $\mathbf{z}_i$. The norm constraint on the size of the anchor service, i.e., $||\mathbf{a}_j||^2 \leq c$, avoids arbitrarily large anchor services that keep $\mathbf{A}\mathbf{z}_i$ unchanged while making $\mathbf{z}_i$ arbitrarily close to zero.

It is worth to note that $\mathbf{z}_i$ is non-negative, which allows more intuitive interpretation of the proposed sparse functional representation. More specifically, the functionality of each service is represented as an additive combination of functionalities encoded by a small number of anchor services.

3.2 Relaxation of the Objective Function

It has been proved that finding $\mathbf{A}$ and $\mathbf{s}_i$ that optimize objective function in Eq. (3) is NP-hard [19]. Therefore, instead of directly solving Eq. (3), we tackle the following optimization problem with a relaxed constraint:

$$\min_{\mathbf{W}, \mathbf{S} \geq 0} J_1 = \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{Z}\|_F^2 + \lambda \sum_{i=1}^{m} ||\mathbf{z}_i||_1 \tag{4}$$
$$\text{subject to } ||\mathbf{X}\mathbf{w}_j||^2 \leq c, \forall j = 1, ..., k$$

where $||\mathbf{Y}||_F = \sum_{ij} \sqrt{\mathbf{Y}_{ij}}$ stands for Frobenius norm; $\mathbf{z}_i$ is the i-th column of $\mathbf{Z} \in \mathbb{R}^{k \times m}$; $||\mathbf{z}_i||_1 = \sum_{j=1}^{k} |\mathbf{Z}_{ji}|$ is the L_1 norm of $\mathbf{z}_i$. We replace $\mathbf{A}$ and $\mathbf{a}_j$ by $\mathbf{X}\mathbf{W}$ and $\mathbf{X}\mathbf{w}_j$, respectively, due to Equations (1) and (2).

The first term of J_1 is equivalent to the first term of J_0 reformulated in the matrix form. The key difference between J_0 and J_1 is the change from the L_0 norm of $\mathbf{z}_i$ to the L_1 norm. The relaxed optimization is in essence to minimize a quadratic function with a L_1 norm constraint on $\mathbf{z}_i$ and a L_2 norm constraint on $\mathbf{a}_j$. The optimization problem in the form of Eq.(4) is commonly known as *basis pursuit*, which has been demonstrated to be effective in finding sparse coefficient vectors (i.e., $\mathbf{z}_i$'s). Therefore, the solution of J_1 is expected to provide a good approximation to the optimal solution of J_0. Furthermore, the relaxed optimization problem is computational attractive, which can be efficiently tackled by iteratively solving a L_1 regularized least squares problem and L_2 regularized least square problem to obtain $\mathbf{Z}$ and $\mathbf{W}$, respectively [8].

3.3 An Illustrating Example

In what follows, we use a simple example to further illustrate the key ideas of SFR as presented above. We randomly choose six services from a real-world service dataset, in which three services are from the travel domain and the other three are from the medical domain. Processing the service descriptions results in 12 distinct terms. Hence, a 12×6 service matrix $\mathbf{X}$ is constructed. The transpose of $\mathbf{X}$ is given in Eq. (5) for the convenience of presentation, in which each row denotes a service and each column denotes a distinct term. Each entry corresponds to a term frequency and each row vector is further normalized to have L_2 norm equal to 1.

$$\mathbf{X}' = \begin{pmatrix} 0.29 & 0.29 & 0.86 & 0.29 & 0.096 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.26 & 0.26 & 0.61 & 0.26 & 0.088 & 0.61 & 0.18 & 0 & 0 & 0 & 0 & 0 \\ 0.33 & 0.33 & 0.33 & 0.33 & 0.11 & 0 & 0 & 0.33 & 0.66 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.092 & 0 & 0 & 0 & 0 & 0.55 & 0.83 & 0 \\ 0 & 0 & 0 & 0 & 0.092 & 0 & 0 & 0 & 0 & 0.55 & 0 & 0.83 \\ 0 & 0 & 0 & 0 & 0.092 & 0 & 0 & 0 & 0 & 0.55 & 0 & 0.83 \end{pmatrix} \tag{5}$$

It is clear from Eq. (5) that even for a small service set with just six services, the service matrix $\mathbf{X}$ is already very sparse. There are around 60% (42 out of 72) zero entries. The sparsity ratio will increase dramatically when the set scales to a large number of services.

We set the number of anchor services as 4 (i.e., $k = 4$) and solve the relaxed optimization problem in Eq. (4), which leads to:

$$\mathbf{W} = \begin{pmatrix} -0.12 & 0.4 & 0.37 & 0.14 \\ -0.13 & 0.39 & 0.35 & 0.15 \\ -0.14 & 0.31 & 0.34 & 0.17 \\ 0.55 & -0.17 & 0.26 & 0.098 \\ 0.32 & -0.12 & -0.19 & 0.44 \\ 0.3 & -0.12 & -0.2 & 0.44 \end{pmatrix}, \quad \mathbf{Z} = \begin{pmatrix} 0 & 0 & 0 & 0.24 & 0.22 & 0.21 \\ 0.23 & 0.23 & 0.2 & 0 & 0 & 0 \\ 0.27 & 0.26 & 0.24 & 0.13 & 0.0087 & 0.0054 \\ 0.26 & 0.26 & 0.26 & 0.26 & 0.42 & 0.42 \end{pmatrix} \tag{6}$$

The i-th column of $\mathbf{Z}$ (i.e., $\mathbf{z}_i$) corresponds to the new representation of the i-th service (i.e., i-th row of $\mathbf{X}'$ in Eq. (5)) in the anchor service space. As expected, $\mathbf{Z}$ has a sparse structure, which justifies the effectiveness of L_1 norm approximation of the original optimization problem. The first three columns of $\mathbf{Z}$ imply that the first three services are only relevant to the last three anchor services. In contrast, the other three services, which correspond to the last three columns of $\mathbf{Z}$, are tightly coupled with the first and last anchor services (the third entries of these three columns are close to zero). All entries in $\mathbf{Z}$ are non-negative, which allows functionality of each service to be represented as an additive combination of functionalities encoded by a small number of relevant anchor services.

Some interesting observations are also revealed from the weight matrix $\mathbf{W}$. These observations demonstrate that the interplay between the weight matrix $\mathbf{W}$ and the coefficient matrix $\mathbf{Z}$ helps achieve the effectiveness of SFR. For example, the first three entries of $\mathbf{w}_1$ (i.e., the first column of $\mathbf{W}$) take negative values. These negative entries imply that information related to the first three services are removed from the first anchor service $\mathbf{a}_1$. Therefore, a negative entry $\mathbf{W}_{ij}$ has the effect of "decoupling" the i-th service from j-th anchor service. In contrast, a positive entry $\mathbf{W}_{ij}$ signifies the "addition" of the i-th service's functionality to the j-th anchor service. The decoupling and addition mechanism helps discover cohesive concepts or service functionalities that are captured by the anchor services. It also leads to unambiguous service-to-anchor service relationships. For example, $\mathbf{z}_1^T$, the first row of the coefficient matrix $\mathbf{Z}$, consists of three zero and three nonzero entries. This implies that the first three services are completely irrelevant to anchor service $\mathbf{a}_1$ whereas the last three are tightly coupled with $\mathbf{a}_1$. In fact, the three zero entries are resulted from the first three entries in $\mathbf{w}_1$, which decouple the first three services from $\mathbf{a}_1$. The three nonzero entries are due to the last three entries of $\mathbf{w}_1$ that add the functionalities of the last three service into $\mathbf{a}_1$. Similarly, the second row of $\mathbf{Z}$ shows that the first three services are relevant to anchor service $\mathbf{a}_2$ while the last three service are irrelevant to it.

3.4 Clustering in Anchor Service Space

By optimizing objection function J_0 or its relaxed version J_1, we aim to find an anchor service set $\mathbf{A}$ and a new sparse representation $\mathbf{z}_i$ to best approximate $\mathbf{x}_i$:

$$\mathbf{x}_i \approx \mathbf{A}\mathbf{z}_i = \sum_{j=1}^{k} \mathbf{a}_j \mathbf{Z}_{ji} \tag{7}$$

Therefore, the new representation $\mathbf{z}_i$ can be regarded as the projection of $\mathbf{x}_i$ onto the anchor service space $\mathbf{A} = \{\mathbf{a}_1, ..., \mathbf{a}_k\}$. The coefficient vector $\mathbf{z}_i$ captures the relevance between the i-th service and all the k anchor services. The sparsity constraint on $\mathbf{z}_i$ leads to clear-cut relationships between a service and anchor services. Hence, services can be easily separated based on their distinct relationships with the anchor services using sparse functional representation.

Figure 1 provides a schematic view of service clustering in the anchor service space. Since each service is only related to a small subset of anchor services, the similarity between two services can be easily computed based on how they are related to the anchor services. More specifically, two services are similar if they are related to a similar set of anchor services. Therefore, the anchor services serve as a bridge to relate different services. Since the anchor services capture the high-level concepts of the services, projection onto the anchor service space provides a better way to assess the similarity between services than using terms. The sparsity constraint on the coefficient vectors provides a clear separation between services, which significantly facilitates service clustering. Any simple clustering

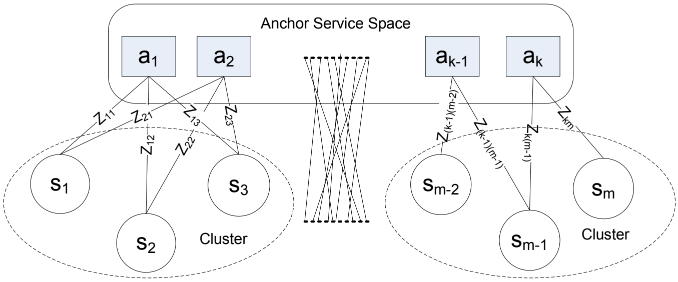

Fig. 1. Clustering in Anchor Service Space

algorithms, such as K-means, may be directly applied to the coefficient matrix $\mathbf{Z}$ to generate service clusters. Meanwhile, the anchor service space has a much low dimensionality than the term vector space (i.e., $k \ll n$), clustering in the anchor service space can easily scale to a massive number of services.

3.5 Term Vector and Anchor Service Integration

Sparse functional representation is formed by projecting term vectors $\mathbf{x}_i$'s onto an anchor service space. Service clustering is then performed on the projected representation, which is independent on the original term vector space. In this section, we present a new clustering scheme that integrates information from both the anchor service space and the term vector space.

One piece of information in the term vector space that can be leveraged is the term vectors that share a decent number of distinct terms. If two term vectors have a reasonable number of common terms, it means that they share similar high-level concept and hence should be clustered together. This useful information can be incorporated into anchor space clustering in a semi-supervisory manner to improve the overall clustering quality. More specifically, we construct a neighborhood graph G, in which each vertex corresponds to a term vector $\mathbf{x}_i \in \mathbf{X}$. Two vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected in G if the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ is no less than a threshold value. The similarity can be simply computed by using cosine similarity. Assume that $\mathbf{B}$ is the incidence matrix of G. Therefore, $\mathbf{B}_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected in G (i.e., similar to each other) and 0 otherwise. We expect $\mathbf{B}$ to be sparse as it is less likely for most term vectors to share many common terms.

Consider any pair of term vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ and their corresponding sparse functional representations $\mathbf{z}_i$ and $\mathbf{z}_j$. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, we expect $\mathbf{z}_i$ and $\mathbf{z}_j$ to be similar as well. In contrast, if $\mathbf{z}_i$ and $\mathbf{z}_j$ significantly deviate from each other, $||\mathbf{z}_i - \mathbf{z}_j||^2\mathbf{B}_{ij}$ will be large as $\mathbf{B}_{ij} = 1$ when $\mathbf{x}_i$ and $\mathbf{x}_j$ are close in the term vector space. Therefore, we can use $||\mathbf{z}_i - \mathbf{z}_j||^2\mathbf{B}_{ij}$ as a penalty term and incorporate it into objection function J_1, which leads to

$$\min_{\mathbf{W},\mathbf{S}\geq 0} J_2 = \|\mathbf{X} - \mathbf{XWZ}\|_F^2 + \lambda \sum_{i=1}^{m} ||\mathbf{z}_i||_1 + \gamma \sum_{i=1}^{m}\sum_{j=1}^{m} ||\mathbf{z}_i - \mathbf{z}_j||^2 \mathbf{B}_{ij} \tag{8}$$

subject to $||\mathbf{X}\mathbf{w}_j||^2 \leq c, \forall j = 1, ..., k$

where γ is the penalty parameter. It is worth to note that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are not evaluated to be similar, we have $\mathbf{B}_{ij} = 0$. In this case, the term $||\mathbf{z}_i - \mathbf{z}_j||^2\mathbf{B}_{ij}$ is set to 0 and will not affect the objective function J_2.

4 Experimental Study

We apply the proposed SFR to two real-world service datasets. To evaluate its effectiveness in service clustering, we will compare our approach with a set competitive service clustering algorithms. As both clustering quality and efficiency are important evaluation metrics, we will report both clustering accuracy and CPU times in our experimental results.

4.1 Service Dataset Description

We include two real-world service datasets: one middle scale dataset with 452 services [7] and one large-scale dataset with 3,738 services [18]. We describe the properties of each dataset in what follows:

– **Dataset_1**: The first service dataset consists of 452 WSDL descriptions of services from 7 different application domains. More specifically, the services are distributed as follows: communication (42), education (139), economy (83), food (23), medical (45), travel (90), and weapon (30). The domain information provides labels of the service clusters, which will be used to evaluate the accuracy of the clustering algorithms in our experiments.
– **Dataset_2**: The second service dataset consists of WSDL descriptions of 3,738 services located in more than 20 countries. The services are more diverse and complicated, coming from a large number of domains varying from government to academia and industry. Unlike **Dataset_1**, no cluster labels are available in this dataset.

4.2 Metrics for Clustering Quality

For **Dataset_1**, the service domains will serve as the ground truth to evaluate the clustering quality. More specifically, we adopt two metrics to measure the service clustering quality: ACcuracy (i.e., AC) and Mutual Information (i.e., MI). Both AC and MI are widely used metrics to assess the performance of clustering algorithms [15,2]. For **Dataset_2**, since no true service cluster labels are available, we cannot use the above two metrics to evaluate clustering quality. Instead, we choose to use the Silhouette Value (i.e., SV), which is a commonly used metric for clustering quality evaluation when no ground truth is available.

– **Accuracy:** For a given service $\mathbf{s}_i$, assume that its cluster label is c_i and its domain label is d_i based on the domain information. The AC metric is defined as follows:

$$AC = \frac{\sum_{i=1}^{m} \delta(d_i, map(c_i))}{m} \tag{9}$$

where m is the total number of Web services in the service dataset. $\delta(x, y)$ is the delta function that equals to one if $x = y$ and equals to zero if otherwise. $map(c_i)$ is the permutation mapping function that maps each assigned cluster label to the equivalent domain label. The best mapping between the two sets of labels is achieved by the Kuhn-Munkres algorithm [11].

- **Mutual Information:** Let $\mathcal{D}$ be the set of application domains obtained from the service dataset and $\mathcal{C}$ be the service clusters obtained a service clustering algorithm. The mutual information metric $MI(\mathcal{D}, \mathcal{C})$ is defined as follows:

$$MI(\mathcal{D}, \mathcal{C}) = \sum_{d_i \in \mathcal{D}, c_j \in \mathcal{C}} p(d_i, c_j) \log_2 \frac{p(d_i, c_j)}{p(d_i)p(c_j)} \tag{10}$$

  where $p(d_i)$ and $p(c_j)$ are the probabilities that a randomly selected service from the service set belongs to domain d_i and cluster c_j, respectively. $p(d_i, c_j)$ is the joint probability that the randomly selected service belongs to both domain d_i and cluster c_j.
- **Silhouette Value:** The silhouette value for service $\mathbf{s}_i$ measures how similar that $\mathbf{s}_i$ is to the services in its own cluster compared to services in other clusters, and ranges from -1 to +1. More specifically, SV is defined as the average over the silhouette values of all services:

$$SV = \frac{\sum_{i=1}^{m} SV_i}{m} \tag{11}$$

$$SV_i = \frac{(b_i - a_i)}{\max(a_i, b_i)} \tag{12}$$

  where SV_i is the silhouette value for the i-th service $\mathbf{s}_i$; a_i is the average distance from $\mathbf{s}_i$ to the other services in the same cluster as $\mathbf{s}_i$, and b_i is the minimum average distance from $\mathbf{s}_i$ to services in a different cluster, minimized over clusters. Therefore, SV essentially measures the "cohesiveness" of the clusters.

4.3 Clustering on Dataset_1

Before running any service clustering algorithms, we need to preprocess the service descriptions in **Dataset_1**. We apply a standard text processing procedure that includes tokenization, stopword removal, and stemming to extract distinct terms from the service descriptions. As a result, 803 distinct terms are extracted. Thus, a 803×452 service matrix $\mathbf{X}$ is constructed. We compare the proposed SFR based clustering with the following service clustering algorithms:

- **NMTF**: Non-negative matrix tri-factorization based approach to simultaneously cluster services and operations offered by the services [4].
- **NMTFS**: Extending service descriptions by including semantically similar terms to address the sparsity issue and then applying NMTF to the extended service descriptions [16].

Table 2. Clustering Results on **Dataset_1**

Algorithms	Quality			Performance
	Accuracy (%)	Mutual Information (%)	Silhouette Value	CPU time (s)
NMTF	51.1	48.0	0.28	17.9
NMTFS	56.6	46.5	0.28	19.0
KmeanS	42.7	21.2	-0.1	33.0
SVDC	45.3	36.3	0.22	**0.9**
SFR	**60.4**	**56.6**	**0.7**	19.1

– **KmeanS**: Applying K-means clustering to the semantically extended service descriptions.
– **SVDC**: Applying Singular Value Decomposition (SVD) to co-cluster services and operations they offers [17].

We set the number of anchor services as 30, i.e., $k = 30$. The two penalty parameters λ and γ in objection function J_2 are set to 1 and 0.1, respectively. These will be used as default parameter values in our experiments unless specified otherwise. It is worth to note that a wide range of values work reasonably well for these parameters. We will investigate the impact of different parameters in Section 4.5.

Table 2 reports both clustering quality and CPU times from all the algorithms under comparison. The clustering quality is evaluated using all the three evaluation metrics described in Section 4.2. SFR clearly outperforms all other competitors in terms of clustering quality. It achieves 60.4% in clustering accuracy, which is 7% better than the second highest accuracy achieved by NMTFS. In terms of mutual information, it is 17.9% better than second best, NMTF. The results on silhouette value are pretty much consistent with those on accuracy and mutual information. SFR achieves a silhouette value at 0.7, which is much higher than all other algorithms. This demonstrates that sparse functional representation provides good separation between similar services and dissimilar ones, which makes service clustering much easier. A higher silhouette value signifies that the generated clusters are more cohesive.

In terms of performance, the CPU time used by SFR is similar to other matrix factorization based clustering algorithms, including NMTF and NMTFS. This is reasonable for a middle scale service dataset especially when the number of distinct terms are relatively small. It is worth to note that the time for SFR includes both finding the anchor service set and performing clustering in the anchor service space. In fact, most time is spent on former as clustering in SFR is just applying K-means to a compact sparse functional representation of the service space. SVDC achieves a very fast response time, which is only 0.9 second. This is because it computes a service-operation correlation matrix in order to perform co-clustering. Since the number of operations is much less than the number of terms, SVDC actually works on a much smaller matrix, which justifies its fast performance. Nonetheless, the poor clustering quality of SVDC implies that the service-operation correlation matrix does not provides a good representation for service clustering.

Table 3. Clustering Results on **Dataset_2**

Algorithms	Quality	CPU time (s)	
	Silhouette Value	Construction	Clustering
NMTF	0.05	-	359.9
PKmeans	0.31	-	3.97
Kmeans	0.21	-	3.23
SVDC	NaN	-	1131.3
SFR	**0.38**	367.3	2.7

4.4 Clustering on Dataset_2

We adopt the same standard text processing procedure to process the 3,738 service descriptions in **Dataset_2**, which results in 16,884 distinct terms. Therefore, a $16,884 \times 3,738$ service matrix $\mathbf{X}$ is constructed. Each term vector has a dimensionality of 16,884. Before applying any clustering algorithms on such high dimensional data, a common practice is to first reduce the dimensionality. Thus, we employ Principle Component Analysis (PCA) to reduce the dimensionality to 64. It is also worth to note that algorithms, such as NMTFS and KmeanS, require to perform semantic extensions on each distinct term. This will lead to a huge term dictionary for a large service set, like **Dataset_2**. The resultant service matrix will be several orders larger than $\mathbf{X}$. To avoid prohibitive computational cost, we are not including NMTFS and KmeanS for comparison. Instead, we add another two algorithms into the mix:

– **Kmeans**: Directly applying Kmeans clustering to the terms vectors in $\mathbf{X}$.
– **PKmeans**: Applying Kmeans after PCA dimensionality reduction.

Since there are no cluster labels for **Dataset_2**, we only use silhouette value to evaluate clustering quality. We set the number of clusters to 30. The number of anchor services is set to 128 and all other parameters take their default values for SFR.

Table 3 reports the clustering result on **Dataset_2**. In terms of clustering quality, SFR achieves the highest silhouette value among all the algorithms. This is consistent with the results from **Dataset_1**. It is also worth to note that SVDC fails to converge after spending over 1,000 seconds, so no silhouette value is computed. For the CPU times, we record both the construction time that is used to discover the anchor services and the clustering time for SFR. Even though SFR uses relatively long time (which is similar to the clustering time used by NMTF) for anchor service discovery, it achieves the best clustering time. The fast clustering performance of SFR further justifies that sparse functional representation indeed makes clustering easier. Once the anchor space is discovered, it can be stored and reused. Therefore, for large-scale service clustering, anchor services can be first discovered offline and then service clustering can be performed in realtime to meet different user requirements on number of clusters, distance metrics, clustering algorithms, and so on.

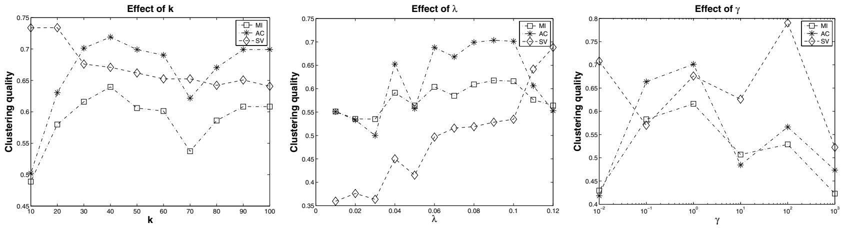

Fig. 2. Impact of Parameters

4.5 Impact of Parameters

We investigate the impact of different parameters in this section, including the number of anchor services (i.e., k), and the two penalty parameters (i.e., λ and γ). We vary one of these three parameters while keeping the other two fixed at their default values. Figure 2 shows how different parameters affect the clustering quality in **Dataset_1**.

Accuracy and mutual information always vary in a similar way with the changes of parameters. Both accuracy and mutual information reach their respective highest values when $k = 40, \lambda = 0.1$, and $\gamma = 1$, respectively. The silhouette value, on the other hand, varies differently with accuracy and mutual information. First, the SV value decreases as k increases. Recall that in SFR, services are clustered based on their relationships with the anchor services. The sparsity constraint forces services to be only related to a small subset of the anchor services. Therefore, when k is small, the sparse representation of a service will "concentrate" on a small number of anchor services. This will lead to very compact and cohesive clusters. Therefore, SV will decrease as k increases. Similar explanation is applied to the impact of λ, which enforces the sparsity constraint. Increasing λ will make $\mathbf{z}_i$'s more sparse, which has the effect of moving services closer to the relevant anchor services and further away from less relevant ones. This will also produce more cohesive clusters. Therefore, SV increases as λ increases. Instead of monotonically decreasing or increasing as with the increase of k and λ, the SV value reaches it peak value when γ is 100 and then decreases when γ increases further. In contrast, accuracy and mutual information reach their peak values when $\gamma = 1$. The discrepancy may be due to that the domain definition of the service set is not in line with the cohesiveness of the service clusters. For example, some services may be cross-domain in nature but assigned to a domain that is inconsistent with the clustering result.

The results on **Dataset_2** show very similar patterns as those of **Dataset_1** (in term of SV values because only SV values are reported for **Dataset_2**). Therefore, we skip the presentation of the results to avoid repetition.

5 Conclusion and Future Directions

We present Sparse Functional Representation (SFR), a novel service representation scheme, which is economical to store, efficient to process, and intuitive to interpret. SFR projects long and sparse term vectors onto an anchor service space, which consists of a small number of anchor services. The similarity between services is encoded by their proximity to the anchor services. The sparsity constraints enforce that each service is only related to a small subset of anchor services. This has the effect of moving services closer to the relevant anchor services and further away from irrelevant ones. These key features significantly facilitate large-scale service clustering. Comprehensive experiments on two real-world service datasets clearly demonstrate the effectiveness of SFR and its ability to scale to a large number of services. An interesting future direction is to further improve the construction performance of SFR. We plan to apply the recently developed low-rank approximation techniques, such as Colibri [14], to filter nearly duplicate or linearly dependent term vectors from the service matrix $\mathbf{X}$. Low-rank approximation allows SFR to work on a much smaller service matrix, from which anchor services are expected to be discovered more efficiently.

References

1. Bose, A., Nayak, R., Bruza, P.: Improving Web Service Discovery by Using Semantic Models. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 366–380. Springer, Heidelberg (2008)
2. Cai, D., He, X., Han, J.: Document clustering using locality preserving indexing. IEEE Trans. Knowl. Data Eng. 17(12), 1624–1637 (2005)
3. Chen, L., Hu, L., Zheng, Z., Wu, J., Yin, J., Li, Y., Deng, S.: WTCluster: Utilizing Tags for Web Services Clustering. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 204–218. Springer, Heidelberg (2011)
4. Ding, C.H.Q., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: KDD, pp. 126–135 (2006)
5. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: VLDB 2004: Proceedings of the Thirtieth International Conference on Very Large Data Bases, pp. 372–383. VLDB Endowment (2004)
6. Elgazzar, K., Hassan, A.E., Martin, P.: Clustering wsdl documents to bootstrap the discovery of web services. In: ICWS, pp. 147–154 (2010)
7. Klusch, M., Fries, B., Sycara, K.: Automated semantic web service discovery with owls-mx. In: AAMAS, pp. 915–922. ACM, New York (2006)
8. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS, pp. 801–808 (2006)
9. Liu, F., Shi, Y., Yu, J., Wang, T., Wu, J.: Measuring similarity of web services based on WSDL. In: ICWS, pp. 155–162 (2010)
10. Liu, X., Huang, G., Mei, H.: Discovering homogeneous web service community in the user-centric web environment. IEEE T. Services Computing 2(2), 167–181 (2009)
11. Lovasz, L.: Matching Theory (North-Holland mathematics studies). Elsevier Science Ltd. (1986)
12. Ma, J., Zhang, Y., He, J.: Efficiently finding web services using a clustering semantic approach. In: CSSSIA 2008: Proceedings of the 2008 International Workshop on Context Enabled Source and Service Selection, Integration and Adaptation, pp. 1–8. ACM, New York (2008)
13. Segev, A., Sheng, Q.Z.: Bootstrapping ontologies for web services. IEEE Transactions on Services Computing 5, 33–44 (2012)

14. Tong, H., Papadimitriou, S., Sun, J., Yu, P.S., Faloutsos, C.: Colibri: fast mining of large static and dynamic graphs. In: KDD, pp. 686–694 (2008)
15. Xu, W., Liu, X., Gong, Y.: Document clustering based on non-negative matrix factorization. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR 2003, pp. 267–273. ACM, New York (2003)
16. Yu, Q.: Place Semantics into Context: Service Community Discovery from the WSDL Corpus. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 188–203. Springer, Heidelberg (2011)
17. Yu, Q., Rege, M.: On service community learning: A co-clustering approach. In: ICWS, pp. 283–290 (2010)
18. Zhang, Y., Zheng, Z., Lyu, M.R.: Wsexpress: A qos-aware search engine for web services. In: ICWS, pp. 91–98 (2010)
19. Zheng, M., Bu, J., Chen, C., Wang, C., Zhang, L., Qiu, G., Cai, D.: Graph regularized sparse coding for image representation. IEEE Transactions on Image Processing 20(5), 1327–1336 (2011)