Chapter 22
QoS-Aware Web Service Recommendation via Collaborative Filtering

Xi Chen, Zibin Zheng and Michael R. Lyu

Abstract With the increasing number of Web services on the Internet, selecting appropriate services to build one's application becomes a nontrivial issue. When searching Web services, users are often overwhelmed by a bunch of candidates with similar functionalities. Quality-of-Service (QoS), the non-functional characteristics of Web services, has become an important factor to distinguish the functionally equivalent ones. In this paper, we introduce two collaborative filtering based Web service recommendation approaches to help users select Web service with optimal QoS performance. The basic idea is to leverage user experience provided by similar users and generate recommendation for the target user. Experiments with large scale real world Web services show the effectiveness and efficiency of the two approaches.

22.1 Introduction

Web service, a method of communication between two machines over a network, has been widely adopted as a delivery mode in both industry and academia. This adoption has fostered a new paradigm shift from development of monolithic application to the dynamic set-up of business process. The increasing usage of Web services on the World Wide Web calls for effective recommendation techniques, which help end

X. Chen
Schlumberger Technologies (Beijing) Ltd., Beijing, China
e-mail: bargittachen@gmail.com

Z. Zheng (✉) · M. R. Lyu
Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong, China
e-mail: zbzheng@cse.cuhk.edu.hk

M. R. Lyu
e-mail: lyu@cse.cuhk.edu.hk

A. Bouguettaya et al. (eds.), *Web Services Foundations*,
DOI: 10.1007/978-1-4614-7518-7_22,

users choose the optimal Web service from a large number of functionally equivalent candidates.

In services computing, QoS is a set of properties describing the non-functional characteristics of Web services, such as price, response time, reliability, etc. Some QoS properties have relatively constant value, e.g., the published pricing model of Amazon Web Service (AWS), while other properties like response time vary seriously from user to user, influenced by the unpredictable Internet connections and heterogeneous environments. In this chapter, 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.

QoS plays an important role in service selection and recommendation [37, 36]. However, it is impractical for users to acquire QoS information by evaluating all the service candidates by themselves. Conducting real world Web service invocation is time-consuming and resource-consuming. Moreover, measuring some QoS properties (e.g., *reliability*) requires long time observation and large number of invocations. Besides client-side evaluation, acquiring QoS information from service providers may not be applicable, because QoS performance is susceptible to the uncertain Internet environment and user context (e.g., user location, user network condition, etc.). Therefore, QoS values evaluated by one user cannot be used directly by another in service selection and recommendation.

To make personalized QoS-aware service recommendation to different users, we introduce two collaborative filtering (CF) based Web service recommendation algorithms in this chapter. Our Web service recommender system collects user observed QoS records and matches together users who share the same information needs or same tastes [10]. Users of our recommender system share their observed QoS performance of Web services, and in return, the system provides accurate personalized service recommendations for them. Section 22.2 and Sect. 22.3 present our proposed recommendation approaches; Sect. 22.4 shows our large scale real world experiments; Sect. 22.5 discusses related work, and Sect. 22.6 concludes our work.

22.2 WSRec: A Neighborhood-Based Web Service Recommendation Algorithm

WSRec employs the concept of *user-collaboration* for Web service QoS information sharing between service users. Similar to sharing videos on *YouTube* or knowledge on *Wikipedia*, service users are encouraged to contribute their past Web service user experience to the system. We assume that users are trustworthy, and all submitted records are correct. The more QoS records users contribute, the more accurate the recommendation will be. Figure 22.1 shows the architecture of WSRec system, which includes the following procedures:

- A service user submits Web service QoS data to the centralized server of WSRec. Users who submit QoS records to WSRec are called *training users*. Users who

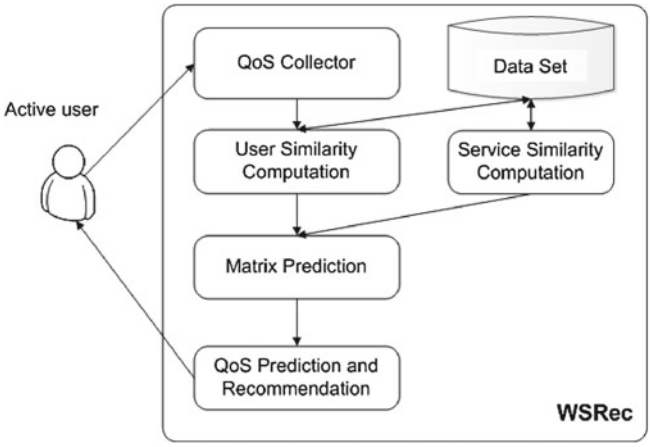

Fig. 22.1 Procedures of QoS value prediction

Table 22.1 An illustration of response time dataset

User	Service 1	Service 2	Service 3	Service 4	Service 5	Service 6	Service 7
Amy	5000 ms	?	2000 ms	?	?	?	2800 ms
Bob	600 ms	3300 ms	?	3300 ms	2000 ms	?	?
Carol	650 ms	2600 ms	200 ms	?	?	?	?
David	600 ms	2500 ms	2000 ms	5000 ms	?	2000 ms	?

require Web service recommendation are called *active users*. Table 22.1 shows an example of the recommendation system data set. There are four users and seven services in the data set. Each user provides some QoS values (response time) they observed, and ? indicates that the user does not use the service.

- WSRec matches the active user with existing training users to find similar users and Web services with similar QoS (details will be introduced in Sect. 22.2.1).
- WSRec predicts QoS values of candidate Web services for the active user (details will be introduced in Sect. 22.2.2).
- WSRec makes Web service recommendation based on the predicted QoS values of Web services (details will be discussed in Sect. 22.2.3).
- The active user receives the predicted QoS values as well as the recommendation results, which can be employed to assist decision making (e.g., service selection, service composition, service ranking, etc.).

22.2.1 Similarity Computation

Similarity computation is used to find users with similar experience as well as Web services with similar QoS in the WSRec system.

22.2.1.1 Pearson Correlation Coefficient

Given a recommender system consisting of m training users and n Web services, the relationship between users and Web services can be denoted by an $m \times n$ user-item matrix. An entry in this matrix $r_{u,i}$ represents a vector of QoS values (e.g., response time, failure rate, etc.) observed by user u of Web service i. If user u has never used Web service i before, $r_{u,i} = null$.

Pearson Correlation Coefficient (PCC) is widely used to measure user similarity in recommender systems [21]. PCC measures the similarity between two service users a and u based on the Web services they both invoked:

$$Sim(a,u) = \frac{\sum_{i \in I}(r_{a,i} - \overline{r}_a)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I}(r_{a,i} - \overline{r}_a)^2}\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r}_u)^2}}, \tag{22.1}$$

where $I = I_a \cap I_u$ is the set of Web services invoked by both user a and user u, $r_{a,i}$ is the QoS values of Web service i observed by service user a, $\overline{r}_a$ and $\overline{r}_u$ represent the average QoS values observed by service user a and u respectively. The PCC similarity of two service users, $Sim(a,u)$ ranges from -1 to 1. Positive PCC value indicates that the two users have similar preferences, while negative PCC value means that the two user preferences are opposite. $Sim(a,u) = null$ when two users have no Web service intersection.

PCC is used to measure the similarity between Web services in WSRec as well. The similarity computation of two Web services i and j can be calculated by:

$$Sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r}_i)(r_{u,j} - \overline{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r}_i)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{r}_j)^2}}, \tag{22.2}$$

where $Sim(i,j)$ is the similarity between Web services i and j, $U = U_i \cap U_j$ is the set of users who have invoked both Web services i and j, $\overline{r}_i$ represents the average QoS values of Web service i submitted by all users. The range of $Sim(i,j)$ is [−1,1]. $Sim(i,j) = null$ when there is no user who have used both services.

22.2.1.2 Significance Weight

PCC only considers the QoS difference between services invoked by both users. It can overestimate the similarity of two users that are not similar, but happen to have a few services with very similar QoS records [21]. To devalue the overestimated similarity, we add a correlation significance weight to PCC. An adjusted PCC for

user similarity is defined as:

$$Sim'(a,u) = \frac{2 \times |I_a \cap I_u|}{|I_a| + |I_u|} Sim(a,u), \tag{22.3}$$

where $Sim'(a,u)$ is the adjusted similarity value, $|I_a \cap I_u|$ is the number of services invoked by both users (co-invoked services), $|I_a|$ and $|I_u|$ are the number of Web services invoked by user a and user u, respectively. When the number of co-invoked Web service $|I_a \cap I_u|$ is small, the significance weight $\frac{2\times|I_a\cap I_u|}{|I_a|+|I_u|}$ will decrease the similarity estimation between users a and u. Since the value of $\frac{2\times|I_a\cap I_u|}{|I_a|+|I_u|}$ is in the interval of [0, 1], $Sim(a,u)$ is in the interval of $[-1, 1]$, the value of $Sim'(a,u)$ is in the interval of $[-1, 1]$.

Similar to Eq. (22.3), an adjusted PCC for the Web service similarity is defined as:

$$Sim'(i,j) = \frac{2 \times |U_i \cap U_j|}{|U_i| + |U_j|} Sim(i,j), \tag{22.4}$$

where $|U_i \cap U_j|$ is the number of service users who invoked both Web services i and j. The range of $Sim'(i,j)$ is $[-1, 1]$.

22.2.2 QoS Value Prediction

In reality, the user-item matrix is usually very sparse, since service users usually have QoS values on a small number of services. Predicting missing values for the user-item matrix can improve the prediction accuracy of active users. In this section, we present a missing value prediction approach to tackle this problem by making the matrix denser.

22.2.2.1 Neighbor Selection

To predict missing values, we first need to find the underlying relationship between the missing values and the existing ones, and then use this information to predict the missing ones. In the user-item matrix of WSRec system, each missing entry $r_{u,i}$ is associated with two sets of neighbors: a set of similar users $S(u)$ and a set of similar items (services) $S(i)$, which can be found by the following equations:

$$S(u) = \{u_a | u_a \in T(u), Sim'(u_a,u) > 0, u_a \neq u\}, \tag{22.5}$$

$$S(i) = \{i_k | i_k \in T(i), Sim'(i_k,i) > 0, i_k \neq i\}, \tag{22.6}$$

where $T(u)$ is a set of similar users to the user u, and $T(i)$ is a set of similar items to the item i. Both $T(u)$ and $T(i)$ are selected using enhanced PCC (Eq. (22.3) and

Eq. (22.4)). Neighbors without correlations or with negative ones are discarded from the neighbor sets.

22.2.2.2 Missing Value Prediction

For each missing entry, we use both its user neighbors and item neighbors to predict the missing value. User-based CF methods (UPCC) employ similar users to predict the missing QoS values:

$$\widehat{r_{u,i}} = \overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u)(r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)}, \tag{22.7}$$

where $\widehat{r_{u,i}}$ is the predicted QoS vector of service i for user u, $\overline{u}$ and $\overline{u}_a$ are vectors of average QoS values of all Web services observed by active user u and neighbor user u_a respectively.

Item-based CF methods (IPCC) [8, 17, 27] apply similar Web services to predict the missing value:

$$\widehat{r_{u,i}} = \overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i)(r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)}, \tag{22.8}$$

where $\overline{i}$ is a vector of average QoS values of Web service i observed by all service users.

When a missing entry only has user neighbors or item neighbors, we will employ either Eqs. (22.7) or (22.8) to predict the value. When it has both type of neighbors, we combine the two methods to make prediction. We will not predict the value if it has no neighbors.

User-based method and item-based method may have different prediction accuracy, we use *confidence weights*, con_u and con_i, to reflect our confidence in the two prediction methods. For example, assuming a missing entry has three similar users with PCC similarities {1,1,1} and three similar items with {0.1, 0.1, 0.1}. Intuitively, we have more confidence in the prediction by user-based method rather than item-based one. We define con_u as:

$$con_u = \sum_{u_a \in S(u)} \frac{Sim'(u_a, u)}{\sum_{u_a \in S(u)} Sim'(u_a, u)} \times Sim'(u_a, u), \tag{22.9}$$

and con_i as:

$$con_i = \sum_{i_k \in S(i)} \frac{Sim'(i_k, i)}{\sum_{i_k \in S(i)} Sim'(i_k, i)} \times Sim'(i_k, i), \tag{22.10}$$

where con_u and con_i are the prediction confidences of the user-based method and item-based method respectively. The higher the value, the more confidence we have in the predicted value $\widehat{r_{u,i}}$.

Since different datasets may inherit their own data distribution and correlation natures, a parameter λ ($0 \leq \lambda \leq 1$) is employed to tune the the final result combining both user-based method and item-based method. When $S(u) \neq \emptyset \wedge S(i) \neq \emptyset$, our method predicts the missing QoS value $r_{u,i}$ by employing the following equation:

$$\begin{aligned} \widehat{r_{u,i}} = w_u \times \left(\overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u)(r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)} \right) \\ + w_i \times \left(\overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i)(r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)} \right), \end{aligned} \tag{22.11}$$

where w_u and w_i are the weights of the user-based method and the item-based method respectively ($w_u + w_i = 1$). w_u is defined as:

$$w_u = \frac{con_u \times \lambda}{con_u \times \lambda + con_i \times (1 - \lambda)}, \tag{22.12}$$

and w_i is defined as:

$$w_i = \frac{con_i \times (1 - \lambda)}{con_u \times \lambda + con_i \times (1 - \lambda)}. \tag{22.13}$$

The prediction confidence of the missing value $\widehat{r_{u,i}}$ by our approach using Eq. (22.11) can be calculated by equation:

$$con = w_u \times con_u + w_i \times con_i. \tag{22.14}$$

22.2.3 Recommendation for Active Users

We use the matrix with predicted missing values to generate recommendations for active users. We first predict Web service QoS values for the active user, which is similar to the missing value prediction in Sect. 22.2.2.2. The only difference is that when $S(u) = \emptyset \wedge S(i) = \emptyset$, we predict the QoS values with user-mean (UMEAN)

and item-mean (IMEAN). UMEAN is a vector of average QoS values of all the Web services observed by the active user a and IMEAN is a vector of average QoS values of Web service i observed by all service users. The prediction formula is defined as:

$$\widehat{r_{a,i}} = w_u \times \overline{r}_a + w_i \times \overline{r}_i, \tag{22.15}$$

where $\overline{r}_a$ is the average QoS submitted by user a, and $\overline{r}_i$ is the average QoS of service i. In this case, the confidence of the predicted value is $con = 0$.

The predicted QoS values can be used in the following ways: (1) For a set of functionally equivalent Web services, the one with optimal predicted QoS values is recommended to the active user. (2) For Web services with different functionalities, the top k best performing ones will be recommended to service users to help them discover potential Web services.

22.2.4 Time Complexity Analysis

Worst-case analysis of the QoS value prediction algorithm is presented in this section. The input is a full user-item matrix with m users and n Web services.

22.2.4.1 Time Complexity of Similarity Computation

In Sect. 22.2.1, the time complexity of user similarity $Sim(a, u)$ is $O(n)$, since there are at most n intersected services between user a and u. The time complexity of service similarity $Sim(i, j)$ is $O(m)$ with at most m users who used both Web service i and j.

22.2.4.2 Time Complexity of UPCC

To predict missing values with UPCC (Eq. 22.7), we need to first compute similarities of the active user with all users in the matrix (totally m similarity computations). As discussed in Sect. 22.2.4.1, the time complexity of each similarity computation is $O(n)$. Therefore, the complexity of similarity computation is $O(mn)$.

The time complexity of each missing value prediction is $O(m)$, since at most m similar users are employed in the prediction. The complexity of the value prediction for an active user is $O(mn)$ with at most n missing values. Thus the time complexity of UPCC (including similarity computation and value prediction) is $O(mn)$.

22.2.4.3 Time Complexity of IPCC

To predict missing values with IPCC (Eq. (22.8)), we need to compute similarities of the current Web service with all Web services in the matrix (totally n similarity computations). As discussed in Sect. 22.2.4.1, the time complexity of each similarity computation is $O(m)$. Therefore, the complexity of similarity computation is $O(mn)$.

The missing value prediction computational complexity is $O(n)$, since at most n similar Web services are employed for prediction. The complexity of the value prediction for a Web service is $O(mn)$ with at most m users. Therefore, the time complexity of IPCC is $O(mn)$.

22.2.4.4 Time Complexity of Training Matrix Prediction

Training matrix prediction is an offline process, which helps reduce the sparseness of the training matrix and improve the prediction accuracy (Sect. 22.2.2.2). This process is a linear combination of UPCC and IPCC. For UPCC approach, the computational complexity is $O(m^2n)$, since there are at most m rows (users) need prediction. For IPCC approach, the complexity is $O(mn^2)$, because there are at most n columns (Web services) to be predicted. Therefore, the time complexity of matrix prediction is $O(m^2n + mn^2)$.

22.2.4.5 Time Complexity of Active User Prediction

As discussed in Sect. 22.2.4.2, the computational complexity of UPCC for predicting values of an active user is $O(mn)$. When employing IPCC, the similarities of different columns (Web services) can be computed offline, and there are at most n missing values in the active user. For the prediction of each missing value, the computational complexity is $O(n)$, since at most n similar Web services will be employed for the prediction. Therefore, the computational complexity of IPCC for an active user is $O(n^2)$. Since our online QoS value prediction approach is a linear combination of UPCC and IPCC, the complexity of our approach for an active user is $O(mn + n^2)$.

22.3 A Region-Based Web Service Recommendation Algorithm

We present a region-based Web service recommendation algorithm in this section. The main hypothesis is that some QoS properties vary according to users' physical locations. Through the analysis of a real world Web service data set (see Sect. 22.4), which contains 1.5 millions service invocation records evaluated by users from more than twenty countries, we discover that some QoS properties like response time highly relate to 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.

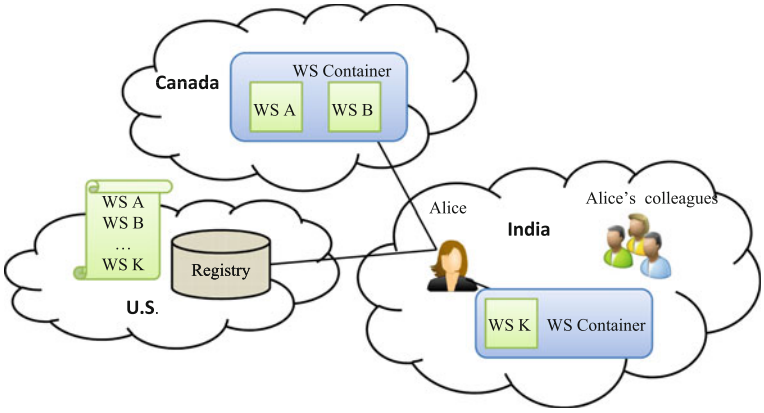

Fig. 22.2 A motivating scenario

22.3.1 A Motivating Scenario

Alice is a software engineer working in India. She needs an email validation service to filter emails. By querying a registry in U.S, she gets a list of service candidates and sorts the services in ascending order of the response time. Alice then tries the first two services provided by a Canadian company. However, she finds that the response time is much higher than her expectation. She then realizes that the response time is based on the evaluation conducted by servers in U.S., and it can vary greatly due to different user contexts, such as user location, user network conditions, etc. Alice then turns to her colleagues for suggestion. They suggest a local service though ranked lower in the previous search result. After trying it, Alice finds that that service has good performance and meets her requirements.

Intuitively, users closely located with each other are more likely to have similar service experience than those who are far away from each other. Our recommendation approach is designed as a two-phase process. In the first phase, we divide users into different regions based on their physical locations and historical Web service QoS experience. In the second phase, we find similar users for the active user, make QoS predictions for the unused services, and finally recommend the one with best predicted value to the user.

22.3.2 Phase One: Region Creation

A *region* is defined 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. To create regions, we first form small regions by putting users with similar locations

together and extract region features. Then we aggregate highly correlated regions to form a certain number of large regions. Steps to create regions are presented in Sect. 22.3.2.1–22.3.2.2 respectively.

22.3.2.1 Region Features

Region center is used to reflect the average performance of Web services observed by users of one region. Region center is defined as the median vector of all QoS vectors associated with the region users. The element i of the center is the median QoS 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 QoS performance, QoS fluctuation is another feature deserves our attention. From large real data analysis, we discover that user-dependent QoS properties (e.g., response time) usually varies from region to region. Some services have unexpected long response time, and some are even inaccessible to a few regions. Inspired by the three-sigma rule which is often used to test outliers, we use similar method to distinguish services with unstable performance and regard them as region-sensitive services.

We pick one QoS property r (response time) to simplify the description of this approach. The set of non-zero QoS values of service s, $r_{.s} = \{r_{1,s}, r_{2,s}, \ldots, r_{k,s}\}$, $1 \leq k \leq m$, collected from users of all regions is a sample from the population of service s QoS property R. To estimate the mean μ and the standard deviation σ of the population, we use two robust measures: median and Median Absolute Deviation (MAD). MAD is defined as the median of the absolute deviations from the sample's median.

$$MAD = median_i(|r_{i,s} - median_j(r_{j,s})|), i = 1, \ldots, k, j = 1, \ldots, k \quad (22.16)$$

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

$$\widehat{\mu} = median_i(r_{i,s}), i = 1, \ldots, k \quad (22.17)$$

$$\widehat{\sigma} = MAD_i(r_{i,s}), i = 1, \ldots, k \quad (22.18)$$

Definition 22.1 Let $r_{.s} = \{r_{1,s}, r_{2,s}, \ldots, r_{k,s}\}$, $1 \leq k \leq m$ be the set of Web service s QoS values provided by all users. Service s is a sensitive service to region M iff $\exists r_{j,s} \in r_{.s}((r_{j,s} > \widehat{\mu} + 3\widehat{\sigma}) \wedge region(j) = M)$, where $\widehat{\mu} = median(r_{.s})$, $\widehat{\sigma} = MAD(r_{.s})$ and $region(u)$ function defines the region of user u.

Definition 22.2 The sensitivity of region r_m is the fraction between the number of sensitive services in region r_m over the total number of services.

Definition 22.3 Region r_m is a sensitive region iff its region sensitivity exceeds the sensitivity threshold λ.

22.3.2.2 Region Aggregation

Each region formed by users' physical locations at the outset always has a very sparse QoS dataset, since the amounts of users and QoS records are relatively small. In this case, it is difficult to find similar users and predict missing QoS records. To solve this problem, we aggregate these small regions based on the similarity of their features. The similarity of two regions r_m and r_n is measured by the similarity of their region centers c_{r_m} and c_{r_n} using Eq. (22.3).

We use a bottom-up hierarchical clustering algorithm to aggregate regions [20]. 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 is met. The result is stored as a list of aggregates in A.

Step one: Initialization

1. Compute the similarity between each two regions with Eq. (22.3), 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 two: Aggregation

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.

22.3.3 Phase Two: QoS Prediction and Recommendation

The region aggregation step clusters thousands of users into a certain number of regions based on their physical locations and historical QoS similarities. With the compressed QoS data, searching neighbors and making predictions for an active user can be computed faster than the conventional methods. Instead of computing the similarity between the active user and each existing user, we only compute the similarity

Algorithm 1: Region Aggregation Algorithm

```
Input: a list of regions r_1, ... r_l
Output: result list A
foreach n ← 1 to l − 1 do
    foreach i ← n + 1 to N do
        C[n][i].sim ← SIM(r_n, r_i);
        C[n][i].index ← i;
    I[n].sensitivity ← ISSENSITIVE(r_n);
    if I[n].sensitivity = 0 then
        I[n].aggregate ← 1;
    I[n].aggregate ← 0;
    P[n] ← priority queue for C[n] sorted on sim;
calculate the sensitivity and aggregate of I[l] ;
A ← [];
while true do
    k_1 ← argmax_{k:I[k].aggregate=1} P[k].MAX().sim;
    if k_1 = null or sim < μ then
        return A;
    k_2 ← P[k_1].MAX().index;
    A.APPEND(< k_1, k_2 >) and compute k_1 center;
    I[k_2].aggregate ← 0;
    P[k_1] ← [];
    I[k_1].sensitivity ← ISSENSITIVE(k_1);
    if I[k_1].sensitivity = 1 then
        I[k_1].aggregate ← 0;
        foreach i with I[i].aggregate = 1 do
            P[i].DELETE(C[i][k_1]);
            P[i].DELETE(C[i][k_2]);
    else
        foreach i with I[i].aggregate = 1 ∧ i ≠ k_1 do
            P[i].DELETE(C[i][k_1]);
            P[i].DELETE(C[i][k_2]);
            C[i][k_1].sim ← SIM(i, k_1);
            P[i].INSERT(C[i][k_1]);
            C[k_1][i].sim ← SIM(i, k_1);
            P[k_1].INSERT(C[k_1][i]);
```

between the active user and each region center. Moreover, it is more reasonable to predict the QoS value based on one's region, 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 unused Web service s's QoS value for an active user a, we take the following steps:

- Find the region of the active 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 user a's region. If region-sensitive, then the prediction is generated from the region center, because QoS of service s observed by users from this region is significantly different from others.

$$\widehat{r_{a,s}} = r_{c,s} \tag{22.19}$$

- Otherwise, use Eq. (22.3) 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 Sect. 22.4.
- If the active user's region center has QoS value of s, the prediction is computed using the following equation:

$$\widehat{r_{a,s}} = r_{c,s} + \frac{\sum_{j=1}^{k} Sim'(a, c_j)(r_{c_j,s} - \overline{r_{c_j.}})}{\sum_{j=1}^{k} Sim'(a, c_j)} \tag{22.20}$$

  where $r_{c_j,s}$ is the QoS of service s provided by center c_j, and $\overline{r_{c_j.}}$ is the average QoS of center c_j. The prediction is composed of two parts. One is the QoS value of the active user's region center $r_{c,s}$,which denotes the average QoS of service s observed by this region users. The other part is the normalized weighted sum of the deviations of the k most similar neighbors.
- Otherwise, we use the service QoS observed by the k neighbors to compute the prediction. The more similar the active user a and the neighbor c_j are, the more weights the QoS of c_j will carry in the prediction.

$$\widehat{r_{a,s}} = \frac{\sum_{j=1}^{k} Sim'(a, c_j) r_{c_j,s}}{\sum_{j=1}^{k} Sim'(a, c_j)} \tag{22.21}$$

22.3.4 Time Complexity Analysis

We discuss the worst-case time complexity of the region-based Web service recommendation algorithm. We analyze the two phases, region creation and QoS value prediction respectively in Sect. 22.3.4.1 and 22.3.4.2. We assume the input is a full matrix with n users and m Web services.

22.3.4.1 Time Complexity of Region Creation

Time complexity of calculating the median and MAD of each service is $O(n \log n)$. For m services, the time complexity is $O(mn \log n)$. 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 22.1. Therefore, the total time complexity of region-sensitive service identification is $O(mn \log n + mn) = O(mn \log n)$.

In terms of the region aggregation part, we assume there are l_0 regions in the beginning. Since there are at most m services used by both regions, the time complexity of the region similarity is $O(m)$ using Eq. (22.3), and the complexity for computing similarity matrix C is $O(l_0^2 m)$.

The aggregation of two regions will execute at most $l_0 - 1$ times, 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. Since the number of regions that can be aggregated decreases with iteration, the real search time will be less than $l_0 - 1$ the following iterations. 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 $O(mn \log n)$. The insertion and deletion of a similar region is $O(log\ l_0)$, since we employ the priority queue to sort similar regions. Thus, the time complexity is $O(l_0^2(\log l_0 + mn \log n)) = O(l_0^2 mn \log n)$.

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

22.3.4.2 Time Complexity of QoS Prediction

Let l_1 be the number of regions after the region creation. To predict 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. Therefore the time complexity of similarity computation is $O(l_1 m)$.

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 Eqs. (22.20) and (22.21) state. There are at most m services without QoS values, so the time complexity of the prediction for an active user is $O(l_1 m)$. Thus the time complexity for online phase including similarity computation and missing value prediction is $O(l_1 m) \approx O(m)$ (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.

22.4 Experiments

22.4.1 Experiment Setup

In this experiment, 21,197 publicly available Web services are crawled from three sources (1) well-known companies (e.g., *Google, Amazon, etc.*); (2) portals listing publicly available Web services (e.g., *xmethods.net, webservicex.net, etc.*); and (3)

Web service searching engines (e.g., *seekda.com, esynaps.com, etc.*). 18,102 Web services stubs with 343,917 Java classes are generated using *WSDL2Java* tool of *Axis2* package. Failures to generate client stub are mainly caused by *network connection problems* (e.g., connection timeout, HTTP 400, 401, 403, 500, 502 and 503), *FileNotFoundException* and *InvalidWSDLFiles*.

To monitor Web service performance, we randomly select 100 Web services located in 22 countries for our experiments. 150 computers in 24 countries from Planet-Lab [7] are employed to monitor and collect QoS information of the selected Web services. The result set contains about 1.5 millions Web service invocation records.

By processing the experimental results, we obtain a 150×100 user-item matrix, where each entry in the matrix is a vector including two QoS values, i.e., *response time* and *failure rate*. *Response time* represents the time duration between the client sending a request and receiving a response, while *failure rate* represents the ratio between the number of invocation failures and the total number of invocations. In our experiments, each service user invokes each Web service for 100 times. Figure 22.3a, b show the value distributions of *response time* and *failure rate* of the 15,000 entries in the matrix, respectively. Figure 22.3a shows that the means of response times of most entries are smaller than 5000 milliseconds and different Web service invocations contain large variances in real environment. Figure 22.3b shows that failure probabilities of most entries (85.68 %) are less than 1 %, while failure probabilities of a small part of entries (8.34 %) are larger than 16 %. We divide the 150 service users into two parts, one part as training users and the other part as active users. For the training matrix, we randomly remove entries to generate a series of sparse matrices (e.g., with density 10, 20 %, ect.). For an active user, we also randomly remove several entries and name the number of remaining entries as *given number*, which denotes the number of entries (QoS values) provided by the active user. The original values of the removed entries are used as the expected values to study the prediction accuracy.

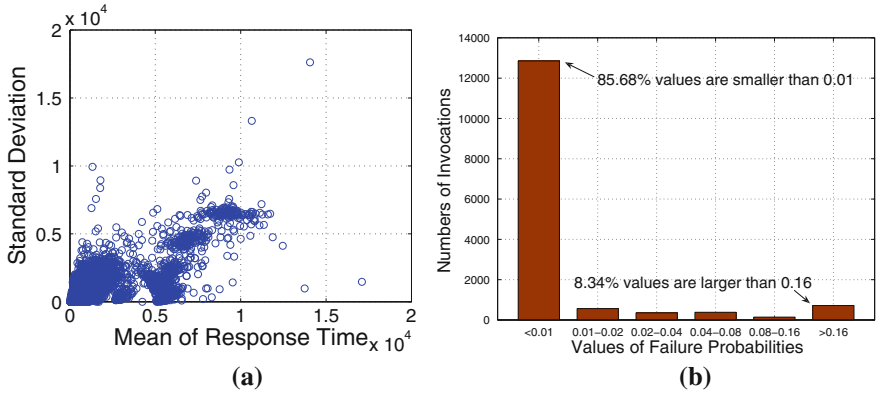

Fig. 22.3 Value distributions of the user-item matrix

We use Mean Absolute Error (MAE) to measure the prediction quality of the recommendation algorithms. MAE is the average absolute deviation of predictions to the ground truth data. Smaller MAE indicates better prediction accuracy.

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \widehat{r}_{i,j}|}{N}, \tag{22.22}$$

where $r_{i,j}$ denotes the expected QoS value of Web service j observed by user i, $\widehat{r}_{i,j}$ is the predicted QoS value, and N is the number of predicted values. MAE reflects how close predictions are to the eventual outcomes on average, which gives an overview of the prediction quality.

22.4.2 WSRec Performance Evaluation

To study the prediction performance, we compare our approach (*WSRec*) with user-based prediction algorithm using PCC (*UPCC*) [3], and item-based algorithm using PCC (*IPCC*) [27]. UPCC employs similar users for QoS performance prediction (Eqs. (22.1) and (22.7)), while IPCC employs similar Web services for prediction (Eqs. (22.2) and (22.8)).

Table 22.2 shows the MAE result of different prediction methods on *response time* and *failure rate* employing matrices with 10, 20, and 30 % density. We vary the number of QoS values (*given number*) provided by the active user from 10, 20 to 30 (named as G10, G20, and G30 in Table 22.2). We also vary the number of training users as 100 and 140. We set $\lambda = 0.1$, since the item-based approach achieves better prediction accuracy than the user-based approach in our Web service QoS dataset. The detailed investigation of λ value setting will be shown in Sect. 22.4.2.2.

Table 22.2 MAE performance comparison (smaller MAE value means better prediction accuracy)

		Training users = 100						Training users = 140					
		Response time			**Failure rate**			**Response time**			**Failure rate**		
Den %	**Method**	**G10**	**G20**	**G30**	**G10 %**	**G20 %**	**G30 %**	**G10**	**G20**	**G30**	**G10 %**	**G20 %**	**G30 %**
	UPCC	1148	877	810	4.85	4.20	3.86	968	782	684	4.11	3.47	3.28
	IPCC	768	736	736	2.24	2.16	2.21	585	596	605	1.39	1.33	1.42
10	**WSRec**	**758**	**700**	**672**	**2.21**	**2.08**	**2.08**	**560**	**533**	**500**	**1.36**	**1.26**	**1.24**
	UPCC	904	722	626	4.40	3.43	2.85	794	626	540	3.93	2.96	2.43
	IPCC	606	610	639	2.01	1.98	1.98	479	509	538	1.17	1.22	1.28
20	**WSRec**	**586**	**551**	**546**	**1.93**	**1.80**	**1.70**	**445**	**428**	**416**	**1.10**	**1.08**	**1.07**
	UPCC	915	671	572	4.25	3.25	2.58	803	576	491	3.76	2.86	2.06
	IPCC	563	566	602	1.84	1.83	1.86	439	467	507	1.10	1.12	1.17
30	**WSRec**	**538**	**504**	**499**	**1.78**	**1.69**	**1.63**	**405**	**385**	**378**	**1.05**	**1.00**	**0.98**

Each experiment is run for 50 times and the average MAE value is reported. The experimental results of Table 22.2 shows that:

- *WSRec* method consistently outperforms other algorithms under all experimental settings.
- The performance of all approaches enhances significantly with the increase of matrix density, the number of training users as well as the number of QoS values provided by active users.
- The item-based approach (IPCC) outperforms the user-based approach (UPCC). This observation indicates that similar Web services provide more information to the prediction than similar users do.

22.4.2.1 Impact of Missing Value Prediction

The *missing value prediction* in Sect. 22.2.2.2 makes use of the similar users and similar items to predict the missing values of the training matrix to make it denser. To study the impact of the *missing value prediction*, we implement two versions of *WSRec*. One version employs missing value prediction (*WSRec**), while the other version does not (*WSRec*). We vary the *given number* of the active users from 5 to 50 with a step of 5 and vary the values of *training users* from 20 to 140 with a step value of 20. We set $density = 10\,\%$ and $TopK = 10$, which means that the top 10 neighbors will be employed for value prediction.

Figure 22.4 shows the experimental results, where Fig. 22.4a–b show the experimental results of *response time* and Fig. 22.4c–d show the experimental results of *failure rate*. Figure 22.4 shows that predicting the missing values of the training matrix will improve the overall prediction accuracy.

22.4.2.2 Impact of λ

Different datasets have different data characteristics. Parameter λ makes our prediction method feasible and adaptable to different datasets. If $\lambda = 1$, we only extract information from similar users, and if $\lambda = 0$, we only consider valuable information from similar services. In other cases, we leverage both similar users and services to predict missing values for active users.

To study the impact of λ on our collaborative filtering method, we set $Top\ K = 10$ and $training\ users = 140$. We vary the value of λ from 0 to 1 with a step of 0.1. Figure 22.5a, c show the results of $given\ number = 10$, $given\ number = 20$ and $given\ number = 30$ with 20 % density training matrix of *response time* and *failure rate*, respectively. Figure 22.5b, d show the results of $density = 10\,\%$, $density = 20\,\%$ and $density = 30\,\%$ with $given\ number = 20$ of *response time* and *failure rate*, respectively.

The experiment shows that λ impacts the recommendation results significantly, and a proper λ value will provide better prediction accuracy. For both the *response*

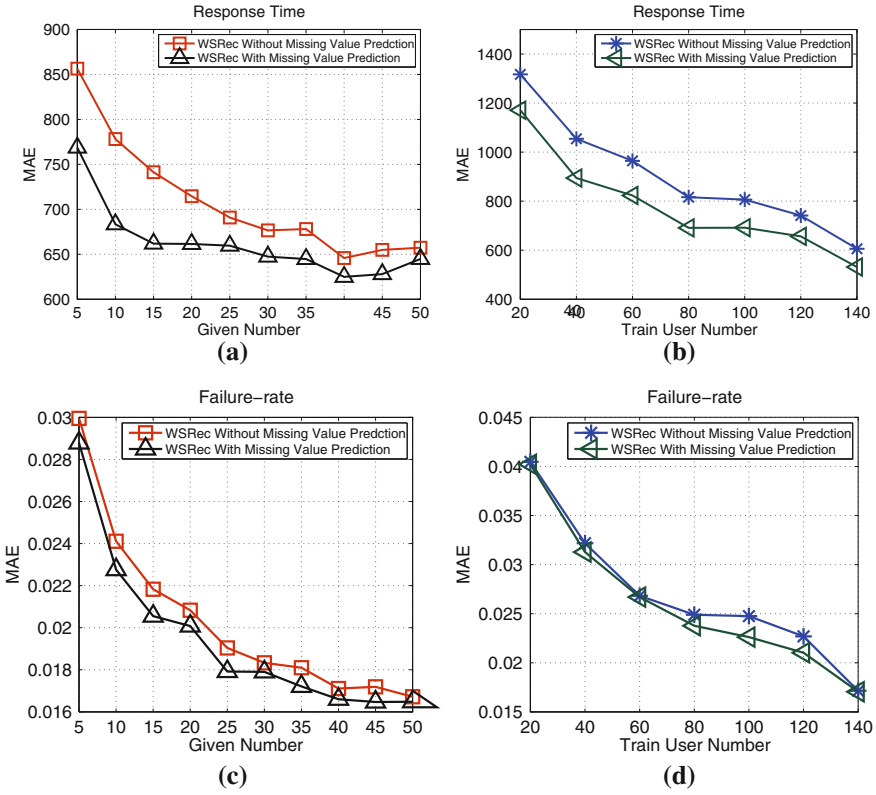

Fig. 22.4 Impact of the training matrix prediction

time and *failure rate*, similar Web services are more important than similar users in prediction QoS when limited QoS values are given by active users, while the similar users become more important when more QoS values are available from active users. As shown in Fig. 22.5b, d, with the given number of 20, all the three curves ($Density$ 10, 20, *and* 30 %) of *response time* and *failure rate* obtain the best prediction performance with the same λ value ($\lambda = 0.2$ for *response time* and $\lambda = 0$ for *failure rate*), indicating that the optimal λ value is not influenced by the training matrix density.

22.4.3 Region-Based Recommender System Performance Evaluation

As mentioned in Sect. 22.4.1, QoS records are collected by 150 nodes from the PlanetLab. For each node, there are more than 100 QoS profiles, and each profile contains

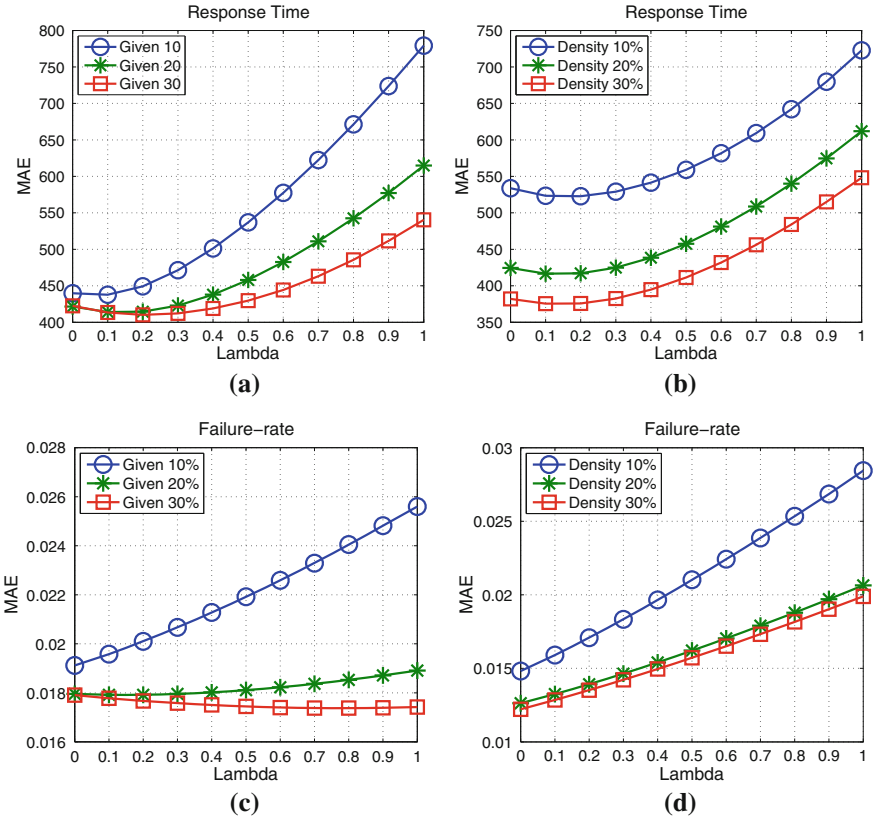

Fig. 22.5 Impact of the lambda

the response time (also called Round Trip Time, 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 randomly remove 90 and 80 % RTTs of the initial training matrix to generate two sparse matrices with density 10 and 20 % respectively. We vary the number of RTT values given by active users from 10, 20 to 30. The removed records of active users are used to study the prediction accuracy. In this experiment, we set $\mu = 0.3$, $\lambda = 0.8$, and $top - k = 10$. To get a reliable error estimate, we use 10 times 10-fold cross-validation [32] to evaluate the prediction accuracy and report the average MAE value.

Table 22.3 shows the prediction performance of different methods employing the 10 and 20 % density training matrix. It shows that our method (RBCF) 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).

Table 22.3 MAE comparison on response time (smaller value means better performance)

	Density = 10%			Density = 20%		
Method	**G10**	**G20**	**G30**	**G10**	**G20**	**G30**
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
RBCF	**638.21**	**624.51**	**623.90**	**573.85**	**560.13**	**556.75**

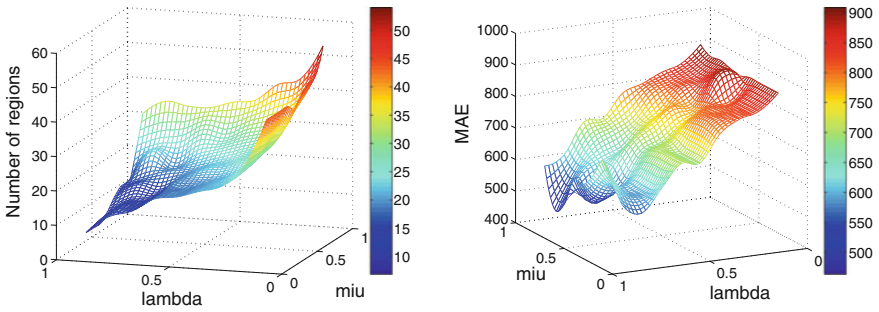

Fig. 22.6 Impact of thresholds λ and μ. **a** Impact on the number of regions. **b** Impact on the prediction performance (MAE)

22.4.3.1 Impact of λ and μ

In region creation phase, the two thresholds λ and μ play a very important role in determining the number of regions and can impact the final performance of the prediction. As mentioned in Sect. 22.3.2.2, only regions with similarity higher than μ and sensitivity less than λ can be aggregated into one region. We test 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. Figure 22.6 shows how the two thresholds affect the number of regions and the final prediction accuracy. 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. Note that the optimal value of λ is related to the sensitivity of the original regions at the outset. Figure 22.7 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. Figure 22.8 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 $\lambda = 1$, so that we do not consider the factor

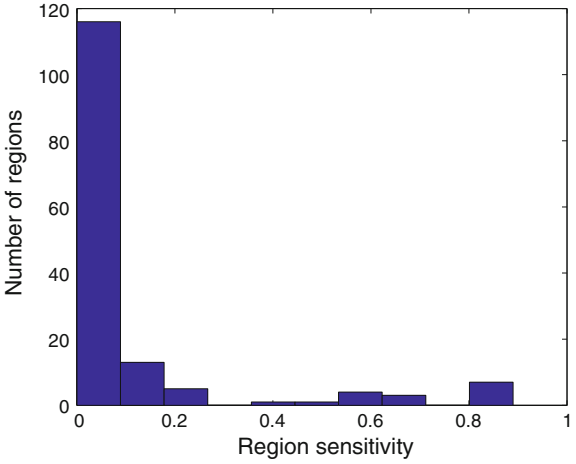

Fig. 22.7 The distribution of region sensitivity

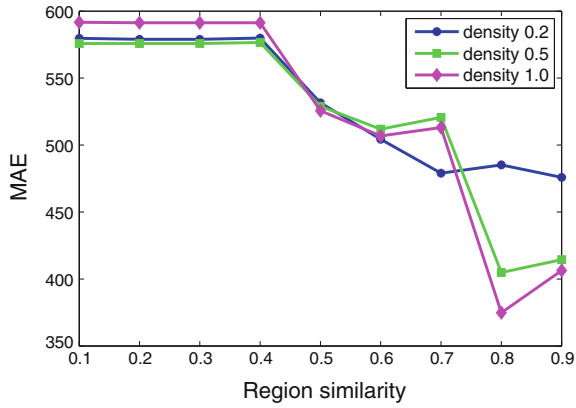

Fig. 22.8 The distribution of region sensitivity

of sensitivity in region aggregation. Similarity becomes the single factor. Obviously, for denser matrix, with higher μ we obtain a set of coherent regions, and better prediction accuracy.

22.5 Related Work

22.5.1 Collaborative Filtering

Collaborative Filtering is firstly proposed by Rich [25] and widely used in commercial recommender systems, such as Amazon.com [4, 17, 19, 24]. 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. Essentially, CF is based on processing the

user-item matrix. Breese et al. [3] divide the CF algorithms into two broad classes: memory-based algorithms and model-based algorithms. The most analyzed examples of memory-based collaborative filtering include user-based approaches [3, 10, 14], item-based approaches [8, 17, 27], and their fusion [37, 31]. User-based approaches predict the ratings of active users based on the ratings of their similar users, and item-based approaches predict the ratings of active users based on the computed information of items similar to those chosen by the active users. These algorithms are easy to implement, require little or no training cost, and can easily take new users's ratings into account. However, memory-based algorithms cannot cope well with large number of users and items, since their online performance is often slow.

Model-based CF algorithms learn the model from the dataset using statistical and machine learning techniques. Examples include clustering model [33], latent semantic models [11, 12] and latent factor model [5]. These 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 system.

22.5.2 Web Service Selection and Recommendation

Web service selection and recommendation has been extensively studied to facilitate Web service composition in recent years. El Hadad et al. [9] propose a selection method considering both the transactional properties and QoS characteristics of a Web service. Hwang et al. [13] find that both composite and individual web services constrain the sequences of invoking operations. They use finite state machine to model the permitted invocation sequences of Web service operations, and propose two strategies to select Web services that are likely to successfully complete the execution of a given sequence of operations. Kang et al. [15] propose AWSR system to recommend services based on users' historical functional interests and QoS preferences. Barakat [2] models the quality dependencies among services and propose a Web service selection method for Web service composition. Alrifai and Risse [1] propose a method to meet a user's end-to-end QoS requirement. Their method consists of two steps: first, they use mixed integer programming (MIP) to find the optimal decomposition of global QoS constraints into local constraints. After that they use distributed local selection to find the best web services that satisfy the local constraints. This approach achieves suboptimal results, but it is more efficient than solutions based on global optimization.

A large amount of work has been done to apply CF to Web service recommendation. Shao et al. [28] use a user-based CF algorithm to predict QoS values. Work [16, 29] apply the idea of CF in their systems, and use MovieLens data for experimental analysis. Combination of different type of CF algorithms are also used in Web service recommendation. Zheng et al. [40] combine the user-based and item-based CF algorithms to recommend Web services; They also integrate Neighborhood approach with Matrix Factorization in work [39]. Qi [23] presents a strategy that integrates matrix factorization with decision tree learning to bootstrap service recommenda-

tion systems. Meanwhile, several work employs location information to Web service recommendation. Chen et al. [6] first use a region-based CF algorithm to make Web service recommendations. To help users know more about Web service performance, they also propose a visualization method showing recommendation results on a map . Lo et al. [18] employs the user location in matrix factorization model to predict QoS values. Tang et al. [30] consider the impact of both user location and Web service location on QoS values and propose a CF recommendation approach based on that.

22.6 Conclusion and Future Work

We have presented two Web service recommendation approaches in this chapter. The basic ideas of the two are the same: to predict Web service future QoS performance and recommend the best one for active users by using historical QoS data from similar users. The difference is how the two approaches find similar users. Neighborhood-based approach searches users and Web services in the entire data set to find similar ones. It is straightforward and easy to implement. Moreover, this approach can easily handle new data (new users, Web services and submitted QoS values) by adding new rows or columns to the data set. On the other hand, region-based approach leverages location information to find similar users and achieves better online performance. The drawback of this approach is that we need to recompute the region model when a certain amount of new data coming in. For example, when one normal region becomes sensitive or when a lot of new users go to one region and make it not coherent, we will regenerate all the regions.

In our future work, we will consider several aspects to further improve the proposed Web service recommendation approaches. In terms of the recommendation accuracy, we find that contextual information can greatly influence Web service QoS performance, such as server workload, network condition and the tasks that users carry out with Web services (e.g., computation-intensive or I/O-intensive task). Besides physical location, we will take these factors into account and refine the steps of similarity computation and region aggregation. In terms of the experiment, we use MAE to measure the overall recommendation accuracy currently. Similar to web page search results, users may only consider and try the top three or five recommended services. Thus improving the accuracy of top-k recommended services is another task to investigate. Our future work also includes the study of QoS characteristic. We plan to investigate the distribution of response time and the correlation between different QoS properties such as response time and reliability.

References

1. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient QoS-aware service composition. In: Proceedings of the 18th International Conference on World Wide Web (WWW'09), pp. 881–890 (2009)
2. Barakat, L.: Efficient correlation-aware service selection. In: Proceedings of the 19th International Conference on Web Services (ICWS'12), pp. 1–8 (2012)
3. Breese, J.S., Heckerman, D., Kadie C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Annual Conference Uncertainty in Artificial Intelligence (UAI'98), pp. 43–52 (1998)
4. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adap. Inter. **12**(4), 331–370 (2002)
5. Canny J.: Collaborative filtering with privacy via factor analysis. In: Proceedings of the 25th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'02), pp. 238–245 (2002)
6. Chen, X., Zheng, Z., Liu, X., Huang, Z., Sun, H.: Personalized QoS-aware web service recommendation and visualization. IEEE Trans. Serv Comput. **6**(1), 35–47(2013)
7. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: Planetlab: an overlay testbed for broad-coverage services. ACM SIGCOMM Comput. Commun. Rev. **33**(3), 3–12 (2003)
8. Deshpande, M., Karypis, G.: Item-based top-n recommendation. ACM Trans. Inf. Syst. **22**(1), 143–177 (2004)
9. El Hadad, J., Manouvrier, M., Rukoz, M.: TQoS: transactional and QoS-aware selection algorithm for automatic Web service composition. IEEE Trans. Serv. Comput. **3**(1), 73–85 (2010)
10. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), pp. 230–237 (1999)
11. Hofmann, T.: Collaborative filtering via gaussian probabilistic latent semantic analysis. In: Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03), pp. 259–266 (2003)
12. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst. **22**(1), 89–115 (2004)
13. Hwang, S., Lim, E., Lee, C., Chen, C.: Dynamic web service selection for reliable web service composition. IEEE Trans. Serv. Comput. **1**(2), 104–116 (2008)
14. Jin, R., Chai, J.Y., Si, L.: An automatic weighting scheme for collaborative filtering. In: Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04), pp. 337–344 (2004)
15. Kang, G., Liu, J., Tang, M., Liu, X., Cao, B., Xu, Y.: AWSR: active web service recommendation based on usage history. In: Proceedings of the 19th International Conference on Web Services (ICWS'12), pp. 186–193 (2012)
16. Karta, K.: An investigation on personalized collaborative filtering for web service selection. Honours Programme thesis, University of Western Australia, Brisbane (2005)
17. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003)
18. Lo, W., Yin, J., Deng, S., Li, Y., Wu, Z.: Collaborative web service QoS prediction with location-based regularization. In: Proceedings of the 19th International Conference on Web Services (ICWS'12), pp. 464–471 (2012)
19. Ma, H., King, I., Lyu, M.R.: Effective missing data prediction for collaborative filtering. In: Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07), pp. 39–46 (2007)
20. Manning, C.D., Raghavan, P., Schtze H.: An Introduction to Information Retrieval. Cambridge University Press, Cambridge (2009)

21. McLaughlin M.R., Herlocker J. L.: A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In: Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04), pp. 329–336 (2004)
22. Ouzzani, M., Bouguettaya, A.: Efficient access to web services. IEEE Internet Comput. **8**(2), 34–44 (2004)
23. Qi, Y.: Decision tree learning from incomplete QoS to bootstrap service recommendation. In: Proceedings of the 19th International Conference on Web Services (ICWS'12), pp. 194–201 (2012)
24. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of ACM Conference on Computer Supported Cooperative, Work, pp. 175–186 (1994)
25. Rich, E.: User modeling via stereotypes. Cognitive Sci. **3**(4), 329–354 (1979)
26. Rosario, S., Benveniste, A., Haar, S., Jard, C.: Probabilistic Qos and soft contracts for transaction-based web services orchestrations. IEEE Trans. Serv. Comput. **1**(4), 187–200 (2008)
27. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web (WWW'01), pp. 285–295 (2001)
28. Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., Mei, H.: Personalized qos prediction for web services via collaborative filtering. In: Proceedings of the 5th International Conference on Web Services (ICWS'07), pp. 439–446 (2007)
29. Sreenath, R.M., Singh, M.P.: Agent-based service selection. J. Web Seman **1**(3), 261–279 (2003)
30. Tang, M., Jin, Y., Liu, J., Liu, X.: Location-aware collaborative filtering for QoS-based service recommendation. In: Proceedings of the 19th International Conference on Web Services (ICWS'12), pp. 202–209 (2012)
31. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06), pp. 501–508 (2006)
32. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, vol. 2. Elsevier, Amsterdam (2005)
33. Xue, G., Lin, C., Yang, Q., Xi, W., Zeng, H., Yu, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05), pp. 114–121 (2005)
34. Yu, T., Zhang, Y., Lin, K.-J.: Efficient algorithms for web services selection with end-to-end qos constraints. ACM Trans. Web **1**(1), 1–26 (2007)
35. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw Eng **30**(5), 311–327 (2004)
36. Zhang, L.-J., Zhang, J., Cai, H.: Services Computing. Springer and Tsinghua University Press, New York and Beijing (2007)
37. Zheng, Z., Ma, H., Lyu, M.R., King I.: Wsrec: a collaborative filtering based web service recommender system. In: Proceedings of the 7th International Conference Web Services (ICWS'09), pp. 437–444 (2009)
38. Zheng, Z., Zhang, Y., Lyu, M.: CloudRank: A QoS-Driven component ranking framework for cloud computing. In: Proceedings of the International Symposium Reliable Distributed Systems (SRDS'10), pp. 184–193 (2010)
39. Zheng, Z., Ma, H., Lyu, M., King, I.: Collaborative web service QoS prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. (2011)
40. Zheng, Z., Ma, H., Lyu, M., King, I.: Qos-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**(2), 140–152 (2011)