Personalized QoS-Aware Web Service Recommendation and Visualization

Xi Chen, *Member*, *IEEE*, Zibin Zheng, *Student Member*, *IEEE*, Xudong Liu, Zicheng Huang, and Hailong Sun, *Member*, *IEEE*

Abstract—With the proliferation of Web services, effective QoS-based approach to service recommendation is becoming more and more important. Although service recommendation has been studied in recent literature, the performance of existing ones is not satisfactory, since (1) previous approaches fail to consider the QoS variance according to users' locations; and (2) previous recommender systems are all black boxes providing limited information on the performance of the service candidates. In this paper, we propose a novel collaborative filtering algorithm designed for large scale Web service recommendation. Different from previous work, our approach employs the characteristic of QoS and achieves considerable improvement on the recommendation accuracy.To help service users better understand the rationale of the recommendation and remove some of the mystery, we use a recommendation visualization technique to show how a recommendation is grouped with other choices. Comprehensive experiments are conducted using more than 1.5 million QoS records of real world Web service invocations. The experimental results show the efficiency and effectiveness of our approach.

Index Terms—Service recommendation, QoS, collaborative filtering, self-organizing map, visualization

1 Introduction

Web services are software components designed to support interoperable machine-to-machine interaction over a network. The adoption of Web services as a delivery mode in business has fostered a new paradigm shift from the development of monolithic applications to the dynamic set-up of business process. In recent years, Web services have attracted wide attentions from both industry and academia, and the number of public Web services is steadily increasing.

When implementing service-oriented applications, service engineers (also called service users) usually get a list of Web services from service brokers or search engines that meet the specific functional requirements. They need to identify the optimal one from the functionally equivalent candidates. However, it is difficult to select the best performing one, since service users usually have limited knowledge of their performances. Effective approaches to service selection and recommendation are urgently needed.

Quality-of-Service (QoS) is widely employed to represent the non-functional performance of Web services and has been considered as the key factor in service selection [33], [34], [35]. QoS is defined as a set of user-perceived properties, including response time, availability, reputation, etc. Currently, it's not practical for users to acquire QoS information by evaluating all the service candidates, since conducting real world Web service invocations is time-consuming and resource-consuming. Moreover, some QoS properties (e.g., reputation and reliability) are difficult to be evaluated, since long-duration observation and a number of invocations are required. Besides client-side evaluation, it's impractical to acquire QoS information from service providers or third-party communities, because service QoS performance is susceptible to the uncertain Internet environment and user context (e.g., user location, user network condition, etc.). Therefore, different users may observe quite different QoS performance of the same Web service, and QoS values evaluated by one user cannot be used directly by another in service selection and recommendation.

The objective of this paper is to make personalized QoS-based Web service recommendations for different users and thus help them select the optimal one among the functional equivalents. Several previous work [19], [20], [22], [29] has applied collaborative filtering (CF) to Web service recommendation. These CF-based Web service recommender systems work by collecting user observed QoS records for different Web services and matching together users who share the same information needs or same tastes. Users of a CF system share their judgments and opinions on Web services, and in return, the system provides useful personalized recommendations. However, three unsolved problems of the previous work affect the performance of current service recommender systems. The first problem is that the existing approaches fail to recognize the QoS variation with users' physical locations. After the analysis of a real world Web service dataset[1], which contains 1.5 millions service invocation results of 100 public services evaluated by users from more than twenty contries, we discover that some QoS

- *Xi Chen, Xudong Liu, Zicheng Huang, and Hailong Sun are with the Schoold of Computer Science and Engineering, Beihang University, Beijing China. E-mail:bargittachen@gmail.com; {liuxd, huangzc, sunhl}@act.buaa.edu.cn.*
- *Zibin Zheng is with the Department of Computer Science and Engineering, The Chinese University of HongKong, Shatin, Hong Kong, China. E-mail: zbzheng@cse.cuhk.edu.hk*

Manuscript received Aug.15th,2010

[1] http://www.wsdream.net

Digital Object Indentifier 10.1109/TSC.2011.35 1939-1374/11/$26.00 © 2011 IEEE

properties like response time and availability highly relate to the users' physical locations. For example, the response time of a service observed by users who are closely located with each other usually fluctuates mildly around a certain value, while it sometimes varies significantly between users far away from each other.

The second problem is the online time complexity of memory-based CF recommender systems [20], [29]. The increasing number of Web services and users will pose a great challenge to current systems. With O(mn) time complexity where m is the number of services and n the number of users, exisiting systems cannot generate recommendations for tens of thousands users in real time.

The last problem is that current Web service recommender systems are all black boxes, providing a list of ranked Web services with no transparency into the reasoning behind the recommendation results [19], [20], [22], [29], [36]. It is less likely for users to trust a recommendation when they have no knowledge of the underlying rationale. The opaque recommendation approaches prevent the acceptance of the recommended services. Herlocker et al. [5] mention that explanation capabilities is an important way of building trust in recommender systems, since users are more likely to trust a recommendation when they know the reason behind it.

To address the first two problems, we propose an innovative CF algorithm for QoS-based Web service recommendation. To address the third problem and enable an improved understanding of the Web service recommendation rationale, we provide a personalized map for browsing the recommendation results. The map explicitly shows the QoS relationships of the recommended Web services as well as the underlying structure of the QoS space by using map metaphor such as dots, areas and spatial arrangement.

The main contributions of this work are threefold:

- Firstly, we combine the model-based and memory-based CF algorithms for Web service recommendation, which significantly improves the recommendation accuracy and time complexity compared with previous service recommendation algorithms.
- Secondly, we design a visually rich interface to browse the recommended Web services, which enables a better understanding of the service performance.
- Finally, we conduct comprehensive experiments to evaluate our approach by employing real world Web service QoS dataset. More than 1.5 millions real world Web service QoS records from more than 20 countries are used in our experiments.

The remainder of this paper is organized as follows: Section 2 describes the recommendation approach in detail. Section 3 presents the method for recommendation visualization. Section 4 and 5 show the experiments of the recommendation and visualization respectively. Section 6 discusses the related work, and Section 7 concludes the paper with a summary and a description of future work.

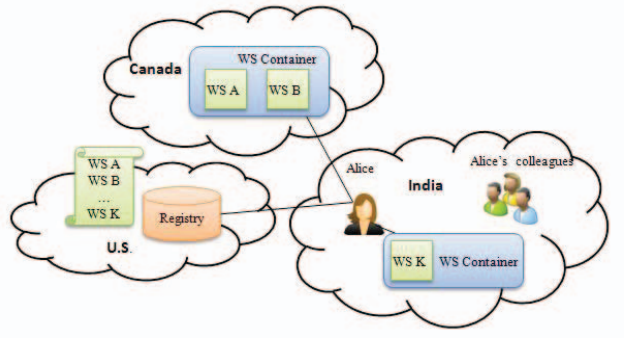


Fig. 1. Alice's situational problem.

2 The Recommendation Approach

2.1 A Motivating Scenario

In this section, we present an online service searching scenario to show the research problem of this paper. As Fig. 1 depicts, Alice is a software engineer working in India. She needs an email validation service to filter emails. After searching a service registry located in U.S., she gets a list of recommended services in ascending order of the service average response time. Alice tries the first two services provided by a Canadian company and finds that the response time is much higher than her expectation. She then realizes that the service ranking is based on the evaluation conducted by the registry in U.S., and the response time of the same service may vary greatly due to the different user context, such as user location, user network conditions, etc. Alice then turns to her colleagues in India for suggestion. They suggest her try service k provided by a local company though ranked lower in the previous recommendation list. After trying it, Alice thinks that service k has a good performance and meets her requirements.

The problem that Alice faces is to find a service that meets both functional and non-functional requirements. The current way of finding a suitable Web service is rather inefficient, since Alice needs to try the recommended services one by one. To address this challenge, we propose a more accurate approach to service recommendation with consideration of the region factor. Moreover, we try to provide a more informative and user-friendly interface for browsing the recommendation results rather than a ranked list. By this way, users are able to know more about the overall performance of the recommended services, and thus trust the recommendations.

The basic idea of our approach is that users closely located with each other are more likely to have similar service experience than those who live far away from each other. Inspired by the success of Web 2.0 websites that emphasize information sharing, collaboration and interaction, we employ the idea of user-collaboration in our Web service recommender system. Different from sharing information or knowledge on blogs or wikis, users are encouraged to share their observed Web service QoS performance with others in our recommender system. The more QoS information the user contributes, the more ac-

curate service recommendations the user can obtain, since more user characteristics can be analyzed from the user contributed information.

Based on the collected QoS records, our recommendation approach is designed as a two-phase process. In the first phase, we divide the users into different regions based on their physical locations and historical QoS experience on Web services. In the second phase, we find similar users for the current user and make QoS prediction for the unused services. Services with the best predicted QoS will be recommended to the current user. These two phases are presented in Section 2.2 and Section 2.3 respectively, and the time complexity analysis of the proposed algorithm is presented in Section 2.4.

2.2 Phase 1: Region Creation

In Web service recommender system, users usually provide QoS values on a small number of Web services. Traditional memory-based CF algorithms suffer from the sparse user-contributed dataset, since it's hard to find similar users without enough knowledge of their service experience. Different from existing methods, we employ the correlation between users' physical locations and QoS properties to solve this problem. In this paper, we focus on the QoS properties that are prone to change and can be easily obtained and objectively measured by individual users, such as response time and availability. To simplify the description of our approach, we use response time (also called round-trip time (RTT)) to describe our approach.

We assume that there are n users and m services. The relationship between users and services is denoted by an $n \times m$ matrix R. Each entry $R_{i,j}$ of the matrix represents the RTT of service j observed by user i, and $\perp$ is the symbol of no RTT value. Each user i ($i \in \{1,2,\ldots,n\}$) is associated with a row vector $R_{i,}$ representing his/her observed RTT values on different Web services. The user a ($a \in \{1,2,\ldots,n\}$) is called the active user or current user if he/she has provided some RTT records and needs service recommendations.

We define a *region* as a group of users who are closely located with each other and likely to have similar QoS profiles. Each user is a member of exactly one region. Regions need to be internally coherent, but clearly different from each other. The region creation phase is designed as a three-step process. In the first step, we put users with similar IP addresses into a small region and extract region features. In the second step, we calculate the similarity between different regions. In the last step, we aggregate highly correlated regions to form a certain number of large regions. Details of these three steps are presented in Section 2.2.1 to Section 2.2.3 respectively.

2.2.1 Region Feature Extraction

For each region, we use region center as the main feature to reflect the average performance of Web services observed by region users. Region center is defined as the median vector of all the RTT vectors associated with the region users. The element i of the center is the median RTT value of service i observed by users from the region. Median is the numeric value separating the higher half of a sample from the lower half.

Besides the average Web service quality observed by the region users, we also pay attention to the fluctuation of the service performance. From large number of QoS records, we discover that the service response time usually varies from region to region. Some services have unexpected long response time or even unavailable to some regions. Inspired by the three-sigma rule [40] which is often used to test outlier, we use similar method to distinguish services with unstable performance to different regions and regard them as region-sensitive services, which is another important region feature besides the region center.

The set of non-zero RTTs of service s, $R_{.s}=\{R_{1,s}, R_{2,s},\ldots, R_{k,s}\}, 1 \le k \le n$, collected from users of all regions is a sample from the population of service s response time. To estimate the mean μ and the standard deviation σ of the population, we use two robust measures: median and median absolute deviation (MAD) [32]. MAD is defined as the median of the absolute deviations from the sample's median.

$$\text{MAD} = median_i(|R_{i,s} - median_j(R_{j,s})|) \quad i = 1,\ldots,k,\ j = 1,\ldots,k \tag{1}$$

Based on the median and MAD, the two estimators can be calculated by:

$$\hat{\mu} = median_i(R_{i,S}) \quad i = 1,\ldots,k \tag{2}$$

$$\hat{\sigma} = MAD_i(R_{i,S}) \quad i = 1,\ldots,k \tag{3}$$

Definition1 (Region-Sensitive Service) Let $R_{.s}=\{R_{1,s}, R_{2,s},\ldots, R_{k,s}\}, 1 \le k \le n$, be the set of RTTs of service s provided by users from all regions. Service s is a sensitive service to region M *iff* $\exists R_{j,s} \in R_{.s}((R_{j,s} > \hat{\mu} + 3\hat{\sigma}) \wedge region(j) = M)$,where $\hat{\mu} = median_i(R_{.s})$, $\hat{\sigma} = MAD(R_{.s})$ and $region(u)$ function defines the region of user u.

Definition2 (Region Sensitivity) The sensitivity of region M is the fraction between the number of sensitive services in region M over the total number of services.

Definition3 (Sensitive Region) Region M is a sensitive region *iff* its region sensitivity exceeds the sensitivity threshold λ.

By the above definitions, we can identify services with drastically fluctuating response time and those regions where the fluctuation occurs, which is an important feature for service QoS prediction and recommendation. We detail how to set λ in Section 4.3.

2.2.2 Region Similarity Computation

Determining whether two regions are similar is a key step before region aggregation. The similarity of two regions M and N is measured by the similarity of their region centers m and n. Pearson Correlation Coefficient (PCC) is widely used in recommender systems to calculate the similarity of two users [2]. PCC value ranges from -1 to 1. Positive PCC value indicates that the two users have simi-

lar preferences, while negative PCC value means that the two user preferences are opposite. PCC computes the similarity between two regions M and N based on Eq.(4).

$$Sim(m,n)=\frac{\sum_{s\in S(n)\cap S(m)}(R_{m,s}-\bar{R}_{m.})\cdot(R_{n,s}-\bar{R}_{n.})}{\sqrt{\sum_{s\in S(n)\cap S(m)}(R_{m,s}-\bar{R}_{m.})^2}\sqrt{\sum_{s\in S(n)\cap S(m)}(R_{n,s}-\bar{R}_{n.})^2}}, \quad (4)$$

where $S(n)\cap S(m)$ is the set of co-invoked services by users from region M and N, $R_{m,s}$ is the RTT value of service s provided by region center m. $\bar{R}_{m.}$ and $\bar{R}_{n.}$ represent the average RTT of all the services of center m and n respectively. Because PCC only considers the RTT difference of the co-invoked services by both regions, it often overestimates the similarity of the two regions that are not similar, but happen to have a few co-invoked services with very similar RTTs [13]. Intuitively, we hypothesize that the accuracy of prediction can be improved if we add a correlation significance weighting factor that can devalue the overestimated similarity. We use the following adjusted PCC equation to calculate the similarity between two regions:

$$Sim'(m,n)=\frac{|S(m)\cap S(n)|}{|S(m)\cup S(n)|}Sim(m,n), \quad (5)$$

where $|S(m)\cup S(n)|$ is the number of Web services invoked by users either in region M or region N. The experiment which compares the result with (Eq. (5)) and without (Eq. (4)) the significance weighting is shown in Section 4.6.

2.2.3 Region Aggregation

Each region formed by users' physical locations at the outset always has a very sparse QoS dataset, since users only use a small number of Web services and provide limited QoS records. In this case, it is difficult to find similar users and predict the QoS values of the unused Web services for the active user. To solve this problem, we propose a region aggregation method based on the region features. As shown in Algorithm 1, the region aggregation approach is a bottom-up hierarchical clustering algorithm [16]. The input is a set of small regions $r_1,\ldots,r_l$. Each region consists of users with similar locations. The algorithm successively aggregates pairs of the most similar non-sensitive regions until the stopping criterion (line 16) is met. The result is stored as a list of aggregates in A.

Algorithm 1. Region Aggregation

```
   in: regions r_1,...,r_l
   out: result list A
 1: for n←1 to l - 1
 2:     for i←n + 1 to N
 3:         C[n][i].sim←SIM(r_n,r_i)
 4:         C[n][i].index←i
 5:     end for
 6:     I[n].sensitivity←ISSENSITIVE(r_n)
 7:     if I[n].sensitivity=0
 8:         then I[n].aggregate←1
 9:     else I[n].aggregate←0
10:     P[n]←priority queue for C[n] sorted on sim
11: end for
12: calculate the sensitivity and aggregate of I[l]
13: A ←[]
14: while true
15:         k_1←argmax_{k:I[k].aggregate=1} P[k].MAX().sim
16:         if k_1=null or sim< μ
17:             then return A
18:         k_2←P[k_1].MAX().index
19:         A.APPEND(<k_1, k_2>) and comput k_1 center
20:         I[k_2].aggregate←0
21:         P[k_1]←[]
22:         I[k_1].sensitivity ←ISSENSITIVE(k_1)
23:         if I[k_1].sensitivity=1
24:             then I[k_1].aggregate←0
25:                 for each i with I[i].aggregate=1
26:                     P[i].DELETE(C[i][k_1])
27:                     P[i].DELETE(C[i][k_2])
28:                 end for
29:         else
30:             for each i with I[i].aggregate=1 ∧ i≠k_1
31:                 P[i].DELETE(C[i][k_1])
32:                 P[i].DELETE(C[i][k_2])
33:                 C[i][k_1].sim←SIM(i, k_1)
34:                 P[i].INSERT(C[i][k_1])
35:                 C[k_1][i].sim←SIM(i, k_1)
36:                 P[k_1].INSERT(C[k_1][i])
37:             end for
38: end while
```

- Step 1. Initialization (lines 1-12):
  1. Compute the similarity between each two regions using Eq. (5), store the similarity and the similar region index in the similarity matrix C.
  2. Calculate the sensitivity of each region and identify whether it can be aggregated. Store the result in the indicator vector I. $I[k]$.*sensitivity* indicates whether region k is sensitive, and $I[k]$.*aggregate* indicates whether region k can be aggregated.
  3. Use a set of priority queues P to sort the rows of C in decreasing order of the similarity. Function $P[k]$.*MAX*() returns the index of the region that is most similar to region k.

- Step 2. Aggregation (lines 13-38):
  1. In each iteration, select the two most similar and non-sensitive regions from the priority queues if their similarity exceeds threshold μ, otherwise return A.
  2. Aggregate the selected two regions and store their region index in result list A. Use the smaller region index of the two as the new region index and compute the new region center. Mark the indicator vector I of the aggregated region.
  3. Calculate the sensitivity of the new region and set indicator I. If it is sensitive and cannot be

aggregated, remove this region from other regions' priority queues. Otherwise, update the elements of both priority queues and similarity matrix related to the aggregated two regions. Repeat the above three steps.

2.3 Phase 2: QoS Value Prediction

After the phase of region aggregation, thousands of users are clustered into a certain number of regions based on their physical locations and historical QoS similarities. The service experience of users in a region is represented by the region center. With the compressed QoS data, searching neighbors and making predictions for an active user can be computed quickly. Traditionally, the QoS prediction methods need to search the entire dataset [20], [29], which is rather inefficient. In our approach, similarity between the active user and users of a region is computed by the similarity between the active user and the region center. Moreover, it is more reasonable to predict the QoS value for active users based on their regions, for users in the same region are more likely to have similar QoS experience on the same Web service, especially on those region-sensitive ones. To predict the RTT value for the active user a on an unused service s, we take the following steps:

- Find the region of user a by IP address. If no appropriate region is found, the active user will be treated as a member of a new region.

- Identify whether service s is sensitive to the specific region. If it is region-sensitive, then the prediction is generated from the region center, because the service performance observed by users from this region is significantly different from others.

$$\hat{R}_{a,s} = R_{center,s} \tag{6}$$

- Otherwise, use Eq. (5) to compute the similarity between the active user and each region center that has evaluated service s, and find up to k most similar centers $\{c_1, c_2, \ldots, c_k\}$. We discuss how to choose k (also called top-k) in Section 4.4.

- If the active user's region center has the RTT value of s, the prediction is computed using Eq. (7).

$$\hat{R}_{a,s} = R_{center,s} + \frac{\sum_{j=1}^{k}(R_{c_j,s} - \overline{R}_{c_j\cdot})Sim'(a,c_j)}{\sum_{j=1}^{k} Sim'(a,c_j)}, \tag{7}$$

where $R_{c_j,s}$ is the RTT of service s provided by center c_j, and $\overline{R}_{c_j\cdot}$ is the average RTT of center c_j. The prediction is composed of two parts. One is the RTT value of the region center of the active user $R_{center,s}$,which denotes the average QoS observed by this region users. The other part is the normalized weighted sum of the deviations of the service s RTT from the average RTT observed by the k most similar neighbors.

- Otherwise, we use the service s RTTs observed by the k neighbors to compute the prediction as Eq. (8) shows. The more similar the active user a and the neighbor c_j are, the more weighting the RTT of c_j will carry in the prediction.

$$\hat{R}_{a,s} = \frac{\sum_{j=1}^{k} R_{c_j,s} Sim'(a,c_j)}{\sum_{j=1}^{k} Sim'(a,c_j)} \tag{8}$$

Note that previous CF-based Web service recommendation algorithms [20] [29] use Eq. (9), a rating aggregate method commonly adopted in recommender systems [13] [15], to predict the missing QoS value.

$$\hat{R}_{a,s} = \overline{R}_{a\cdot} + \frac{\sum_{j=1}^{k}(R_{c_j,s} - \overline{R}_{c_j\cdot})Sim'(a,c_j)}{\sum_{j=1}^{k} Sim'(a,c_j)} \tag{9}$$

However, it is not applicable in our context, since this equation is based on the idea that each user's rating range is subjective and comparatively fixed (e.g., critical users always rate items with lower ratings), whereas the range of RTT varies largely from service to service. The average RTT of all services provided by user a cannot reveal the performance of a specific Web service. Instead, we turn to the RTT profile of the region center and use its RTT of service s to predict the missing value (Eq. (7)).

2.4 Time Complexity Analysis

We discuss the worst-case time complexity of the proposed algorithm. Since there are two phases in our algorithm: the offline phase for region creation and the online phase for the QoS value prediction, we analyze their time complexity separately. We assume the input is a full matrix with n users and m services.

2.4.1 Offline Time Complexity

In Section 2.2.1, the time complexity of calculating the median and MAD of each service is O($nlogn$). For m services, the time complexity is O($mnlogn$). With MAD and median, we identify the region-sensitive services from the service perspective. Since there are at most n records for each service, the time complexity of each service is O(n) using definition 1. Therefore, the total time complexity of region-sensitive service identification is O($mnlogn + mn$) = O($mnlogn$).

The time complexity of region aggregation (see Algorithm 1) is analyzed as follows.

We assume there are l_0 regions at the beginning. Since there are at most m intersecting services for two regions, the time complexity of the region similarity is O(m) using Eq.(5), and the complexity for computing similarity matrix C is O($l_0^2 m$) (lines 1-10 of Algorithm 1).

The aggregation of two regions will execute at most l_0-1 times (lines 14-38), in case that all regions are non-sensitive, extremely correlate to each other and finally aggregate into one region. In each iteration, we first compare at most l_0-1 heads of the priority queues to find the most similar pairs (line 15). Since the number of regions that can be aggregated decreases with iteration, the real search time will be less than l_0-1 in the following itera-

tions. For the selected pair of regions, we calculate the new center and update their similar regions. Because the number of users involved in the two regions are uncertain, we use the number of all users as the upper bound and the complexity is O($mnlogn$) .The insertion and deletion of a similar region is O($logl_0$), since we employ the priority queue to sort similar regions. Thus, the time complexity of Algorithm 1 is O($l_0^2(logl_0+mnlogn)$)=O($l_0^2\ mnlogn$).

As the above steps are linearly combined, the total time complexity of the offline part is O($l_0^2\ mnlogn$).

2.4.2 Online Time Complexity

Let l_1 be the number of regions after the phase of region creation. To predict the QoS value for an active user, O(l_1) similarity calculations between the active user and region centers are needed, each of which takes O(m) time. There for, the time complexity of similarity computation is O(l_1m).

For each service the active user has not evaluated, the QoS value prediction complexity is O(l_1), because at most l_1 centers are employed in the prediction as Eq.(7) and Eq.(8) state. There are at most m services without QoS values, so the time complexity of the prediction for an active user is O(l_1m). Therefore, the time complexity for the online phase including similarity computation and missing value prediction is O(l_1m)≈O(m) (l_1 is rather small compared to m or n). Compared to the memory-based CF algorithm used in previous work with O(mn) online time-complexity, our approach is more efficient and well suited for large dataset.

3 Recommendation Visualization

Conventionally, CF-based Web service recommender systems employ the predicted QoS mainly in two ways. 1) When users query a service with specific functionality, the one with the best predicted QoS is recommended to them. 2) Top-k best-performing services are recommended to help users discover potential services. While this kind of recommendation is useful, it is not obvious to users why certain services are recommended. More than a service list ranked by predicted QoS as recommendation, we need to develop an exploratory recommendation tool that provides valuable insight into the QoS space and enables an improved understanding of the overall performance of Web services.

The QoS space visualization of all Web services on a map will reveal the rationale behind QoS-based service recommendations. QoS space visualization is more than a picture or method of computing. It transforms the information of high dimensional QoS data into a visual form enabling service users to observe, browse, and understand the information.

We draw the QoS map by two steps: dimension reduction step and map creation step. In the first step, we create a two-dimensional representation of the high dimensional QoS space by using self-organizing map (SOM), and each Web service is mapped to a unique two-dimensional coordinates. In the second step, we create a geographic-like QoS map based on the SOM training results. We detail the two drawing steps in section 3.1 and 3.2 respectively.

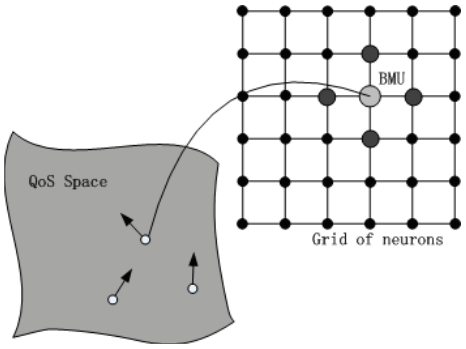


Fig. 2. Mapping QoS space to the two-dimensional output space of SOM. Each QoS vector is mapped to the BMU with closest Euclidean distance.

3.1 SOM Training

The SOM [8] is a popular unsupervised artificial neural network that has been successfully applied to a broad range of areas, such as medical engineering, document organization and speech recognition. When SOM is used for information visualization, it can be viewed as a mapping of a high dimensional input space to a lower dimensional output space (usually one or two dimensions).

The output space of SOM is a network of neurons located on a regular, usually two-dimensional grid. Each neuron is equipped with a prototype vector which has the same dimension of the input space. The neurons are connected to adjacent ones by a neighborhood relation indicating the structure of the map such as a rectangular or hexagonal lattice. After the training phase, data points close to each other in the input space are mapped onto the nearby neurons. With this topology preserving projection property, SOM is frequently employed in data survey applications to help visualize the inherent structure of high-dimensional complex data.

The principal goal of using SOM in our context is to transform the QoS data employed by CF into a two-dimensional discrete map in a topologically ordered fashion. In this context, the QoS map is to show the similarity of RTT variance of different Web services. Intuitively, the input of the SOM is the QoS matrix containing all the services (rows) and their respective QoS values provided by all users (columns). However, since the data set is rather sparse and the number of QoS values varies from service to service, the original data set cannot reveal the underlying structure of the QoS space. To address this problem, the QoS set derived from region centers is employed to train the SOM.

Let l denote the dimension of the input space (QoS data) and $q = [q_1, q_2, ..., q_l]^T$ denote an input pattern (QoS vector of a service). The prototype of neuron j is denoted by

$$\omega_j = [\omega_{j1}, \omega_{j2}, \ldots, \omega_{jl}]^T, j=1,2,\ldots, k, \qquad (10)$$

where k is the total number of neurons in the network.

The SOM is trained iteratively. In each training step, a

QoS vector q is randomly chosen from the input dataset. The Euclidean distance between q and all prototypes is calculated. The neuron with the prototype closest to q is chosen as the best-matching unit (BMU) as Fig.2 depicts. Let $i(x)$ be the index of BMU, we determine $i(x)$ by applying

$$i(x) = \arg\min_j \| q - \omega_j \|, j=1,2,\ldots, k. \quad (11)$$

Then the prototype of BMU and those of BMU's neighbors are adapted to move closer to the input vector q in the input space. Given the prototype $\omega_j(n)$ of neuron j at time n, the updated vector $\omega_j(n+1)$ of this neuron at time n+1 is defined by:

$$\omega_j(n+1) = \omega_j(n) + \eta(n)h_{j,i}(n)(q - \omega_j(n)), \quad (12)$$

where $\eta(n)$ is the learning rate function, and $h_{j,i}(n)$ is the neighborhood function which determines how strongly the neurons are connected with each other. The commonly used Gaussian neighborhood function is defined as:

$$h_{j,i}(n) = \exp(-\frac{\| r_j - r_i \|}{2\sigma^2(n)}) \quad (13)$$

,where r_j is the location of neuron j, and $\sigma(n)$is the neighborhood radius at time n. Both $\eta(n)$ and $h_{j,i}(n)$ decrease monotonically with time.

The map is usually trained in two phases [4]: a rough training phase that neurons are topologically ordered with relatively large initial neighborhood radius and learning rate; a convergence phase to fine tune the map with small initial neighborhood radius and learning rate. After the training of SOM, services with similar QoS are mapped onto the same neuron or nearby neurons. The mapping result of the QoS data reflects the QoS similarities between services.

3.2 Map Creation

The direct approach to Web service QoS map is to assign each Web service a distinct portion of the two-dimensional display area, and put services with similar QoS performance next to each other. With the training result, we first assign each service unique coordinates by randomly distributing them within the cell boundary of the corresponding neuron. Then the Voronoi diagram [21], [30] is used to form a base map in which each service corresponds to a unique polygon.

When applied to a large set of services, base map alone is insufficient and will quickly become too complex to reveal the underlying data relationships. A generalized map explicitly telling the cluster information is needed. We apply hierarchical clustering method [16] to cluster Web services based on their QoS similarity and form a generalized map by merging service polygons belonging to the same cluster. The topological preserving feature of SOM guarantees that services belong to the same cluster are usually neighboring polygons on the map.

We put Web service recommendations on the map by using the predicted QoS values. For those functionally equivalent services, the one with the best predicted QoS will be marked on the map. We also highlight the top-k best performing services to help users find potential ones. Section 5 provides the detail of the map implementation.

4 Experiments

In this section, we give a comprehensive study on the QoS prediction performance of our proposed algorithm.

4.1 Experimental Setup

We adopt a real world Web service QoS performance dataset[2] for the experiment. The dataset contains about 1.5 million Web service invocation records of 100 Web services from more than 20 countries. The RTT records are collected by 150 computer nodes from the Planet-Lab[3], which are distributed over 20 countries. For each computer node, there are 100 RTT profiles, and each profile contains the RTT records of 100 services. We randomly extract 20 profiles from each node, and obtain 3000 users with RTTs ranging from 2 to 31407 milliseconds.

We divide the 3000 users into two groups, one as training users and the rest as active (test) users. To simulate the real situation, we randomly remove a certain number of RTT records of the training users to obtain a sparse training matrix. We also remove some records of the active users, since active users usually only have invoked a small number of Web services in reality.

To evaluate the prediction performance, we compare our approach with user-based CF algorithm using PCC (UPCC) [20], item-based CF algorithm using PCC (IPCC) [11], and WSRec [29] which combines UPCC and IPCC.

We use Mean Absolute Error (MAE), the well-known statistical accuracy metric, to measure the prediction accuracy. MAE is the average absolute deviation of predictions to the ground truth data. For all test services and test users:

$$MAE = \frac{\sum_{u,s} | R_{u,s} - \hat{R}_{u,s} |}{L}, \quad (14)$$

where $R_{u,s}$ denotes the actual RTT of Web service s observed by user u, $\hat{R}_{u,s}$ denotes the predicted RTT value, and L denotes the number of predicted values. Smaller MAE indicates better prediction accuracy.

4.2 Prediction Evaluation

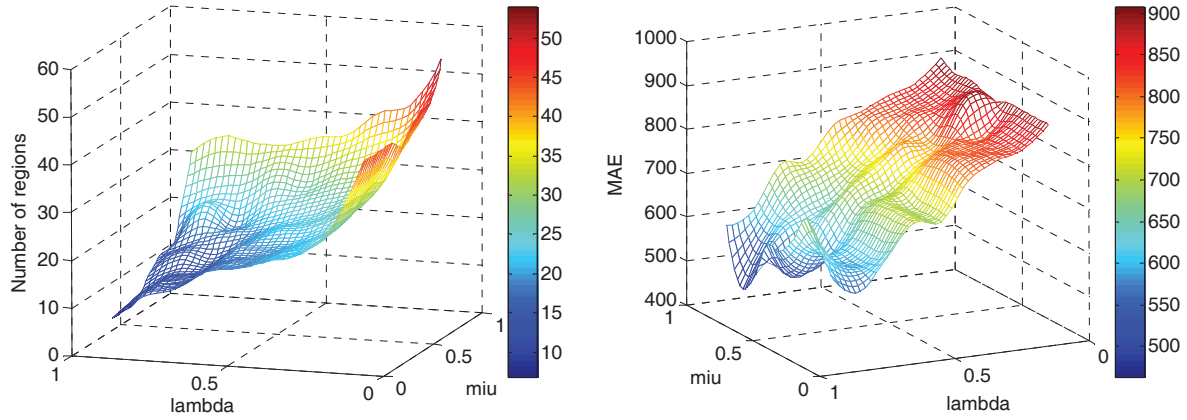
In this experiment, we randomly remove 90% and 80% RTTs of the initial training matrix to generate two sparse matrices with density 0.1 and 0.2 respectively. We vary the number of RTT values given by active users from 10, 20 to 30, and name them given 10, given 20, and given 30 respectively. The removed records of active users are used to study the prediction accuracy. In this experiment, we set μ=0.3, λ=0.8, top-k=10. To get a reliable error estimate, we use 10 times 10-fold cross-validation [31] to evaluate the prediction performance and report the average MAE value.

Table 1 shows the prediction performance of different

[2] http://www.wsdream.net
[3] http://www.planet-lab.org

TABLE 1 Prediction Performance Comparison

Method	Density = 0.1			Density = 0.2		
	Given 10	Given 20	Given 30	Given 10	Given 20	Given 30
IPCC	1179.32	1170.73	1160.45	1104.02	1094.63	1086.08
UPCC	1280.95	1145.80	1085.85	1167.84	846.54	674.32
WSRec	976.01	805.60	772.34	968.69	788.37	742.15
Our Method	**643.26**	**622.02**	**617.20**	**466.12**	**457.21**	**451.88**



(a) (b)

Fig. 3. Impact of thresholds λ and μ. (a) Impact on the number of regions. (b) Impact on the prediction performance (MAE).

methods employing the 0.1 and 0.2 density training matrix. We observe that our method significantly improves the prediction accuracy, and outperforms others consistently. The performance of UPCC, WSRec and our approach enhances significantly with the increase of matrix density as well as the number of QoS values provided by active users (given number). On the contrary, there is only a slight improvement of IPCC. The original idea of IPCC is to match items with similar user ratings and combine them to recommendations. Apparently, it is not appropriate to apply this idea to our context, because even services provided by the same company are hardly to have similar response times to different users.

4.3 Impact of λ and μ

The two thresholds λ and μ in the phase of region creation play a very important role in determining the number of regions and thus impact the final performance of our approach. As shown in Algorithm 1, only those regions with similarity higher than μ and sensitivity less than λ are able to be aggregated. In this experiment, we study the impact of λ and μ on a sparse matrix with 2700 training users and 300 active users. We set density=0.2, given=10 and employ all the neighbors with positive PCC for QoS prediction. We vary the two thresholds λ and μ both from 0.1 to 0.9 with a step of 0.1. Fig. 3 shows how λ and μ affect the number of regions and the final performance. It shows that lower μ and higher λ result in fewer regions, but fewer regions does not necessarily mean better prediction accuracy. For this dataset, better prediction accuracy is achieved with higher λ and μ.

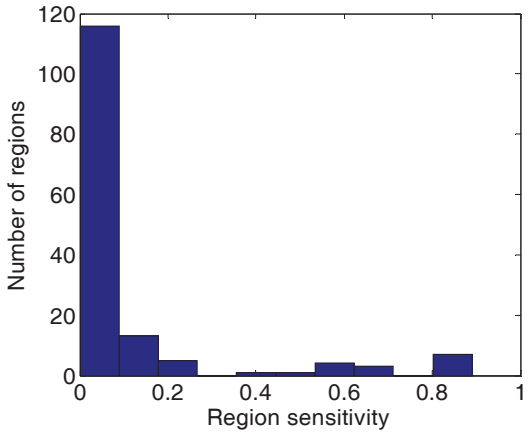


Fig. 4. The distribution of region sensitivity.

Note that the optimal value of λ is related to the sensitivity of the original regions at the outset. Fig. 4 shows the distribution of the region sensitivity before aggregation. It shows that the sensitivity of most regions (81.3%) is less than 0.1, while the sensitivity of a few regions (4.67%) is around 0.8. Higher λ and μ allow very similar regions with high sensitivity to be aggregated and achieve good performance in this experiment.

Fig. 5 (a) shows the relation between μ and prediction accuracy with training matrix density 0.2, 0.5 and 1. We employ all the neighbors with positive PCC values for QoS prediction and set λ=1, so that we do not consider the factor of sensitivity in region aggregation. Similarity becomes the single factor. Obviously, for denser matrix,

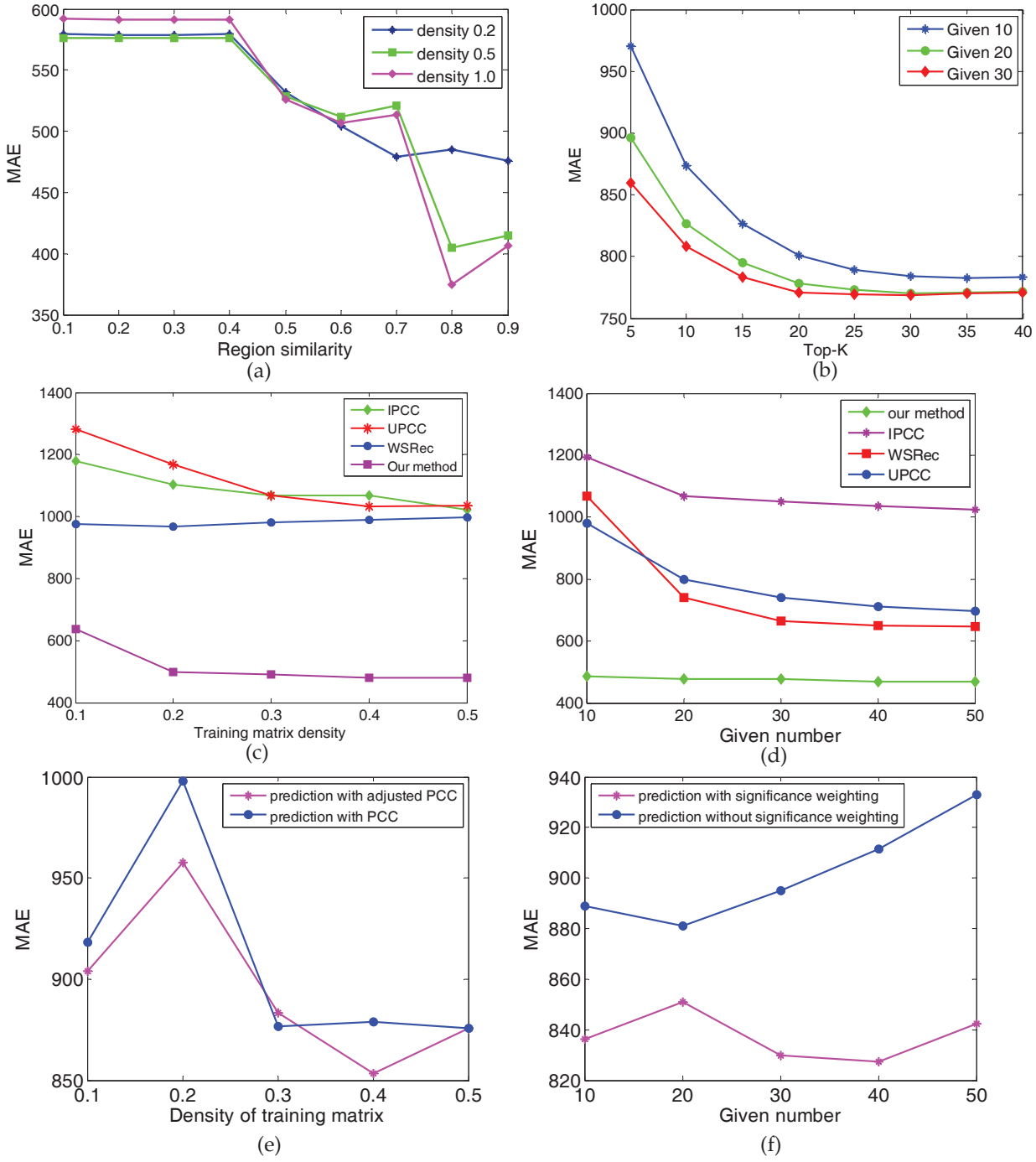


Fig. 5. Parameters impact on prediction performance (MAE). (a) Impact of region similarity μ (b) Impact of Top-K (c) Impact of the training matrix density (d) Impact of the given number (e) Significance weighting Impact with changing density (f) Significance weighting Impact with changing given number

with higher μ we obtain a set of coherent regions and better prediction is obtained.

4.4 Impact of *Top-k*

Top-k determines how many neighbors are employed in the phase of QoS prediction which relates to the prediction accuracy. We employ a training matrix of density 0.3, and set λ=0.2, μ=0.8. After the model building phase, we obtain 39 regions. To study the impact of neighborhood size, we vary top-k from 5 to 40 with a step of 5. Fig. 5 (b) shows the result with given number from 10 to 30. The trends of the three curves are the same that MAE decreases sharply with an increasing neighborhood size at the beginning, and then stays around a certain value. As top-k grows, more regions are selected in the QoS prediction, while those later added regions are usually less similar and make little contribution to the final result.

4.5 Data Sparseness

This experiment investigates the impact of data sparse-

ness on the prediction accuracy. We examine the impact from two aspects: the density of training matrix and the number of QoS values given by active users (given number). We divide the experiment into two parts and use 10 times 10 fold cross-validation to assess the prediction results and report the average MAE.

We first study the impact of training matrix density. We vary the density of the training matrix from 0.1 to 0.5 with a step of 0.1, and set given=10. Fig. 5 (c) shows the experimental results. It shows that: (1) With the increase of the training matrix density, the performance of IPCC, UPCC and our method enhances indicating that better prediction is achieved with more QoS data. WSRec is not sensitive to the data sparseness, and it stays around a certain value. (2) Our method outperforms others consistently.

To study the impact of given number on the prediction results, we employ the training matrix with density 0.3 and vary the given number from 10 to 50 with a step of 10. Fig. 5 (d) shows the experimental results. It shows that the prediction performance of IPCC, UPCC and WSRec improves greatly with the growth of the given number, while our method enhances slightly. This observation indicates that our method is not sensitive to the value of given number. It can achieve good prediction result even when the given number is rather small.

4.6 Significance Weighting

Significance weighting factor is added to devalue similarity weights that are based on a small number of co-invoked Web services. To study the impact of this factor, we implement two versions of the algorithm, one with the significance weighting (Eq. 5) and the other without (Eq. 4). Since the overestimation of the similarity occurs when the active user and training users have few co-invoked services, we study the impact by varying the number of QoS provided by both the training users (training matrix density) and active users (given number) with two experiments.

In the first experiment, we employ 2700 training users, 300 active users, set given=5, λ=0.2, μ=0.3, top-k=20. We vary the density of the training matrix from 0.1 to 0.5 with a step of 0.1. As Fig. 5 (e) shows, applying the significance weighting increases the accuracy of the prediction algorithm in most cases.

The other experiment is carried out with the same number of training users and active users. We set the density of the training matrix 0.05, λ=0.2, μ=0.3, top-k=10. We vary the given number from 10 to 50 by a step of 10. Fig. 5 (f) shows that the algorithm with the significance weighting consistently increases the accuracy of the prediction by a relatively large amount. As the given number increases, the gap between the two becomes more obvious. This is because with a sparse training matrix (density=0.05) and high given number, it is common for the active user to have neighbors that were based on tiny samples (two or three co-invoked services with similar QoS). In this case, overestimation of the similarity by Eq. (4) frequently leads to poor prediction accuracy.

5 A Map Display for Recommendation

In this section, we demonstrate how to create a map showing the similarity of RTT variance of Web services, and how to put personalized Web service recommendations on the map for an active user. We use 2700 training users and set given=10, density=0.5. After the region aggregation phase, 17 regions are formed. The input of SOM is a 100×17 RTT matrix containing 100 services (rows) and their respective performance on 17 regions (columns). Each Web service's QoS (row) is an input vector. We train an SOM with neurons arranged in a 60×80 hexagonal lattice. The prototypes of SOM are randomly initialized, and Gaussian function is adopted as the neighborhood function (see Eq. (13)). We train the SOM in two phases: a rough training phase with initial neighborhood width 15 and learning rate 0.05; a fine tuning phase with initial neighborhood width 2 and learning rate 0.01. The training lengths of the two phases are 500 and 3000 epochs respectively. The learning rate decreases linearly to zero during the training.

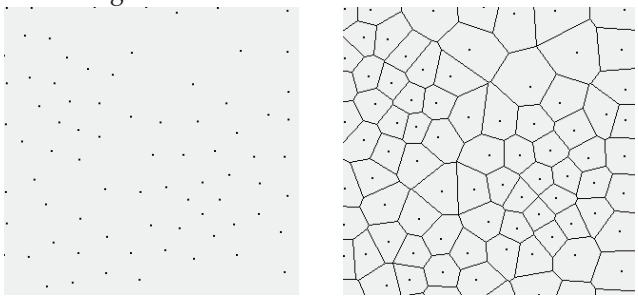


Fig. 6. Base map creation. (a) 2D locations of services derived from SOM training. (b) Voronoi diagram of services based on the 2D locations.

When the training process is completed, each service is mapped on to a neuron. We assign unique coordinates to each service by randomly distributing them within the boundaries of the corresponding neuron cell (see Fig. 6 (a)). To create a geographic map, each point is assigned to a distinct portion of the map display by forming a Voronoi diagram (see Fig. 6(b)). After that, we adopt the hierarchical clustering to the services based on their QoS similarities and obtained 42 clusters. We simplified the base map by merging the neighboring polygons if they are in the same cluster. By this way, we form a generalized map highlighting the underlying structure of the QoS space.

Labeling individual service is an integral part of the map creation. The goal is to help users identify the potential services with optimal QoS values. We use different label styles to mark services showing how strongly we recommend them. For example, the top 10 best performing services are labeled with 12-point boldface; services with good predicted QoS are labeled with 8-point non-boldface; while dots indicate those services with poor predicted QoS values. Fig. 7 shows the final map for Web service recommendations.

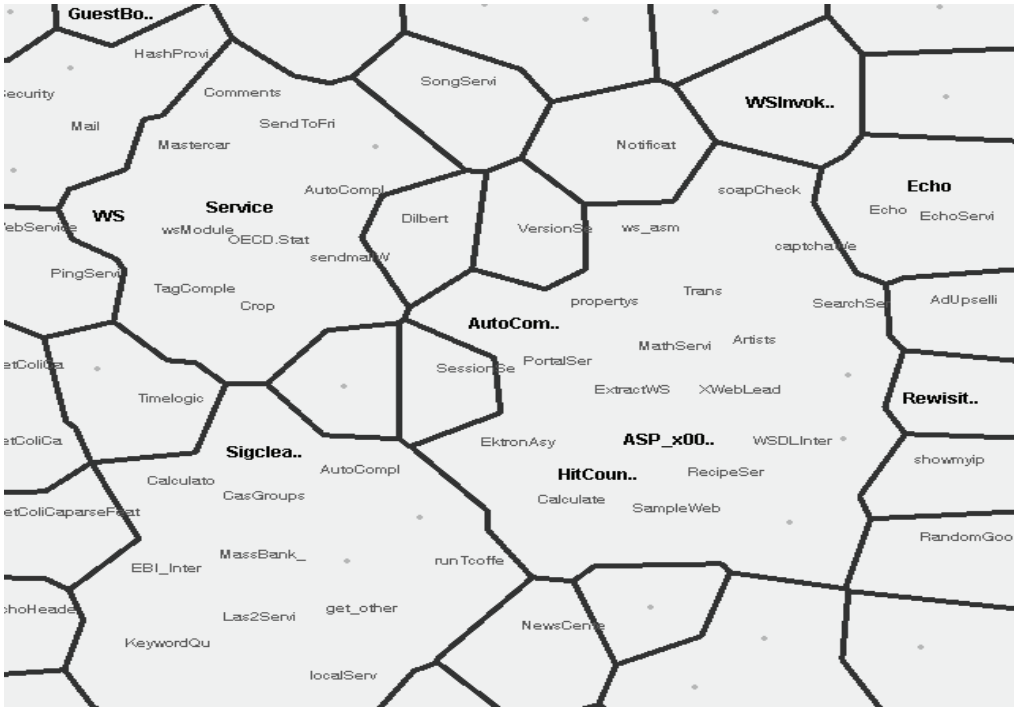


Fig. 7. Final map with service recommendations.

	RTT	Cost	Reputation
WS_1	1200	10	1
WS_2	1000	20	2
…	…	…	…
WS_{100}	800	50	5

Fig. 8. Region center matrix.

The map can also be created for one region to show the similarity of Web services based on a set of QoS properties. For example, we can employ three QoS properties: RTT, cost and reputation. RTT and cost are quantitative properties, while reputation is qualitative (usually 1-5 starts). Each region center is a matrix composed of three column vector (RTT, Cost, Reputation) as Fig. 8 shows. Since different properties have different ranges, we first normalize each of them to [0, 1] by the following steps: 1) find the minimum R_{min} and maximum R_{max} value of the property; 2) For each original value R submitted by the user, the normalized value R' is calculated by Eq. (15).

$$R' = \frac{R - R_{\min}}{R_{\max} - R_{\min}} \tag{15}$$

With the normalized values, each property will have equal weights in the SOM training. The map creation process is the same, and we can obtain a map reflecting the Web service QoS similarity of a specific region.

6 Related Work

In this section, we discuss related work regarding CF and Web service recommendation.

6.1 Collaborative Filtering

Collaborative Filtering is firstly proposed by Rich [18] and widely used in commercial recommender systems, such as Amazon.com [11]. The basic idea of CF is to predict and recommend the potential favorite items for a particular user by leveraging rating data collected from similar users. Formally, a CF domain consists of n users $\{u_1, u_2, \ldots, u_n\}$, m items $\{i_1, i_2, \ldots, i_m\}$, and users' ratings on items, which is often denoted by a user-item matrix. Each entry $r_{x,y}$ ($1 \le x \le n$, $1 \le y \le m$) in this matrix represents user x' s rating on item y. The rating score usually has a fixed range, like 1-5. Since users only express their preference on a small number of items, the matrix is very sparse in reality.

Essentially, CF is based on processing the user-item matrix. Breese et al. [2] divided the CF algorithms into two broad classes: memory-based algorithms and model-based algorithms.

Memory-based algorithms such as user-based KNN [13] [15] use the entire user-item matrix when computing recommendations. These algorithms are easy to implement, require little or no training cost, and can easily take new users' ratings into account. However, memory-based algorithms cannot cope well with large number of users and items, since their online performance is often slow.

Alternatively, model-based algorithms, such as K-means clustering [26], Bayesian model [3], etc. learn the model from the dataset using statistical and machine learning techniques. These model-based algorithms can quickly generate recommendations and achieve good online performance. However, the model must be performed anew when new users or items are added to the matrix.

6.2 Web Service Recommendation

Web service discovery is a hot topic which plays a crucial role in the area of services computing [33]. Some syntactic and semantic based Web service search engines have been proposed in recent literature. Dong et al. [37] found that the traditional key word-based Web service search was insufficient, and they provided a similarity search algorithm for Web services underlying the Woogle search engine. Liu et al. [38] investigated the similarity measurement of Web services and designed a graph-based search model to find Web services with similar operations. Recommendation techniques have been used in recent research projects to enhance Web service discovery. Mehta et al. [14] found that semantics and syntax were inadequate to discover a service that meets user requirements. They added two more dimensions of service description: quality and usage pattern. Based on this service description, they propose the service mediation architecture. Blake and Nowlan [1] computed a Web service recommendation score by matching strings collected from the user's operational sessions and the description of the Web services. Based on this score, they judged whether a user is interested in the service. Maamar et al. [12] proposed a model for the context of Web service interactions and highlighted the resource on which the Web service performed. Zhao et al. [17] provided a way to model services and their linkages by semantic algorithm. Based on the input keywords, users can get a set of recommendations with linkages to the query. Previous work mainly focused on providing a mechanism to formalize users' preference, resource and the description of Web services, and recommendations are generated based on the predefined semantic models. Different from these methods, our recommendations are generated by mining the QoS records that are automatically collected from interactions between users and services.

Limited work has been done to apply CF to Web service recommendation. Shao et al. [20] proposed a user-based CF algorithm to predict QoS values. Zheng et al. [29] combined the user-based and item-based CF algorithm to recommend Web services. However, since neither of the two approaches recognized the different characteristic between Web service QoS and user ratings, the prediction accuracy of these methods was unsatisfactory. Sreenath and Singh [22] and Rong et al. [19] applied the idea of CF in their systems, and used MovieLens data [15] for experimental analysis. However, using the movie dataset to study Web service recommendation is not convincing.

Different from these existing methods, which suffer from low prediction accuracy, we propose an effective CF algorithm for Web service recommendation with consideration of the region factor. Comprehensive experiments conducted with real QoS records show that our method outperforms others consistently.

There are several SOM-based methods to visualize data structure. U-Matrix [25] is the most popular one that displays the local distance structures of the input vectors. U*-Matrix [24], an enhancement of U-Matrix, combines the density and distance information for visualization. Color assignment is also employed to show the approximate cluster structures [6] [9]. To exploit data topology in visualization, Tasdemir and Merényi [23] introduced a weighted Delaunay triangulation. Different from our approach which clusters the Web services to form a generalized map, data clusters can be computed by applying clustering techniques to the trained prototypes, and clusters can be visualized on top of the map [21] [27].

7 Conclusion and Future Work

In this paper, we have presented an innovative approach to Web service recommendation and visualization. Different from previous work, our algorithm employs the characteristic of QoS by clustering users into different regions. Based on the region feature, a refined nearest-neighbor algorithm is proposed to generate QoS prediction. The final service recommendations are put on a map to reveal the underlying structure of QoS space and help users accept the recommendations. Experimental results show that our approach significantly improves the prediction accuracy than the existing methods regardless of the sparseness of the training matrix. We also demonstrate that the online time complexity of our approach is better than the traditional CF algorithms.

In this paper, our recommendation approach considered the correlation between QoS records and users' physical locations by using IP addresses, which has achieved good prediction performance. In some cases, however, users in the same physical locations may observe different QoS performance of the same Web service. Besides the user physical location, we will investigate more contextual information that influences the client-side QoS performance, such as the workload of the servers, network conditions and the activities that users carry out with Web services (e.g., Web services are used alone or in composition). More investigations on the distribution of RTT and the correlation between different QoS properties such as RTT and availability will be conducted in our future work.

For the visualization of the recommendation results, we plan to add more user interactions such as searching Web services on the QoS map, zooming in and zooming out. Graphic map like google map will be combined to help users navigate their similar users and Web service providers on the map.

User acceptance rate of the recommendation is a key indicator of the effectiveness of the recommender system. We will collect more user feedbacks of the recommendation to help improve the prediction accuracy of our Web service recommendation algorithm.

Acknowledgment

This research is partly supported by China 863 program (No.2007AA010301 & 2009AA01Z419), and the National Natural Science Foundation of China (No. 60731160632 & 60903149).

REFERENCES

[1] M.B. Blake, and M.F. Nowlan. "A Web Service Recommender System Using Enhanced Syntactical Matching," *Proceedings of International Conference on Web Services*, Salt Lake City, Utah, USA, pp.575- 582, 2007

[2] J.S. Breese, D. Heckerman, and C. Kadie. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, WI, USA, pp. 43-52, 1998.

[3] Y.H. Chen, and E.I. George. "A Bayesian Model for Collaborative Filtering," *Proceedings of the Seventh Int'l Workshop Artificial Intelligence and Statistics.* http://www.stat.wharton.upenn.edu/~edgeorge/Research_papers/Bcollab.pdf. 1999.

[4] S. Haykin. *Neural Networks: A Comprehensive Foundation*, Second Edition, Prentice-Hall, Inc,New Jersey,1999

[5] J.L. Herlocker, J.A. Konstan, and J. Riedl. "Explaining Collaborative Filtering Recommendations," *Proceedings of the ACM conference on Computer supported cooperative work*, Philadelphia, Pennsylvania, USA, pp. 241-250, 2000.

[6] J.Himberg. "A SOM Based Cluster Visualization and its Application for False Coloring," *Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks*, Como, Italy, vol.3, pp.587-592, 2000, doi:10.1109/IJCNN.2000.861379

[7] Hsu, and S.K. Halgamuge. "Class Structure Visualization with Semi-supervised Growing Self-organizing Maps," *Neurocomputing*, vol. 71, pp. 3124-3130, 2008.

[8] T. Kohonen. "The Self-organizing Map," *Proceedings of the IEEE*, vol.78, no.9, pp.1464-1480, 1990.

[9] S. Kaski, J. Venna, and T. Kohonen. "Coloring that Reveals High-dimensional Structures in Data," *Proccedings of the International Conference on Neural Information Processing*, Perth, WA, vol. 2, pp.729-734, 1999.

[10] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R, Gordan, and J.Riedl. "GroupLens: Applying Collaborative Filtering to Usenet News," *Communications of the ACM*, vol.40, no.3, pp.77-87, 1997.

[11] G.Linden, B. Smith, and J. York. "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," *IEEE Internet Computing*, vol. 7, no.1, pp. 76-80, 2003.

[12] Z. Maamar, S.K. Mostefaoui, and Q.H. Mahmoud. "Context for Personalized Web Services," *Proceedings of the 38th Annual Hawaii International Conference*, Hawaii, USA, pp. 166b-166b. 2005.

[13] M.R. McLaughlin and J.L. Herlocker. "A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience," *Proceedings of the 27th annual international ACM SIGIR conference*, Sheffield, UK, pp. 329-336, 2004.

[14] B.Mehta, C. Niederee, A. Stewart, C. Muscogiuri, and E.J. Neuhold. "An Architecture for Recommendation Based Service Mediation," *Semantics of a Networked World*, vol.3226, pp.250-262, 2004.

[15] B.N. Miller, I.Albert, S.K. Lam, J. A. Konstan, and J. Riedl. "MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System," *Proceedings of ACM 2003 International Conference on Intelligent User Interfaces*, Miami, Florida, USA, pp. 263-266, 2003.

[16] C.D. Mining, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval,* Cambridge University Press, Cambridge, England, 2009.

[17] C. Zhao, C. Ma, Jing Zhang, Jun Zhang, L. Yi, and X. Mao. "HyperService: Linking and Exploring Services on the Web," *Proceedings of International Conference on Web Services,* Miami, FL, USA, pp. 17-24, 2010.

[18] E. Rich. "User modeling via Stereotypes," *Cognitive Science*, vol.3, no.4, pp.329-354 ,1979.

[19] W. Rong, K. Liu, and L. Liang. "Personalized Web Service Ranking via User Group combining Association Rule," *Proceedings of International Conference on Web Services*, Los Angeles, CA, USA, pp.445-452, 2009.

[20] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei. "Personalized QoS Prediction for Web Services via Collaborative Filtering," *Proceedings of International Conference on Web Services*, Salt Lake City, Utah, USA, pp.439-446, 2007.

[21] A. Skupin. "A Cartographic Approach to Visualizing Conference Abstracts," *Computer Graphics and Applications*, vol.22, no.1, pp.50-58, 2002.

[22] R.M. Sreenath, and M.P. Singh. "Agent-based Service Selection," *Journal of Web Semantics*. vol.1, no.3, pp. 261–279, 2003. doi:10.1016/j.websem.2003.11.006.

[23] K.Tasdemir, and E. Merényi, E. "Exploiting Data Topology in Visualization and Clustering of Self-Organizing Maps," *IEEE Transactions on Neural Networks*, vol.20, no.4, pp.549-562, 2009.

[24] A.Ultsch. "U*-Matrix: A Tool to Visualize Clusters in High Dimensional Data," Technical Report 36, CS Department, Philipps-University Marburg, Germany, 2004.

[25] A.Ultsch, and H.P. Siemon. "Kohonen's Self-organizing Feature Maps for Exploratory Data Analysis," *Proceedings of International Neural Networks Conference*, pp.305-308, 1990.

[26] L.H. Ungar and D.P. Foster. "Clustering Methods for Collaborative Filtering," *Proceedings of the AAAI Workshop on Recommendation Systems*, 1998.

[27] J. Vesanto and E. Alhoniemi. "Clustering of the Self-Organizing Map," *IEEE Transactions on Neural Networks*, vol.11, no.3, pp.586-600, 2000.

[28] J.Zhang, H. Shi, Y. Zhang. "Self-Organizing Map Methodology and Google Maps Services for Geographical Epidemiology Mapping," *Proceedings of Digital Image Computing: Techniques and Applications*, Melbourne, VIC, pp.229-235, 2009, doi:10.1109/DICTA.2009.46.

[29] Z. Zheng, H. Ma, M.R. Lyu, and I. King. "WSRec: A Collaborative Filtering Based Web Service Recommendation System," *Proceedings of International Conference on Web Services*, Los Angeles, CA, USA, pp. 437-444, 2009.

[30] F. Aurenhammer. "Voronoi diagrams—A Survey of a fundamental Geometric Data Structure," ACM Computing Surveys, vol.23, no.3, pp.345-405, 1991.

[31] I.H.Witten, and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*, Second Edition. Elsevier Inc. 2005.

[32] P.J.Rousseeuw, and C. Croux. "Alternatives to the Median Absolute Deviation," *Journal of the American Statistical Association*, vol.88, no.424, pp.1273-1283, 1993.

[33] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Springer and Tsinghua University Press, 2007.

[34] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient Algorithms for Web Services Selection with End-to-end QoS Constraints," *ACM Transactions on the Web*, vol. 1, no. 1, pp. 1–26, 2007.

[35] S. Rosario, A. Benveniste, S. Haar, and C. Jard, "Probabilistic QoS and Soft Contracts for Transaction-based Web Services Orchestrations,"*IEEE Transactions on Services Computing*, vol. 1, no. 4, pp. 187–200,2008.

[36] X.Chen, X.Liu, Z. huang, and H. Sun, "RegionKNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation", *Proceedings of International Conference on Web Services*, Miami, FL, USA, pp.9-16, 2010.

[37] X. Dong, A. Halevy, J. Madhavan, E.Nemes, and J. Zhang, "Similarity Search for Web Services," *Proceedings of the 30th International Conference on Very Large Data Bases*, Toronto, CA, pp.372-383, 2004.

[38] X. Liu, G. Huang, and H. Mei, "Discovering Homogeneous Web Service Community in the User-Centric Web Environment," *IEEE Transactions on Services Computing*, vol. 2, no.2, pp. 167-181, 2009.

[39] E.M. Maximilien and M.P. Singh, "A Framework and Ontology for Dynamic Web Services Selection," I*EEE Internet Computing*, vol. 8, no. 5, pp. 84-93, 2004.

[40] 68-95-99.7 rule, http://en.wikipedia.org/wiki/68-95-99.7_rule.

Xi Chen is a master student in the School of Computer Science and Engineering, Beihang University, Beijing, China. She receieved her B.S. degree in Information and Computing science from Beijing Information Technology Institute, Beijing, China, in 2008. She received Google Excellence Scholarship 2010. Her research interests include services computing, cloud computing and data mining.

Zibin Zheng received his B.Eng. degree and M.Phil. degree in Computer Science from the Sun Yat-sen University, Guangzhou, China, in 2005 and 2007, respectively. He is currently a Ph.D. candidate in the department of Computer Science and Engineering, The Chinese University of Hong Kong. He received SIGSOFT Distringuish Paper Award at ICSE'2010, Best Student Paper Award at ICWS'2010, and IBM Ph.D. Fellowship Award 2010-2011. He served as program committee member of IEEE CLOUD'2009 and CLOUDCOMPUTING'2010 and served as reviewer for international journal and conferences, including TSE, TPDS, TSC, JSS, DSN, ICEBE, ISSRE, KDD, SCC, WSDM, WWW, etc. His research interests include service computing, software reliability engineering, and cloud computing.

Xudong Liu was born in 1965. He received his Ph.D. in Computer Application Technology from Beihang University, Beijing, China. He is a professor and doctoral supervisor at Beihang University. His research interests mainly include middleware technology and application, service oriented computing, trusted network computing and network software development.

Zicheng Huang is currently working toward the Ph.D. in the School of Computer Science and Engineering, Beihang University, Beijing, China. He received his B.S. degree from Beihang University in 2004. His research interests are in the areas of services computing, business process management and social computing.

Hailong Sun is an Assistant Professor in the School of Computer Science and Engineering, Beihang University, Beijing, China. He received his Ph.D. in Computer Software and Theory from Beihang University in 2008, and B.S. degree in Computer Science from Beijing Jiaotong University in 2001. His research interests include services computing, peer-to-peer computing, grid/cloud computing and distributed systems. He is a member of IEEE and ACM.