Clustering and Recommendation for Semantic Web Service in Time Series

Yu Lei[1,2], Wang Zhili[2], Meng Luoming[2], and Qiu Xuesong[2]
[1] College of Computer Science, Inner Mongolia University, Hohhot, 010021 – China
[2] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications
Haidian District, Beijing 100876 – China
[e-mail: yuleiimu@sohu.com]
*Corresponding author: Yu Lei

Received January 5, 2014; revised March 11, 2014; revised March 27, 2014; revised May 16, 2014; accepted June 18, 2014; published August 29, 2014

Abstract

Promoted by cloud technology and new websites, plenty and variety of Web services are emerging in the Internet. Meanwhile some Web services become outdated even obsolete due to new versions, and a normal phenomenon is that some services work well only with other services of older versions. These laggard or improper services are lowering the performance of the composite service they involved in. In addition, using current technology to identify proper semantic services for a composite service is time-consuming and inaccurate. Thus, we proposed a clustering method and a recommendation method to deal with these problems. Clustering technology is used to classify semantic services according to their topics, functionality and other aspects from plenty of services. Recommendation technology is used to predict the possible preference of a composite service, and recommend possible component services to the composite service according to the history information of invocations and similar composite services. The experiments show that our clustering method with the help of Ontology and TF/IDF technology is more accurate than others, and our recommendation method has less average error than others in the series of missing rate.

Keywords: Recommendation, clustering, composite service, semantic

This work was supported by National Natural Science Foundation of China [No: 61262082], Key Project of Chinese Ministry of Education [No.212025], Inner Mongolia Science Foundation for Distinguished Young Scholars [2012JQ03].

http://dx.doi.org/10.3837/tiis.2014.08.010

1. Introduction

In the history of software, reusability is an eternal topic. From the functions in procedure-oriented languages, to the inheritance in object-oriented languages, the aspects in aspect-oriented programming, and SOA (Service-Oriented Architecture), the granularity of reusability is increasingly larger from one piece of code to a whole system. Though refactoring and version updating are inevitable even in SOA systems, some service providers still keep commitment to avoid changing interface invoked by outside users. But some service providers change interface frequently. Thus, we need a method to record the change history of services to predict their future behaviors. Frequent and irregular changes of a service implies that it lacks planning and may be an unsustainable evolution. This kind of history information should be used to infer that the service is not suitable to be a component of a composite service, since many unstable changes of component services will weaken the performance of a composite service. However, if one service required by users only works well with the older version of other services, we should always recommend these older services to it.

Internet and language experts constructed huge ontology to formally represent knowledge as a set of concepts within a domain, using a shared vocabulary to denote the types, properties and interrelationships of those concepts. Its purpose is to make Internet document readable and understandable by machine. WSDL (Web Service Description Language), a kind of Internet language describing how and where to invoke services, is not good enough for a machine-readable language for semantics. To describe Web services from grammar level to semantic level, experts build domain ontology on top of WSDL, including SAWSDL (Semantic Annotation WSDL), WSML (Web Service Modeling Language) and OWL-S (Ontology Web Language-Semantic) [1] etc. Domain ontology, takes much time and manpower of domain experts to build and maintain. With the rapid development of Web service and introduction of new concept, ontology requires frequent update. In this perspective, ontology partially reduces manpower. To alleviate this problem, we propose to use technology from information retrieval domain, TF/IDF (Term Frequency / Inverse Document Frequency) [2]. TF/IDF analyzes a group of related documents to identify the keywords, which represent the main idea of the document. Meanwhile, it is not sensitive to new concepts because that it does not construct the tree of concepts like ontology. In addition, it can capture the related concepts, regardless of new or old concepts, by means of analyzing their co-appearance in the same documents.

Facing variety of services, it is necessary to narrow the scope of required services. As no one knows how many kinds of services should exist, we use clustering instead of classifying to narrow the scope. Clustering provides multiple levels of granularities, making recommendation faster. Small granularity provides more clusters, and each of which contains small number of services; by contrast, large granularity provides less clusters, and each of which contains a large number of services. Therefore, we can begin with the small granularity. If we are able to find out proper services in clusters, this job can be finished quickly; if not, we go to the larger granularity. Furthermore, not all of Web service data are clustered in the shape of sphere. Some technology, such as SVM (Support Vector Machine), can solve this problem by lifting the data to additional dimensions, but its computation complexity is high.

Facing plenty of services, it is necessary to recommend component services to a new composite service according to similar composite services. The similar composite services may have similar aims, approaches or functions, even similar QoS as they have similar

component services. This feature rapidly makes excellent component services popular, once they are selected in the beginning by several composite services. The composite services and component services form a two-dimension matrix, and the element of the matrix indicates one component service's utility to a composite service. Because we do not know how well some component services are composed to a composite service, the matrix is sparse and has missing data. The methods of matrix factoring have proven to be effective for recommendation of user preference in the contest of Netflix. SVD (Singular Value Decomposition), one of these methods, can derive the latent semantic information from partial information to predict the missing information.

To solve these problems, we proposed a method, called CRE (Clustering and Recommendation for OWL-S Web services in Evolution). In section 2, we reviewed the previous work in service clustering and recommendation. In section 3, a service clustering method that can solve irregular shape was proposed, and the outputs of the method construct two subspaces (component service-topic subspace and composite service-topic subspace). In section 4, a model called service space was constructed for service recommendation, and a service recommendation method, based on matrix decomposition, was proposed. In section 5, experiments about clustering and matrix decomposition are conducted. The steps of our method are as follows.

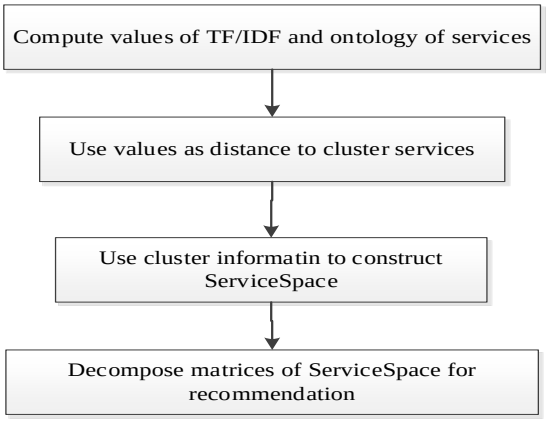

Fig. 1. Steps of propose method

The contributions of this paper are:

First, we combine TF/IDF and ontology to compute more precisely the similarity of Web services.

Second, we use Ward's distance to identify irregular shape, and then use Agglomerative method to expedite the process of clustering.

Third, considering four factors, we build a service space model, by which we propose a service recommendation method.

2. Related Work

With the recent success of recommendation systems in e-commerce, like Amazon, and online content distribution, like Netflix, analogous recommendation techniques have been proposed to recommend services [3, 4]. As the key technology of recommendation systems, Collaborative Filtering (CF) [3] considers user disparity and uses experience of one group of

users to predict experience of another similar group of users, who actually have not invoked the services yet. Current CF techniques generally can be divided into two classes: neighborhood-based and model-based. The former needs to identify users who share similar QoS experience with new users. The number of services invoked by each user is limited, thus identifying the number of users who have invoked the same services is usually difficult. Consequently, the QoS data sparsity issue hinders the neighborhood-based approaches. Model-based approaches build a global model according to the observed QoS data, which can be used to predict QoS values. Nevertheless, model-based approaches need to build models in advance, and some models have difficulty explaining results of predictions.

As the number of Web services grows fast, it is more difficult to discover services that satisfy both functional and non-functional users' requirements. Existing techniques that can be applied to service discovery fall into two broad categories: functionality-based and QoS-based. Functionality-based service discovery is to discover services that satisfy user-required functionality. Information retrieval techniques and semantic technologies have been used to improve the accuracy of service discovery. QoS-based approaches are used to differentiate services based on their QoS performance [5]. [6-8] propose user-based CF algorithms, and they predict QoS by assuming that similar or trustable users tend to receive similar QoS from similar services. A hybrid algorithm was developed, which enhances the user-based approach by integrating item-based CF to obtain better QoS prediction accuracy [7]. Thereafter, many similar algorithms have been developed. They use additional information, such as users' locations [9], invocation frequencies of services, and users' query histories to improve the QoS recommendation. Both user and item based approaches adopt the neighborhood centric strategy in CF, which searches the local neighborhood to find similar users or for recommendation.

Recently, a model based CF algorithm was proposed and obtained higher prediction accuracy [10]. First, it uses the user-based approach to find top-k similar users. Second, matrix factorization is employed to construct a global model according to the user neighborhood information. Services prediction can be modeled as a general matrix completion problem, which has applications in many science and engineering domains. It presumes that the matrix has a low-rank or approximately low-rank structure. However, rank minimization is NP-hard because the rank function is non-convex and discontinuous [11, 12]. It has been demonstrated that the nuclear norm provides the tightest convex relaxation of the rank minimization problem [13]. Many optimization algorithms have been developed recently, which apply nuclear norm as a convex surrogate to effectively tackle the matrix completion problem.

Zhang [14] etc. proposed an approach which combines social network and collaborative filtering techniques in a unified framework to predict the missing QoS values of manufacturing services for an active service user. Their method alleviates the data sparsity and the cold start problems that hinder the traditional collaborative filtering techniques. Sun [15] etc. presented a new similarity measure for web service similarity computation and proposed a novel collaborative filtering approach, called normal recovery collaborative filtering, for personalized web service recommendation. They conducted large-scale real-world experiments, and the results showed their approach achieves better accuracy than other competing approaches. Cao [16] etc. designed a cube model to describe the relationship among providers, consumers and Web services. They presented a standard deviation based hybrid collaborative filtering and an inverse consumer frequency based User Collaborative Filtering (UCF) for potential consumers recommendation. A decision-making process of bidirectional recommendation is provided for both providers and consumers. Using Planet-Lab for experiments, they verified their method is much better than extant methods on

recommendation quality, including the CF based on user, the CF based on item and general HCF(Hybrid Collaborative Filtering). Wu [17] etc. presented a neighborhood-based collaborative filtering approach for QoS-based selection. Compared with existing methods, the proposed method has three new features: the adjusted-cosine-based similarity calculation to remove the impact of different QoS scale; a data smoothing process to improve prediction accuracy; a similarity fusion approach to handle the data sparsity problem. In addition, a two-phase neighbor selection strategy is proposed to improve its scalability. Deng [18] etc. studied the trust relationships between users and Web services using network modeling and analysis techniques. They proposed a collaborative filtering algorithm called Trust-Based Service Recommendation (TSR) to provide personalized service recommendations. These methods extend the traditional Web service composition techniques [29-21].

Related works have some deficiencies. First, they do not combine TF/IDF and ontology to effectively cluster Web services. Second, they use general classifying or clustering method, whereas we use Agglomerative method and Ward's distance to expedite recommendation and identify irregular shape. Third, they do not consider version changes of web services. Our method can effectively solve these deficiencies.

3. Clustering

3.1 Service Similarity

OWL-S is an ontology, within the OWL-based framework of the Semantic Web, for describing Semantic Web Services. It enables users and software agents to automatically discover, invoke, compose, and monitor Web services, under specified constraints. The OWL-S ontology has three main parts: the service profile, the process model and the grounding [1]. The service profile is used to describe what a service does. This information is primarily meant for human reading, including the service name and description, limitations on applicability and quality of service, publisher and contact information. Our method aims to further reduce the human labor by "computing" the service profile. In another words, we compute similarity of every profile of Web services to recommend proper services to a composite service.

The similarity of two Web services will be computed in two aspects, functionality description and text description. Functionality description, a component of OWL-S profile, is the specification of what functionality a service provides and the specification of the conditions that must be satisfied for a successful result. In addition, the profile also specifies what conditions result from the service, including the expected and unexpected results of the service activity. These properties are defined as IOPE, which means Input, Output, Precondition and Effect (a.k.a. result). We use $Sim_{function}$ to denote this kind of similarity in Equ.1. Text description, another component of OWL-S profile, provides a brief description of the service. It summarizes what the service offers, and describes what the service requires to work. We use Sim_{text} to denote this kind of similarity in Equ.1.

$$Sim_{Service}(S_1,S_2)=w_1 \times Sim_{function}(S_1,S_2)+w_2 \times Sim_{text}(S_1,S_2) \quad (1)$$

3.1.1 Functionality similarity

In the view of ontology, knowledge is represented formally by a set of concepts and relationships among these concepts within a domain. Ontology can be used to model a knowledge domain and support reasoning concepts in it. The relation of two terms showed in a

domain can be measured by computing their semantic distances in ontology. There are several standardized ontologies available for use, and we use WordNet (ontology) in this paper. Supposing $Sim_{ontology}$ is the semantic similarity between two terms (T_1 and T_2), Equ.2 denotes that their similarity is decided by their common father's level (comF) and their own level in an ontology tree. In short, the closer the distance, the similar the concepts.

$$Sim_{Ontology}(T_1,T_2)=1-[(2^{-1-level\ (comF)}-2^{-1-level\ (T_1)})+(2^{-1-level\ (comF)}-2^{-1-level\ (T_2)})] \quad (2)$$

For instance, **Fig. 2** is an ontology tree. Equ.2 can be used to compute the similarity of terms T_1 and T_2.

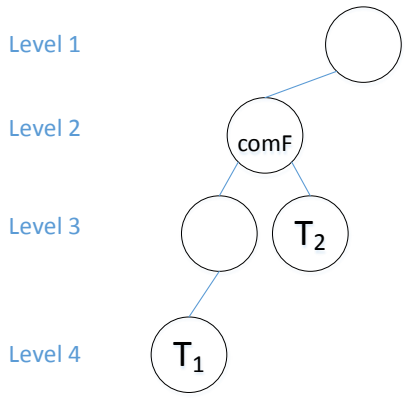

Fig. 2. An ontology tree

Unfortunately, ontology has its Achilles' heel. Not all terms used in Web services are included in ontology. The time for constructing the ontology is long. For example, WordNet cannot measure the semantic similarity of input names "Android4.4" and "IOS7", because terms of "Android4.4" and "IOS7" have not been recorded in WordNet. Therefore, an additional process needs to be introduced to solve this problem. The process should capture these kinds of related terms by means of analyzing documents in the Internet. We randomly obtained documents from www.seekda.com, a Web service search engine which can retrieve more than 5000 Web services information, for our experiments. In fact, any methods that can find out latent semantics between terms will be qualified, and relations between terms will be computed more accurately. Equ.3 is our solution.

$$Sim_{Topic}(T_1,T_2)=\frac{N(T_1\cap T_2)}{N(T_1)+N(T_2)-N(T_1\cap T_2)} \quad (3)$$

Where, $N(T_1\cap T_2)$ is the number of documents that two terms coexist, and $N(T_1)$ is the number of documents that term T_1 exists. This kind of similarity is topic-related. Next, we sum up Equ.2 and Equ.3 to complement each other for measuring the similarity, as shown in Equ.4.

$$Sim(T_1,T_2)=w_3\times Sim_{Ontology}(T_1,T_2)+w_4\times Sim_{Topic}(T_1,T_2) \quad (4)$$

Finally, the functionality similarity can be computed respectively in terms of category, precondition, effect, output and input of services, as shown in Equ.5.

$$\begin{aligned}\mathrm{Sim}_{\mathrm{function}}(S_1,S_2)=&w_5\times\mathrm{Sim}(\mathrm{Category}_1,\mathrm{Category}_2)+w_6\times\mathrm{Sim}(\mathrm{Pre}_1,\mathrm{Pre}_2)\\&+w_7\times\mathrm{Sim}(\mathrm{Eff}_1,\mathrm{Eff}_2)+w_8\times\mathrm{Sim}(\mathrm{Out}_1,\mathrm{Out}_2)+w_9\times\mathrm{Sim}(\mathrm{In}_1,\mathrm{In}_2)\end{aligned} \quad (5)$$

Weighting values can be changed to adapt to different environments. Some environments establish semantic information, and have mature ontology, then we should assign higher weighting values on semantics. Some environments have immature ontology and abundance of text information, then we should assign higher weighting values on text. In another way, we can use parameter learning methods, such as EM (Expectation Maximum) method, to learn $w_1 \ldots w_9$ to decide their optimal values. In this paper, we simply assign an average value to each weight.

3.1.2 Text similarity

To mine the similarity of text organized by paragraphs, we use the TF/IDF technology, which is an information retrieval method. The texts of all Web services in a service recommendation system are considered as a corpus. TF/IDF approach is used to weight the importance of terms in the corpus. TF/IDF will find out the terms that appear more frequently in a specific text description of a Web service than other descriptions. Suppose the text description of a Web service includes its name, functional description, major Web APIs and some social tags. The first step is to preprocess the corpus of text description to standardize words or tokens in this corpus and obtain a set of valid terms. The preprocessing contains word standardization, word rooting and noise word removing. The second step is to use TF/IDF technology to weight the importance of terms in the corpus. TF/IDF is a product of two statistics, TF (Term Frequency) and IDF (Inverse Document Frequency). TF is the number of times for a given term in a given document. Since a term appears in a longer document may have higher TF value than in a shorter document regardless of the actual importance of the term, TF is generally normalized to avoid the bias towards longer documents. Therefore, the frequency of the term T in the text description Text_i can be measured by TF. $N(T,\mathrm{Text}_i)$ denotes the number of times that term T appears in the text_i, and $|\,\mathrm{text}_i\,|$ denotes the total number of terms in this text.

$$TF(T,Text_i)=\frac{N(T,Text_i)}{\left|Text_i\right|}$$

IDF measures whether the term in a Web service is common or rare across all text descriptions of Web services. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$IDF(T,Text_i)=\log\frac{N(Text)}{\left|Text_i:T\in Text_i\right|}$$

Where, N(Text) is the total number of texts in the Web service corpus. $|\,\mathrm{Text}_i : T \in \mathrm{Text}_i\,|$ is the number of texts where term T appears. TFIDF is shown below.

$$TFIDF(T,Text_i)=TF(T,Text_i)\times IDF(T,Text_i)$$

A high TF in the given document and a low text frequency of the term in the whole collection of texts will produce a high weight in TF/IDF. Thus, the TF/IDF tends to filter out the generic terms and preserve the important terms in text description of a Web service. Since

the ratio inside the IDF's log function is always greater than or equal to 1, the value of IDF and TF/IDF is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the IDF and TF/IDF closer to 0.

A vector of term weights will be used to represent the text description. Each text description of a Web service will be transformed into a vector of terms and weights based on the TF/IDF approach. Finally, we compute the similarity between two text descriptions using the cosine similarity measurement, as shown in Equ.6. $TFIDF_{1,i}$ means the value of term *i* in text 1. *n* is the total number of terms in two texts.

$$\mathrm{Sim}_{\mathrm{text}}(S_1,S_2)=\frac{\sum_{i=1}^{n} TFIDF_{1,i}\times TFIDF_{2,i}}{\sqrt{\sum_{i=1}^{n} TFIDF_{1,i}^2}\times\sqrt{\sum_{i=1}^{n} TFIDF_{2,i}^2}} \tag{6}$$

3.2 Clustering Algorithm

There are many famous clustering algorithms, like k-means clustering. The disadvantage of k-means is too sensitivity for outliers and initial centroid, and it is not suitable to discover non-hyper spheres, which means irregular shapes. In contrast, hierarchical clustering can find a centroid of an irregular shape. One strength of hierarchical clustering is that it does not assume the number of clusters, meaning that any desired number of clusters can be obtained by cutting a tree at a proper level. Hierarchical clustering is a method that iteratively divides or aggregates classes. Strategies for hierarchical clustering generally fall into two types: Agglomerative, a "bottom up" approach, merges classes iteratively until one class; Divisive, a "top down" approach, splits the top class into multiple classes recursively.

The key operation is the computation of the distance between two clusters. Different definitions of the distance between clusters lead to different algorithms. Single-link distance between clusters is defined by the two most similar objects, i.e. their minimum distance. However, it is sensitive to noise and outliers, and it produces elongated clusters. Complete-link distance between clusters is defined by the two most dissimilar objects, i.e. their maximum distance. However, it tends to break large clusters, and all clusters tend to have the same diameter. Group average distance between clusters is the average distance between any objects in each cluster. It compromises between Single-Link and Complete-Link, but biases to globular clusters. Centroid distance between clusters is the distance between the centroid of two clusters. Ward's distance is similar to group average and centroid distance, and it is less susceptible to noise and outliers. Assume that there are three clusters called C_i, C_j and C_{ij}. Clusters C_i and C_j are aggregated to form a new single cluster called C_{ij}. Ward's distance between cluster C_i and the new cluster C_{ij} in the example above is calculated as:

$$D_w\left(C_i,C_{ij}\right)=\sum_{x\in C_i}\left(x-r_i\right)^2+\sum_{x\in C_j}\left(x-r_j\right)^2-\sum_{x\in C_{ij}}\left(x-r_{ij}\right)^2 \tag{7}$$

Where, r_i is the centroid of C_i, r_j is the centroid of C_j, and r_{ij} is the centroid of C_{ij}. Our agglomerative algorithm is as follows.

Algorithm 1. Semantic Clustering

Input: Every services ready to be clustered.
Quantity of clusters, k.
Output: k clusters.

Make each service a cluster.
Calculate all pair-wise distances of services
Repeat
Find two clusters that are nearest to each other.
Merge them to form a new cluster.
Calculate the distance from the new cluster to all other clusters by means of Equ.7.
Until there is only one cluster left.

4. Service Recommendation

4.1 Service Space

Service space, shown in **Fig. 3**, is a multi-dimensions space or several intersected two-dimension planes, whose axes include atomic service (F_s), composite service (F_c), class (F_a) and time (F_t).

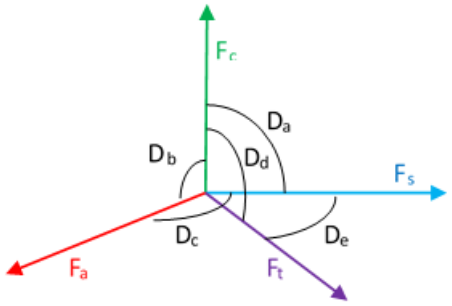

Fig. 3. Service space

In the service space, we are interested in some meaningful subspaces, including F_s-F_c subspace (D_a), F_a-F_c subspace (D_b), F_a-F_t subspace (D_c), F_t-F_c subspace (D_d) and F_t-F_s subspace (D_e). D_a describes which atomic services are composed to a composite service. In the matrix, value one denotes an atomic service is a component service of the composite service, and value zero denotes the opposite. D_b describes each composite service belongs to which class. D_c describes each atomic service belongs which class. Notice that one atomic service and one composite service can coexist in one class, which means they may be replaced by each other with respect to functionality. D_d describes a tendency of a composite service, because of a service evolution or other reasons. Time Series Forecasting methods can be used for predicting this tendency, but we do not discuss them due to limit of pages. D_e describes a tendency of an atomic service. In addition, tendency can be affected by many factors, such as QoS (including price and throughput, etc.). Weightings of these factors can be analyzed by Factor Decomposition.

Each subspace above can be represented by a two-dimension matrix. Assuming D_a, D_b, D_c, D_d and D_e, in the form of matrix, have actual values. F_s, F_c, F_a and F_t, also in the form of matrix, are decomposed matrixes. Each D_i is decomposed by two F_j, e.g. $D_a \approx F_c F_s$.

Our clustering algorithm will fill in the values of D_b and D_c. Afterwards we will use matrix decomposition to fill in other subspaces.

4.2 Model for Prediction

In this section, we model the recommendation problem as a matrix completion problem. In another word, we want to predict the values of blank elements in the matrix. A loss function is used to measure the quality of our prediction. Supposing X is input, and Y is output (we do not know the values of Y yet). We want to calculate an output f(X) to predict the actual output Y, where f(X) and Y are as close as possible. Therefore, our loss function is as follows.

$$L(Y, f(X)) = (Y - f(X))^2 \tag{8}$$

The smaller the value of the loss function, the better the prediction. In another word, Equ. 8 needs to be minimized as follows.

$$\min L(Y, f(X)) \tag{9}$$

However, if just using Equ.9, an "over-fitting" problem may arise. "Over-fitting" is a situation that a generated model is excessively complex and it exaggerate minor fluctuations in the data. To avoid it, a technique called Structural Risk Minimization is used. Similar to regularization, this technique add a penalty term to a loss function, meaning it will punish complex models. The more complex the model, the bigger the penalty. The rational is that simple models well predict the actual data, and have minimum risks. Equ.9 turns to be a new formula as follows.

$$\min\{L(Y, f(X)) + J(f)\} \tag{10}$$

In the service space, above formula would be a sum of loss functions as follows.

$$L=L_a+L_c+L_s+J(F) \tag{11}$$

A good prediction towards blank elements of the matrix is a good recommendation for composite services, which implies an atomic service may be suitable for a component of a composite service. Next step, we will use matrix factorization to predict the component services.

4.3 Matrix Factorization

Matrix factorization is to decompose a matrix into a product of two matrices. In this case, Y and f(X) are two-dimension subspaces, meaning that they can be represented by two matrix. Computing Y-f(X) in the service space is to compute distance between Y and f(X). The distance of the two matrix can be calculated by matrix norm, an extension of the notion of a vector norm to matrices. Frobenius norm, a kind of matrix norm, is a suitable element-by-element measurement:

$$\| A \| = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

To recommend atomic services to a composite service, we focus on the subspace related to F_s and F_c. Given the Frobenius norm, the loss functions can be defined as follows.

$$L_a = \left\| D_a - F_c F_s \right\|^2 \tag{12}$$

$$L_c = \left\| D_b - F_c F_a \right\|^2 + \left\| D_d - F_c F_t \right\|^2 \tag{13}$$

$$L_s = \left\| D_c - F_a F_s \right\|^2 + \left\| D_e - F_s F_t \right\|^2 \tag{14}$$

The loss functions (Equ. 12-14) are the objective function to be optimized, which is NP-hard [11]. Hence, we use a method of gradient descent to compute the loss functions. Gradient indicating a direction of maximum decrement is as follows.

$$\frac{\partial L}{\partial F} = \frac{\partial L_a}{\partial F} + \frac{\partial L_c}{\partial F} + \frac{\partial L_s}{\partial F} \tag{15}$$

Partial derivatives, the components of gradient, are Equ. 16-18. Equ.16 is partial derivatives of the loss function L_a in Equ.12. The partial derivatives of F_t is zero.

$$\begin{cases} \dfrac{\partial L_a}{\partial F_c} = \left(F_c F_s - D_a \right) F_s \\ \dfrac{\partial L_a}{\partial F_s} = \left(F_c F_s - D_a \right) F_c \end{cases} \tag{16}$$

Equ.17 is partial derivatives of the loss function L_c in Equ.13.

$$\begin{cases} \dfrac{\partial L_c}{\partial F_a} = \left(F_c F_a - D_b \right) F_c \\ \dfrac{\partial L_c}{\partial F_c} = \left(F_c F_a - D_b \right) F_a + \left(F_c F_t - D_d \right) F_t \\ \dfrac{\partial L_c}{\partial F_t} = \left(F_c F_t - D_d \right) F_c \end{cases} \tag{17}$$

Equ.18 is partial derivatives of the loss function L_s in Equ.14.

$$\begin{cases} \dfrac{\partial L_s}{\partial F_a} = \left(F_a F_s - D_c \right) F_s \\ \dfrac{\partial L_s}{\partial F_s} = \left(F_a F_s - D_c \right) F_a + \left(F_s F_t - D_e \right) F_t \\ \dfrac{\partial L_s}{\partial F_t} = \left(F_s F_t - D_e \right) F_s \end{cases} \tag{18}$$

4.4 Service Recommendation

The service recommendation method will recommend component services with similar topics (by computing F_a) and good performance (by computing F_t) to composite services. The blank elements of D_a can be thought of missing data, and the recommendation problem becomes a prediction problem using some known data to infer those missing data. The following equation shows two component matrices can "recovery" an initial matrix by matrix multiplications.

$$\hat{D}_a = F_c F_s \tag{19}$$

Gradient descent is a first-order optimization algorithm that can be used for solving the recommendation problem. The extensions of gradient descent show better performance, e.g. stochastic gradient descent and conjugate gradient method. These methods are suitable for matrix factorization. Our algorithm for service recommendation is as follows.

Algorithm 2.

Service Recommendation
/* Obtain F_a, F_c, F_s and F_t */
While not exceed the max number of iterations
or not convergence
  Calculate the objective function, Equ.11.
  Calculate the gradients of objective function, Equ.15.
  Calculate step direction by minimize Equ.15.
  Calculate step size, go one step
End while
/*fill in missing data*/
Generate D_a using Equ.19.
All of elements in each row of D_a are recommendation for a composite Web service.

4.5 Time Complexity

In general cases, the time complexity of our clustering method is $O(n^3)$. But, using priority queue can reduce the time complexity to $O(n^2 \log n)$.

The computation of Equ.11 and Equ.15 takes large parts of our service recommendation method. The time complexity of evaluating Equ.11 is $O(N \times R)$, where N is the number of known data in the matrices, and R is the maxium rank of decomposite matrices. The time complexity of evaluating Equ.15 in one iteration is also $O(N \times R)$. Therefore, our service recommendation method is linear with respect to the scale of service matrices, which means our method is efficent and can scale to large data sets.

Unlike other model-based recommendation systems, which need to obtain models from historical data before recommending component services actually, our method can be deployed immediately in production environments because recommending component services based on ontology similarity dose not need historical data. In this sense, the proposed method takes advantage of content-based recommendation.

5. Experimental Results and Analysis

Our configuration of experimental platform is as follows. Operation system: Microsoft Windows 7. CPU: Intel Core P8700, 2.53 GHz. RAM: 4 GB. Programming language: Java. Ontologies building: Jena Framework. Documents of WSDL and OWL-S are retrieved from

our repository, which stores 7801 records of Web services. Most data in the repository is from datasets of three Web service related contests, which are "Semantic Services Selection (S3) Contest", "Semantic Web Service Challenge" and "Web Services Challenge (WSC)". The first two contests focus on service discovery and mediation, where we use their service ontology. The third contest focuses on service composition, whose data includes ontology and WSDL. For our experiments, the champion method (WSC-2009) was used to compose some composite services to construct the initial matrix D_a. OWL-S were generated from description information. Referring to the real services from www.seekda.com, we modified QoS values of original services to simulate different versions of Web services, by which we can construct the time axis.

5.1 Evaluation for Clustering

For comparison, a set of Web service data have been classified manually in advance. The classes are Insurance, Bank, Hospital and Repair. We use entropy, purity, precision, recall and F-measure as evaluation criteria of cluster quality.

Entropy measures how the various services are distributed within each cluster. Smaller entropy values mean better clustering solutions. The entropy of a cluster is given by:

$$Entropy_{cluster}(C_i) = -\sum_{i=1}^{n} \frac{N^i}{N_{ctotal}} \log \frac{N^i}{N_{ctotal}}$$

Assuming the number of cluster is *n*. N_{ctotal} is the number of services in the cluster. N^i is the number of services supposed to be assigned to cluster *i*, but incorrectly assigned to cluster C_i . The whole entropy of clusters is as follows, where N^i_{ctotal} indicates the number of services in cluster *i*.

$$Entropy_{whole} = \sum_{i=1}^{n} \frac{N^i_{ctotal}}{N_{wtotal}} Entropy_{cluster}(C_i)$$

Purity can be obtained by calculating the number of services that are assigned correctly.

$$Purity = \frac{1}{N_{wtotal}} \sum_{i=1}^{n} \max\{N^i_{correct}\}$$

$N^i_{correct}$ indicates the number of services that are assigned correctly to cluster *i*.

Fig. 4 shows CRE obtains highest purity values among the three methods. Ontology method only uses ontology, and TF/IDF method only uses TF/IDF information.

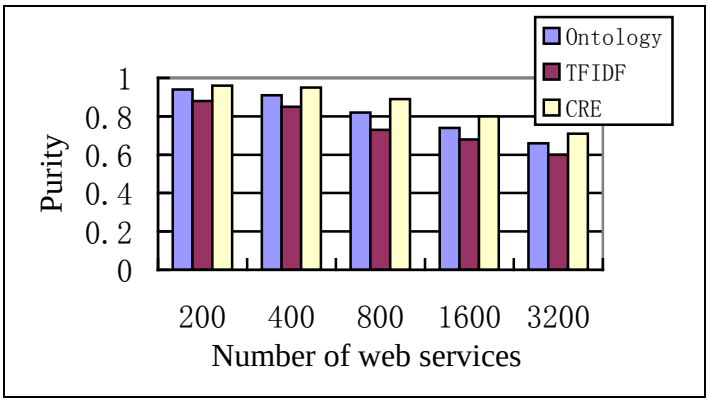

Fig. 4. Purity of clustering

Fig. 5 shows that entropy increases when the number of services increases. However, CRE's increment rate of entropy is slower than others, and the decrement rate of purity is slower than these methods. Moreover, CRE obtains lower entropy values and higher purity values, improving the clustering performance.

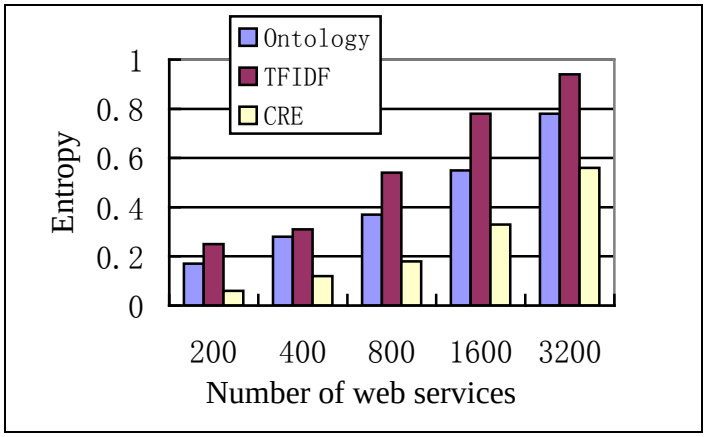

Fig. 5. Entropy of clustering

For evaluation indicators of classification, the terms *true positives*, *true negatives*, *false positives*, and *false negatives* compare results of a classifier with trusted correct judgments. The terms *positive* and *negative* refer to the classifier's prediction (expectation), and the terms *true* and *false* refer to whether this prediction corresponds to the correct judgment. This is illustrated by **Table 1**.

Table 1. Comparison matrix

	Actual class (Observation)	
Predicted class (Expectation)	***tp*** (true positive) Correct result	***fp*** (false positive) Unexpected result
	fn (false negative) Missing result	***tn*** (true negative) Correct absence of result

High recall means that an algorithm returns most of the relevant results, while high precision means that an algorithm returns substantially more relevant results than irrelevant results.

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

F-measure is a harmonic mean of precision and recall, which can be interpreted as a weighted average of the precision and recall.

$$F\text{-}measure = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The experimental results in **Table 2**, **Table 3** and **Table 4** show that the precision values and recall values of the Repair group are highest in our approach, indicating that almost all of Web services belonging to the Repair group are successfully classified into the right cluster.

Table 2. F-measure of Ontology

Class	Ontology		
	Precision	Recall	F-measure
Insurance	83.2	89.1	86.05
Bank	90.7	83.9	87.17
Hospital	93.3	90.5	91.88
Repair	95.7	94.3	94.99

The Insurance and Bank groups have the lowest precision and recall values among three methods, because some functions or their parameters of the services belonging to the Bank group are classified incorrectly into the Insurance group. When we analyze WSDL and OWL-S documents in real cases, ontology and some terms used in services do not provide sufficient details for identification of their domains. For example, the “Withdraw Money” service belonging to the Bank category is not successfully classified into the Bank group. In this case, its service fails to participate with other services, such as “Loan for Short Term” service, “Obtain Credit card” service and many others in the Bank group.

Table 3. F-measure of TF/IDF

Class	TF/IDF		
	Precision	Recall	F-measure
Insurance	79.7	84.9	82.22
Bank	88.5	82.9	85.61
Hospital	91.5	89.4	90.44
Repair	91.3	89.2	90.24

In another example, the TF/IDF method does not identify the correct domain from terms used in the “Withdraw Money” service. According to the three Tables, our approach has highest precision, recall and F-measure values than the other two methods.

Table 4. F-measure of CRE

Class	CRE		
	Precision	Recall	F-measure
Insurance	87.6	92.7	90.08
Bank	91.2	89.3	90.24
Hospital	95.4	93.9	94.64
Repair	97.1	95.6	96.34

5.2 Evaluation for Recommending

We use MAE (Mean absolute error) to evaluate the quality of our recommendation. In statistics, MAE is a quantity used to measure how close forecasts or predictions are to the eventual outcomes. The mean absolute error is given by

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|f_i - y_i\right|$$

As the name suggests, the mean absolute error is an average of the absolute errors, $|f_i - y_i|$, where f_i is the prediction and y_i the true value. We conduct a set of experiments to assess the effectiveness of the proposed algorithm of recommendation. We use the averaged Round-Trip Time (RTT) to evaluate the performance. The experiments are conducted on two QoS datasets: TestSet_1 and TestSet_2. Both test set are collected by exploiting our Web service repository. The two QoS datasets are as follows:

TestSet_1: 100 composite services are used to invoke 200 component services. Thus, a matrix Q with 100 × 200 entries is constructed. Each composite service invokes each component service for 50 times. Thus, the entry of Q_{ij} denotes the averaged RTT that composite service *i* invokes component service *j*. One million service invocation records are collected in total. Then, we randomly delete a certain percentage (60%–90%) of entries from Q, leading to a small subset of entries. At last, we compare the removed RTT entries with the predicted values to evaluate the accuracy of QoS prediction.

TestSet_2: 300 composite services are used to invoke 6000 component services. Each composite service invokes each component service for one time, and the RTT is used for evaluation. Like TestSet_1, we construct a matrix P from the data, which has 300 × 6000 entries, and then apply the same strategy to delete entries and evaluate the results of QoS prediction.

We implemented six competitive collaborative filtering methods for the prediction, which include both neighborhood-based and model-based approaches. These algorithms are described as follows:

1. PearsonU is a user-based algorithm, which uses Pearson Correlation Coefficient as similarity measure.
2. PearsonI is an item-based algorithm, which uses Pearson Correlation Coefficient as similarity measure.
3. CosineU is a user-based algorithm, which uses Cosine distance as similarity measure.
4. HybridUI, a hybrid collaborative algorithm, combines both user-based and item-based approaches. It uses their prediction accuracy as the aggregation weights [7].
5. QoSUI is a NMTF (Non-negative Matrix Tri-Factorization) based approach that co-clusters users and items. It uses the cluster structure for QoS prediction [22]. A weighting mechanism is adopted to minimize a new objective function.

6. TraceM is the singular value threshold algorithm that solves the problem of matrix completion by minimizing trace norm [11].

Table 5. MAE of methods on experiment 1

Missing rate / Method	60%	70%	80%	90%
PearsonU	0.0324	0.0558	0.0641	0.0985
PearsonI	0.0315	0.0417	0.0452	0.0739
CosineU	0.0318	0.0493	0.0614	0.0814
HybridUI	0.0301	0.0432	0.0420	0.0703
QoSUI	0.0290	0.0451	0.0525	0.0598
TraceM	0.0527	0.0688	0.0719	0.0921
CRE	0.0298	0.0435	0.0447	0.0582

Table 5 compares our CRE algorithm with the six representative collaborative filtering algorithms for MAE performance. CRE outperforms QoSUI and TraceM, which clearly justifies the effectiveness of our method. For sparse QoS data (e.g., when the missing rate is no more than 90%), the MAE performance of our method is comparable with the best neighborhood-based method, HybridUI, which integrates the user-based and item-based algorithms, and has their advantages. However, when QoS data becomes sparser, the MAE performances of all neighborhood-based approaches decrease significantly. This indicates that neighborhood-based approaches suffer from data scarcity, making them less attractive for service recommendation given the extremely scarce QoS data. The model-based approaches, especially those that uses clustering-based models, including QoSUI and CRE, are less sensitive to data scarcity.

Table 6. MAE of methods on experiment 2

Missing rate / Method	60%	70%	80%	90%
QoSUI	0.0464	0.0514	0.0553	0.0578
TraceM	0.0782	0.0729	0.0637	0.0663
CRE	0.0479	0.0489	0.0541	0.0552

To demonstrate how CRE performs on large-scale prediction, **Table 6** shows the MAE performance on TestSet_2. We exclude all the neighborhood-based algorithms in this dataset because such algorithms need a similarity search for each individual user based upon each service. For large-scale QoS prediction, these algorithms run extremely slow. In contrast, the model-based approaches, such as TraceM and QoSUI, exploit simple gradient steps or efficient update rules, to compute all missing QoS entries simultaneously, which exhibit excellent computational scalability. In addition, the results from TestSet_1 further show that neighborhood-based algorithms suffer from the data scarcity issue. **Table 6** shows that CRE achieves the best MAE performance over different missing rates. The results also show that these model-based approaches are robust to data scarcity. MAE of CRE and QoSUI only drop slightly when the missing rate increases. TraceM even achieves better MAE for more sparse QoS data.

6. Conclusion

For the convenience of service usage and service composition, a method of service clustering and service recommendation is proposed in this paper. Firstly, we compute the similarity of services with the help of semantic and TF/IDF information, which can be extracted from WSDL or OWL-S documents. Secondly, we classify services by their similarities. Thirdly, we use matrix decomposition, which is able to recovery incomplete information, to recommend component services to composite services. The experiments show our method has higher accuracy, purity and f-measure, as well as lower entropy. Some aspects can be improved and extended in our method. The weightings of the cluster factors can be learned by EM algorithm. Our recommendation model, or service space, is extendable in multiple dimensions due to we use matrices to represent the relations among QoS, functionality, tendency and so on. If extending more relations for recommendation, like social network, we can construct more axes and matrices.

References

[1] http://www.w3.org/Submission/OWL-S/#4

[2] Hong, TP, Lin CW, Yang KTU, et al., "Using TF-IDF to hide sensitive itemsets," *Applied Intelligence*, 4(38): 502-510, 2013. Article (CrossRef Link)

[3] Y. Jiang, J. Liu, M. Tang, and X. F. Liu, "An effective web service recommendation method based on personalized collaborative filtering," in *Proc. of IEEE International Conference on Web Services*, 211–218, 2011. Article (CrossRef Link)

[4] Q. Zhang, C. Ding, and C.-H. Chi, "Collaborative filtering based service ranking using invocation histories," in *Proc. of IEEE International Conference on Web Services*, 195–202, 2011. Article (CrossRef Link)

[5] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Trans. Web*, 1(1):6, 2007. Article (CrossRef Link)

[6] Kim, Hyoung Do, "Applying consistency-based trust definition to collaborative filtering," *KSII Transactions on Internet and Information Systems*,3(4), 2009. Article (CrossRef Link)

[7] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Wsrec, "A collaborative filtering based web service recommender system," in *Proc. of IEEE International Conference on Web Services*, pp.437–444, 2009. Article (CrossRef Link)

[8] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction for web services via collaborative filtering," *IEEE International Conference on Web Services*, pp.439–446, 2007. Article (CrossRef Link)

[9] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized qos-aware web service recommendation and visualization," *IEEE Transactions on Services Computing*, 6(1), pp. 35-47, 2013. Article (CrossRef Link)

[10] Z. Zheng, H. Ma and M. R. Lyu, et al., "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, 6, pp.289-299, 2013. Article (CrossRef Link)

[11] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. on Optimization*, 20(4):1956–1982, Mar. 2010. Article (CrossRef Link)

[12] D. Zhang, Y. Hu, J. Ye, X. Li, and X. He, "Matrix completion by truncated nuclear norm regularization," in *Proc. of Conference on Computer Vision and Pattern Recognition*, pp. 2192–2199, 2012. Article (CrossRef Link)

[13] Y. Hu, D. Zhang, J. Liu, J. Ye, and X. He, "Accelerated singular value thresholding for matrix completion," *ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 298–306, New York, USA, 2012. Article (CrossRef Link)

[14] Zhang, W. Y. and S. Zhang, et al., "Combining social network and collaborative filtering for personalised manufacturing service recommendation," *International Journal of Production Research*, 51(22): 6702-6719. Article (CrossRef Link)

[15] Sun, H. F. and Z. B. Zheng, et al., "Personalized Web Service Recommendation via Normal Recovery Collaborative Filtering," *IEEE Transactions on Services Computing*, 6(4): 573-579. Article (CrossRef Link)

[16] Cao, J. and Z. Wu, et al., "Hybrid Collaborative Filtering algorithm for bidirectional Web service recommendation," *Knowledge and Information Systems*, 36(3): 607-627. Article (CrossRef Link)

[17] Wu, J. and L. Chen, et al., "Predicting Quality of Service for Selection by Neighborhood-Based Collaborative Filtering," *IEEE Transactions on Systems MAN Cybernetics-Systems*, 43(2): 428-439. Article (CrossRef Link)

[18] Deng, S. G. and L. T. Huang, et al., "Trust-Based Personalized Service Recommendation: A Network Perspective," *Journal of Computer Science and Technology*, 29(1): 69-80.

Article (CrossRef Link)

[19] A. Ramon,R. Tomas, and M. Augusto, et al., "Flexible Service Composition Based on Bundle Communication in OSGi," *KSII Transactions on Internet and Information Systems*, 6(1):116-130, 2012. Article (CrossRef Link)

[20] L. Hai, Z. Zibin, and Z. Weimin, "A Global Graph-based Approach for Transaction and QoS-aware Service Composition," *KSII Transactions on Internet and Information Systems*, 5(7):1252-1273, 2011. Article (CrossRef Link)

[21] L. Jianxin, L. Yang, and W. Jingyu, "Service Composition Based on Niching Particle Swarm Optimization in Service Overlay Networks," *KSII Transactions on Internet and Information Systems*, 6(4):1106-1127, 2012. Article (CrossRef Link)

[22] Q. Yu, "Qos-aware service selection via collaborative QoS evaluation," *World Wide Web Journal*, 1-25, 2013. Article (CrossRef Link)

Yu Lei, is a lecturer in college of computer science, Inner Mongolia University, China. His research interests include service computing.

Wang Zhili, is an associate professor in State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. His research interests include network management and communication software.

Meng Luoming, a professor, Ph.D. supervisor and Changjiang Scholar, is with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. His research interests include network management and communication software.

Qiu Xuesong, who is a professor and Ph.D. supervisor at State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. His research interests include network management and communication software.