Web Service Recommendation via Exploiting Location and QoS Information

Xi Chen, Zibin Zheng, *Member, IEEE*, Qi Yu, *Member, IEEE*, and Michael R. Lyu, *Fellow, IEEE*

Abstract—Web services are integrated software components for the support of interoperable machine-to-machine interaction over a network. Web services have been widely employed for building service-oriented applications in both industry and academia in recent years. The number of publicly available Web services is steadily increasing on the Internet. However, this proliferation makes it hard for a user to select a proper Web service among a large amount of service candidates. An inappropriate service selection may cause many problems (e.g., ill-suited performance) to the resulting applications. In this paper, we propose a novel collaborative filtering-based Web service recommender system to help users select services with optimal Quality-of-Service (QoS) performance. Our recommender system employs the location information and QoS values to cluster users and services, and makes personalized service recommendation for users based on the clustering results. Compared with existing service recommendation methods, our approach achieves considerable improvement on the recommendation accuracy. Comprehensive experiments are conducted involving more than 1.5 million QoS records of real-world Web services to demonstrate the effectiveness of our approach.

Index Terms—Web service, quality of service (QoS), recommendation, collaborative filtering

1 Introduction

Web services are software components designed to support interoperable machine-to-machine interaction over a network, usually the Internet. Web service employs WSDL (Web Service Description Language) for interface description and SOAP (Simple Object Access Protocol) for exchanging structured information. Benefiting from the cross-language and cross-platform characteristics, Web services have been widely employed by both enterprises and individual developers for building service-oriented applications. The adoption of Web services as a delivery model in business has fostered a paradigm shift from the development of monolithic applications to the dynamic set-up of business processes.

When developing service-oriented applications, developers first design the business process according to requirements, and then try to find and reuse existing services to build the process. Currently, many developers search services through public sites like Google Developers (developers.google.com), Yahoo! Pipes (pipes.yahoo.com), programmableWeb (programmableweb.com), etc. However, none of them provide location-based QoS information for users. Such information is quite important for software deployment especially when trade compliance is concerned. Some Web services are only available in EU, thus software employing these services cannot be shipped to other countries. Without knowledge of these things, deployment of service-oriented software can be at great risk.

Since selecting a high quality Web service among a large number of candidates is a non-trivial task, some developers choose to implement their own services instead of using publicly available ones, which incurs additional overhead in both time and resource. Using an inappropriate service, on the other hand, may add potential risk to the business process. Therefore, effective approaches to service selection and recommendation are in an urgent need, which can help service users reduce risk and deliver high-quality business processes.

Quality-of-Service (QoS) is widely employed to represent the non-functional characteristics of Web services and has been considered as the key factor in service selection [33]. QoS is defined as a set of properties including response time, throughput, availability, reputation, etc. Among these QoS properties, values of some properties (e.g., response time, user-observed availability, etc.) need to be measured at the client-side [26]. It is impractical to acquire such QoS information from service providers, since these QoS values are 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 values of the same Web service. In other words, QoS values evaluated by one user cannot be employed directly by another for service selection. It is also impractical for users to acquire QoS information by evaluating all service candidates by themselves, since conducting real world Web service invocations is time-consuming and resource-consuming. Moreover, some QoS

- *X. Chen is with Shenzhen Research Institute, The Chinese University of Hong Kong, and also with Schlumberger Technologies (Beijing) Ltd., Beijing, China.*
- *Z. Zheng and M.R. Lyu are with Shenzhen Research Institute, The Chinese University of Hong Kong, and are also with the Ministry of Education Key Laboratory of High Confidence Software Technologies (CUHK Sub-Lab), CSE department, The Chinese University of Hong Kong. E-mail: zbzheng@cse.cuhk.edu.hk.*
- *Q. Yu is with the College of Computing and Information Sciences, Rochester Institute of Technology, USA.*

Manuscript received 23 Apr. 2013; revised 14 Oct. 2013; accepted 3 Dec. 2013. Date of publication 12 Dec. 2013; date of current version 13 June 2014. Recommended for acceptance by A. Kshemkalyani.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.308

properties (e.g., reliability) are difficult to be evaluated as long-duration observation is required.

To attack this challenge, this paper investigates personalized QoS value prediction for service users by employing the available past user experiences of Web services from different users. Our approach requires no additional Web service invocations. Based on the predicted QoS values of Web services, personalized QoS-aware Web service recommendations can be produced to help users select the optimal service among the functionally equivalent ones. From a large number of real-world service QoS data collected from different locations, we find that the user-observed Web service QoS performance has strong correlation to the locations of users. Google Transparency Report[1] has similar observation on Google services.

To enhance the prediction accuracy, we propose a location-aware Web service recommender system (named LoRec), which employs both Web service QoS values and user locations for making personalized QoS prediction. Users of LoRec share their past usage experience of Web services, and in return, the system provides personalized service recommendations to them. LoRec first collects user-observed QoS records of different Web services and then groups users who have similar QoS observations together to generate recommendations. Location information is also considered when clustering users and services. The main contributions of this work are two-fold:

- First, we propose a novel location-aware Web service recommendation approach, which significantly improves the recommendation accuracy and time complexity compared with existing service recommendation algorithms.
- Second, we conduct comprehensive experiments to evaluate our approach by employing a real-world Web service QoS data set. More than 1.5 millions real-world Web service QoS records from more than 20 countries are engaged in our experiments. Comprehensive analysis on the impact of the algorithm parameters is also provided.

The rest of this paper is organized as follows: Section 2 reviews related work of collaborative filtering and Web service recommendation. Section 3 presents the system architecture. Section 4 describes the proposed Web service recommendation algorithm. Section 5 shows our extensive experiment results, employing QoS values of real-world Web services, and Section 6 concludes the paper.

2 Related Work

2.1 Collaborative Filtering

Collaborative Filtering (CF) is widely employed in commercial recommender systems, such as Netflix and Amazon.com [4], [18], [19], [22]. The basic idea of CF is to predict and recommend potential favorite items for a particular user employing rating data collected from other users. 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], [11], [15], item-based approaches [9], [18], [23], and their fusion [27]. User-based approaches predict the ratings of users based on the ratings of their similar users, and item-based approaches predict the ratings of users based on the information of item similarity. Memory-based algorithms are easy to implement, require little or no training cost, and can easily take ratings of new users into account. However, memory-based algorithms do not scale well to a large number of users and items due to the high computation complexity.

Model-based CF algorithms, on the other hand, learn a model from the rating data using statistical and machine learning techniques. Examples include clustering models [30], latent semantic models [12], [13], latent factor models [5], and so on. These algorithms can quickly generate recommendations and achieve good online performance. However, these models must be rebuilt when new users or items are added to the system.

2.2 Service Selection and Recommendation

Service selection and recommendation have been extensively studied to facilitate Web service composition in recent years. Wang *et al.* [28] present a Web service selection method by QoS prediction with mixed integer program. Zhang *et al.* [34] provide a fine grained reputation system for QoS-based service selection in P2P system. Zheng *et al.* [37] provide a QoS-based ranking system for cloud service selection. Zhu *et al.* [38] employ clustering techniques to their QoS monitoring agents and provide Web service recommendations based on the distance between each user and their agents. El Hadadd *et al.* [10] propose a selection method considering both the transactional properties and QoS characteristics of a Web service. Hwang *et al.* [14] 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.* [16] propose AWSR system to recommend services based on users' historical functional interests and QoS preferences. Barakat *et al.* [2] model the quality dependencies among services and proposes a Web service selection method for Web service composition. Alrifai and Risse [1] propose a method to meet users' end-to-end QoS requirements employing integer programming (MIP) to find the optimal decomposition of global QoS constraints into local constraints.

A certain amount of work has been done to apply CF to Web service recommendation. Shao *et al.* [24] employ a user-based CF algorithm to predict QoS values. Works in [17], [25] apply the idea of CF in their systems, and use MovieLens data for experimental analysis. Combination tasks of different types of CF algorithms are also engaged in Web service recommendation. Zheng *et al.* [36] combine user-based and item-based CF algorithms to recommend Web services. They also integrate Neighborhood approach with Matrix Factorization in their work [35]. Yu [32] presents an approach that integrates matrix factorization with decision tree learning to bootstrap service recommender systems. Meanwhile, several tasks employ location information to Web service recommendation. Chen *et al.* [7]

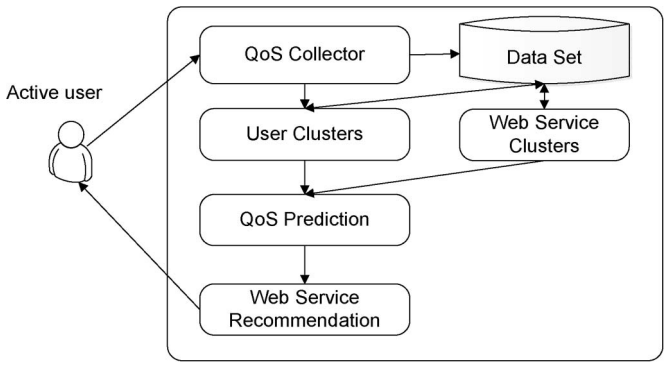


Fig. 1. System overview of LoRec.

use a region-based CF algorithm to make Web service recommendation. To help users know more about Web service performance, they also propose a visualization method showing recommendation results on a map. Lo *et al.* [31] employ the user location in a matrix factorization model to predict QoS values. Different from existing work, this paper interprets Web service QoS information from both user's perspective and Web service's perspective. Clustering technique and location information are employed to achieve more accurate recommendation result and better online performance. Experiments in Section 5 demonstrate the result of the proposed method.

3 Preliminary

3.1 System Overview

Web 2.0 applications such as social networking sites and self-publishing sites encourage users to share their knowledge and learn from others. LoRec employs the idea of *user collaboration* and provides a platform for users to share observed Web service QoS values and search Web services. This system will generate personalized service recommendations based on user shared QoS values. The more QoS records users contribute, the more accurate the recommendations will be, since more information can be mined from the user-contributed QoS values. In this paper, we assume that users are trustworthy. How to detect and handle malicious users and inaccurate QoS values will be addressed in our future work. Fig. 1 shows the architecture of our LoRec recommender system, which includes the following procedures:

- Web service users log on to LoRec system and share observed Web service QoS records with other users. In this paper, users who have submitted Web service QoS records to LoRec are called *training users*. If a training user requires Web service recommendation, then the user becomes an *active user*. QoS values of training users will be employed to make personalized recommendation for the active user.
- LoRec clusters training users into different regions according to their physical locations and past Web service usage experiences (details will be introduced in Section 4.1).
- LoRec clusters functionally similar Web services based on their QoS similarities (details will be introduced in Section 4.2).
- LoRec maps the active user to a user region based on historical QoS and user location (details will be introduced in Section 4.3).
- The recommender system predicts QoS values of candidate Web services for the active user and recommend the best one. (details will be introduced in Section 4.3).
- The active user receives the predicted QoS values of Web services as well as the recommendation results, which can be employed to assist decision making (e.g., service selection, service composition, service ranking, etc.).

Table 1 shows an example of one QoS property in LoRec data set. There are five users (rows) and seven services (columns). Each value in the table stands for the response time of a Web service observed by a user, and ''?'' indicates that the user has not used the service yet. Assume *Amy* is an active user who needs to pick one service with low latency among three candidates, *Service 2*, *Service 4*, and *Service 5*. LoRec will make personalized response time predictions for these three services by using response time values submitted by training users (i.e., *Bob, Carol, David, and Edward*), and recommend the one with best predicted response time value to Amy. LoRec stores different QoS property records separately, which means that for different QoS properties you will find different tables like Table 1. If Amy wants a service with low latency and high availability, LoRec will search both response time table and availability table and predict two property values separately for all candidate services and recommend the best for Amy.

3.2 Region Definition and Features

3.2.1 User Regions and Service Regions

Given a recommender system consisting of m 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 on Web service i. If user u has never used Web service i before, then $r_{u,i} = null$.

A *service region* is a group of services with similar QoS performance. In LoRec, service regions are used to discover potential services and recommend them to active users. A *user region* is defined as a group of users who are closely located with each other and have similar Web service QoS usage experience. Each user belongs to exactly one region. Building regions help LoRec identify relationships in the QoS data set that might not be logically derived through casual observation. Details of building user regions and service regions are presented in Section 4.

3.2.2 Region Centers

Region center is a feature employed by both user region and service region. A user region center reflects the average performance of Web services observed by a set of similar users who belong to one region. A user region center is defined as the median vector of all QoS vectors associated with the region users (row vectors in Table 1). Median is the numeric value separating the higher half of a sample from the lower half. When there is an even number of samples, the median is defined to be the mean of the two middle values. The ith element of the median vector of a region

TABLE 1
Example of LoRec Data Storage

User	Location	Service 1	Service 2	Service 3	Service 4	Service 5	Service 6	Service 7
Amy	Beijing, CN	20000ms	?	2000ms	?	?	?	2800ms
Bob	Houston, US	600ms	3300ms	?	3300ms	2000ms	?	?
Carol	Houston, US	650ms	2600ms	200ms	?	?	?	?
David	Houston, US	620ms	2500ms	2000ms	500ms	?	2000ms	?
Edward	Hong Kong, CN	1000ms	2500ms	2000ms	5000ms	?	2400ms	?

center represents the median QoS value of the ith service observed by users in the region. For example, suppose a user region consists of Bob, Carol, and David (see Table 1). The response time dimension of the user region center will be (620, 2600, 1100, 1900, 2000, 2000, null). Similarly, a service region center is defined as the median QoS vector of all services (column vectors in Table 1). It reflects the average QoS values of a set of similar services that each user may experience. Suppose Services 2, 3, and 6 form one service region, the response time dimension of the service region center will be (2000, 3300, 1400, 2000, 2400) which means that for Amy, David and Edward, the average response time of Services 2, 3, and 6 will be 2000 ms; for Bob, it will be 3300 ms and 1400 ms for Carol.

3.2.3 Sensitive Web Services

Besides region centers, QoS fluctuation is another feature that deserves attention. From a large scale real data analysis, we discover that some QoS properties (e.g., response time) usually varies from one user region to another. Some services have unexpected long response time in certain user regions, and some services are even inaccessible to a few user regions. Inspired by the three-sigma rule which is often applied to test outliers, we employ a similar method to distinguish services with unstable performance and regard them as user region-sensitive services.

For ease of discussion, let's pick one QoS property r (i.e., response time) as an example. 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. 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

$$\begin{aligned} med &= median_i(r_{i,s}), i = 1, \ldots, k; \\ MAD &= median_i\big(|r_{i,s} - med|\big), i = 1, \ldots, k. \end{aligned} \quad (1)$$

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

$$\widehat{\mu} = median_i(r_{i,s}), i = 1, \ldots, k \quad (2)$$

$$\widehat{\sigma} = MAD_i(r_{i,s}), i = 1, \ldots, k. \quad (3)$$

Definition 1. *Let $r_{.s} = \{r_{1,s}, r_{2,s}, \ldots, r_{k,s}\}, 1 \leq k \leq m$ be the set of non-zero response times of Web service s provided by users. Service s is a sensitive service to region M iff $\exists r_{j,s}$, where $(r_{j,s} > \widehat{\mu} + 3\widehat{\sigma}) \wedge (region(j) = M)$.*

In the above definition, $\widehat{\mu}$ and $\widehat{\sigma}$ can be calculated by Eqs. (2) and (3), and the $region(j)$ function identifies the region of user j. The basic idea is that if a user observed QoS is so different from others, we will pay special attention when recommending this service to other users. Take Service 1 from Table 1 as an example, the user-observed response time values are {600, 620, 650, 1000, 20000}. Compared with others, Amy observed response time is unacceptable and deviates greatly from the median value 650. Intuitively, we want to find a way to distinguish this service from others for Amy. With Eqs. (2) and (3), we find that $\widehat{\mu} = 650$ and $\widehat{\sigma} = 50$. It is obvious that $20000 > 650 + 3 \times 50$, and Service 1 is sensitive to Amy's region. Furthermore, if some users from Amy's region log on to LoRec and require service recommendation, it is unlikely that Service 1 will be highly recommended.

Definition 2. *The sensitivity of a region is the fraction between the number of sensitive services in the region over the total number of services.*

Definition 3. *A region is a sensitive region iff its region sensitivity exceeds the predefined sensitivity threshold λ.*

Identifying a region's sensitive services is an important step to make personalized Web service recommendations. With that information, LoRec can make more accurate QoS predictions and provide proper Web services to different users.

3.3 Region Similarity

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 QoS values of Web services they both invoked

$$Sim(a,u) = \frac{\sum_{i \in I}(r_{a,i} - \overline{r}_a)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I}(r_{a,i} - \overline{r}_a)^2}\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r}_u)^2}}, \quad (4)$$

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 two users have similar Web service usage experiences, while negative PCC value means that the Web service usage experiences are opposite. $Sim(a,u) = null$ when two users have no commonly invoked Web service.

PCC only considers the QoS difference between services invoked by both users, which may overestimate the similarity of two users that are not similar but happen to have a few services with very similar QoS records. To

devalue the overestimated similarity, a correlation significance weight can be added [36]. An adjusted PCC for user similarity is defined as

$$Sim'(a,u) = \frac{2 \times |I_a \cap I_u|}{|I_a| + |I_u|} Sim(a,u), \quad (5)$$

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 $[-1,1]$, and the value of $Sim'(a,u)$ is in the interval of $[-1,1]$.

Similar to the way of clustering users, LoRec clusters Web services based on their QoS performance to find underlying relationships. PCC is used to measure the similarity between Web services in LoRec as well. The similarity of two Web services i and j can be calculated by

$$Sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r}_i)(r_{u,j} - \overline{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r}_i)^2} \sqrt{\sum_{u \in U}(r_{u,j} - \overline{r}_j)^2}}, \quad (6)$$

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, and $\overline{r}_i$ 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 has used both services. The adjusted similarity value is defined as:

$$Sim'(i,j) = \frac{2 \times |U_i \cap U_j|}{|U_i| + |U_j|} Sim(i,j), \quad (7)$$

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

4 Methodology

Values of some QoS properties (e.g., response time) on the same Web service vary quite differently from user to user. Through the analysis of a real world Web service QoS data set[2] (see Section 5 for details), which contains 1.5 millions service invocation records evaluated by users from more than twenty countries, we find that some QoS properties highly relate to the physical locations of users.

For example, the response time of a service observed by closely located users usually fluctuates mildly around a certain value. On the other hand, the response time observed by users who are far away from each other sometimes varies significantly. Based on this finding, our recommendation algorithm takes location information into consideration to improve the recommendation accuracy. Our recommendation algorithm is designed as a three-phase process, i.e., 1) user region creation, 2) service region creation, and 3) QoS prediction & recommendation, which will be presented in Section 4.1 to Section 4.3, respectively.

4.1 Phase 1: User Region Creation

In this phase, users will be clustered into different regions according to their locations and historical QoS records. At the beginning, we retrieve users' approximate locations by their IP addresses.[3] The location information reveals a user's country, city, latitude/longitude, ISP and domain name. Then users from the same city will be grouped together to form initial regions. These small regions will be aggregated into large ones with a bottom-up hierarchical clustering method [20].

The clustering method has two parts: initialization and aggregation. In the initialization part, we select non-sensitive user regions for aggregation, and compute the similarity between each region pair with Eq. (5). To aggregate regions,

1. Select the most similar region pair $(region_i, region_j)$, merge the two regions to $region_i$ if their similarity exceeds the similarity threshold μ_u, otherwise stop this region aggregation process. To merge the two regions,
   a. Compute the sensitivity and region center of this newly merged region $region_i$. Remove this region from aggregation process if it becomes a sensitive one.
   b. Remove similarities between $region_j$ and other existing regions.
   c. Update similarities between $region_i$ and other existing regions.
2. Repeat the above step.

Threshold μ_u is a tunable parameter that can be adjusted to trade off accuracy for time and space requirements. μ_u's impact on prediction accuracy will be addressed in Appendix A.

4.2 Phase 2: Service Region Creation

Normally, each user only uses a limited amount of Web services. Compared with the large number of services on the Internet, the number of services with user submitted QoS records is relatively small. Thus, it is difficult to find similar users, and predicting missing QoS values only from user's perspective is not enough. Clustering Web services can help LoRec find potential similar services. Different from retrieving user location from an IP address, LoRec directly clusters Web services based on their QoS similarity. This is because some companies regard the physical location of data center as a secret and use IP address to hide the real locations. Take Google for example. It has data centers located in Asia, Europe, America, etc, but physical locations retrieved from Google's IP addresses used in different country-specific versions of Google Search are all listed to Mountain View, California. Another reason is due to the use of the distributed system architecture. To enhance user interaction and to minimize delay, service

providers will route user requests to different servers according to user locations or application types. Usually the server that processes requests is different from the one that responds to the users. Thus, retrieving a service location from an IP address does not prove much value.

In LoRec, Web services are aggregated with a bottom-up hierarchical clustering algorithm. We use median vector rather than mean vector as the cluster center to minimize the impact of outliers. The similarity between two clusters is defined as the similarity of their centers. Each Web service is regarded as a cluster at the outset. The algorithm aggregates the pairs of the most similar clusters until none of the pairs' similarities exceeds threshold μ_w.

4.3 Phase 3: Personalized QoS Prediction

The first two phases aggregate users and Web services into a certain number of clusters based on their respective similarities. QoS predictions can be generated from both service regions and user regions. With the compressed QoS data, searching neighbors and making Web service QoS predictions for an active user can be computed faster than conventional methods.

4.3.1 Prediction from User Perspective

Instead of computing the similarity between the active user and each training user, we only compute the similarity between the active user and each region center. Moreover, 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 QoS value of Web service s for active user a, we take the following steps:

- Identify the user region of active user a by IP address. The active user will be treated as a member of a new region if no appropriate region is found.
- If service s is sensitive to user a's region, 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

$$\widehat{r_{a,s}} = r_{c,s}. \tag{8}$$

- For non-sensitive services, the prediction value $\widehat{r_{u_{a,s}}}$ will be generated considering QoS values submitted from similar regions. Eq. (5) is employed to calculate the similarity between the active user and each region center that has evaluated service s. Up to k most similar centers with positive PCC values $c_1, c_2, \ldots, c_k$ will be employed. We discuss how to choose k (also called top k) in Appendix A.
- If the active user's region center has QoS value of s, the prediction is computed using the following equation:

$$\widehat{r_{u_{a,s}}} = r_{c,s} + \frac{\sum_{j=1}^{k} Sim'(a, c_j)(r_{c_j,s} - \overline{r_{c_j\cdot}})}{\sum_{j=1}^{k} Sim'(a, c_j)}, \tag{9}$$

where $r_{c_j,s}$ is the QoS of service s provided by center c_j, and $\overline{r_{c_j\cdot}}$ 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 users of this region. 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 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

$$\widehat{r_{u_{a,s}}} = \frac{\sum_{j=1}^{k} Sim'(a, c_j) r_{c_j,s}}{\sum_{j=1}^{k} Sim'(a, c_j)}. \tag{10}$$

4.3.2 Prediction from Service Perspective

Clustering Web services provides another way to view and utilize the data set. It can enhance the prediction accuracy when we only have limited knowledge of user preference. To predict the QoS value of service s observed by user a from the service perspective, we use the Web service cluster center value of user a as a rough prediction if the center has the record of a; otherwise, we do not predict from the service perspective. According to our experiment, good prediction accuracy is achieved with this rough prediction. To achieve a better prediction result, we can tune the result by using Eq. (11)

$$\widehat{r_{i_{a,s}}} = r_{a,c} + \frac{\sum_{j=1}^{k} Sim'(s, c_j)(r_{a,c_j} - \overline{r_{c.j}})}{\sum_{j=1}^{k} Sim'(s, c_j)}, \tag{11}$$

where $r_{a,c}$ is the Web service cluster center value of user a, $Sim'(s, c_j)$ measures the similarity between Web service s and service center c_j. r_{a,c_j} is the QoS of user a from service center c_j, and $\overline{r_{c.j}}$ is the average QoS of service center c_j. Up to k similar service cluster centers will be employed to predict the value.

4.3.3 Prediction Generation

For user a, the final prediction QoS of service s consists of two parts: prediction from user perspective and from service perspective

$$\widehat{r_{a,s}} = \omega \times \widehat{r_{u_{a,s}}} + (1 - \omega) \times \widehat{r_{i_{a,s}}}, \tag{12}$$

where $\widehat{r_{u_{a,s}}}$ is the QoS prediction generated from user regions, $\widehat{r_{i_{a,s}}}$ is the prediction from Web service clusters, and parameter ω determines how much we rely on each prediction result, which ranges from [0, 1].

4.4 Phase 4: Web Service Recommendation

Web service QoS prediction is used in different ways in LoRec to facilitate Web service recommendation. First, when a user searches Web services using LoRec, predicted QoS values will be shown next to each candidate service, and the one with the best predicted value will be highlighted in the search result for the active user. It will be easier for the active user to decide which one to have a try. Moreover, LoRec selects the best performing services (services with the best submitted QoS) and services with the best predicted QoS from the whole service repository

for the active user so that he/she can quickly find potential valuable ones instead of checking the service one by one.

4.5 Time Complexity Analysis

We discuss the worst case time complexity of LoRec recommendation algorithm. We analyze the clustering phase and QoS value prediction phase in Sections 4.5.1 and 4.5.2 respectively. We assume the input is a full matrix with m users and n Web services.

4.5.1 Time Complexity of Clustering

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

In terms of the user region aggregation part, we assume there are l_0 user regions in the beginning. Since there are at most n services used by both regions, the time complexity of the region similarity is $O(n)$ using Eq. (5). We use a matrix to store the similarity between each two regions, and the complexity for computing similarity matrix is $O(l_0^2 n)$.

The aggregation of two user regions will be executed 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 user regions that can be aggregated decreases with each iteration, the real search time will be less than $l_0 - 1$ in the following iterations. For the selected pair of user regions, we calculate the new center and update their similar user regions. Because the number of users involved in the two user regions is uncertain, we use the number of all users as the upper bound and the complexity is $O(mn \log m)$. We employ the priority queue to sort similar user regions, and the insertion and deletion of a similar region is $O(\log l_0)$. Thus, the time complexity is $O(l_0^2(\log l_0 + mn \log m)) = O(l_0^2 mn \log m)$. As the above steps are linearly combined, the total time complexity of user clustering is $O(l_0^2 mn \log m)$.

In the phase of service region creation, there are n services at the beginning. The aggregation of two service regions will be executed at most $n - 1$ times, in case that all services are merged into one cluster. In each iteration, we first compare at most $n - 1$ heads of the priority queues to find the most similar pairs. Since the number of clusters that can be aggregated decreases with each iteration, the real search time will be less than $n - 1$ in the following iterations. For the selected pair, we calculate the new center and update their similar clusters. Because the number of services involved in two clusters is uncertain, we use the number of all services as the upper bound and the complexity is $O(mn \log n)$. The insertion and deletion of a similar region is $O(\log n)$, since we employ the priority queue to sort similar regions. Thus, the time complexity is $O(n^2(\log n + mn \log n)) = O(mn^3 \log n)$.

4.5.2 Time Complexity of QoS Prediction

Let l_1 be the number of user regions after the region creation. To predict QoS value for an active user, $O(l_1)$ similarity calculations between the active user and user 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 that 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. (9) and Eq. (10) show. 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 prediction from the user region perspective 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). Similarly, the online prediction from service region perspective is $O(l_2 n) \approx O(n)$, where l_2 is the number of service regions. Compared to the memory-based CF algorithm used in previous work with $O(mn)$ online time-complexity, our approach is more efficient and better suited for large data set, and the corresponding experiments confirm this in Section 5.

5 Experiments

5.1 Experiment Setup

In this experiment, we crawl publicly available Web services from three sources 1) well-known companies (e.g., *Google, Amazon, ect.*); 2) portals listing publicly available Web services (e.g., *xmethods.net, webservicex.net, etc.*); and 3) Web service search engines (e.g., *seekda.com, esynaps.com, etc.*). Java classes are generated using *WSDL2Java* tool of *Axis2* package.

To obtain QoS values of Web services, we employ 150 computers in 24 countries from Planet-Lab [8] to monitor 100 real Web services in 22 countries. About 1.5 millions Web service invocation records are collected in two days' time. For each user (a computer node from Planet-Lab), there are around 100 profiles, and each profile contains the response time (also called Round Trip Time, RTT) records of 100 services. We randomly extract 20 profiles from each node, and generate 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 use a small number of Web services in reality.

We apply Mean Absolute Error (MAE) to measure the prediction accuracy of the recommendation algorithm. The more accurately the algorithm predicts, the better the recommendations are. MAE is the average absolute deviation of predictions to the ground truth data, where smaller MAE indicates better prediction accuracy

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \widehat{r}_{i,j}|}{N}, \tag{13}$$

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

TABLE 2
Time Usage Comparison of Online QoS Prediction

Method	Online Time Duration (s)
IPCC	2.437
UPCC	5.218
WSRec	6
RegionKNN	0.094
LoRec	0.141

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.

5.2 Performance Evaluation

To study the prediction accuracy, we compare our approach with an item-based prediction algorithm using PCC (*IPCC*) [23], a user-based prediction algorithm using PCC (*UPCC*) [3], WSRec [36], RegionKNN [6].

We randomly remove 90 percent and 80 percent RTTs of the initial training matrix to generate two sparse matrices with density 10 percent and 20 percent respectively. We vary the number of RTT values submitted by active users from 10, 20 to 30 and name them G10, G20, and G30 respectively. The removed records of active users are used to study the prediction accuracy. In this experiment, we set $\mu_u = 0.3$, $\mu_w = 0.1$, $\lambda = 0.8$, $\omega = 0.1$, and $topk = 10$. To get a reliable error estimate, we use 10 times 10 fold cross-validation [29] to evaluate the prediction accuracy and report the average MAE value.

The experiment is conducted on a laptop with Intel Centrino Duo processor (1.836 Hz), 2GB memory, and Window XP SP3 system. Table 2 shows the online time usage of each algorithm predicting 27000 missing QoS values for 300 users (one fold), and each user in that set submits 10 QoS values with 90 missing ones. Obviously, LoRec requires less time than memory-based methods (IPCC, UPCC, and WSRec) to perform online prediction and can scale well for large data sets.

Table 3 shows the prediction performance of different methods employing 10 percent and 20 percent density training matrices. To see how location information improves the accuracy, we also compare LoRec with CBRec, a similar method but removing the location information, sensitive services and sensitive regions concepts. It shows that LoRec significantly improves the prediction accuracy and outperforms others consistently. Performance of all recommendation approaches enhances with the increasing number of QoS provided by active users, from 10 to 30 (G10, G20, G30). On the other hand, the density of training matrix also impacts the performance. All approaches have better prediction accuracy with training matrix density 20 percent than with density 10 percent. Moreover, the approaches employing user similarity to generate recommendations are more sensitive to the amount of data provided by users. For example, the performance of UPCC and WSRec enhances significantly with the QoS values submitted by active users (the given number). IPCC stays stable, since IPCC only employs service similarity instead of user similarity.

5.3 Impact of Data Sparseness

Compared with the amount of services on the Internet, the number of services consumed by each user is small. The data set of recommender systems is usually sparse. We examine how data sparseness impacts the prediction results from two aspects: the density of training matrix which indicates how many QoS records are collected from all users, and the number of QoS values given by active users (the given number).

We first study the impact of training matrix density. We vary the density of the training matrix from 10 percent to 50 percent with a step of 10 percent, and given $= 10$. For parameters of LoRec, we set topk $= 10$, $\omega = 0.1$, $\mu_w = 0.1$, $\lambda = 0.8$, $\mu_u = 0.3$ with data sets of density 10 percent, 20 percent, and 30 percent, $\mu_u = 0.6$ with data sets of density 40 percent and 50 percent. Fig. 2a shows the experimental results. It shows that: 1) With the increase of the training matrix density, the performance of IPCC, UPCC, RegionKNN and LoRec enhances, indicating that a better prediction is achieved with more QoS data. WSRec is not sensitive to the data sparseness, and it stays around a certain value. 2) LoRec outperforms others consistently.

To study the impact of the given number on the prediction quality, we employ the training matrix with density 30 percent and vary the given number from 10 to 50 with a step of 10. Fig. 2b shows the experimental results. It reflects that the prediction performance of IPCC, UPCC, and WSRec generally grows with the increasing given number. The prediction of LoRec improves with the given number at first, but then it does not have a steady improvement when the given number exceeds 30. The above two experiments show that users are more likely to have better prediction result when they contribute more data records to LoRec. For more information on how other parameters impact the accuracy, please refer to Appendix A.

6 Conclusion

This paper presents a QoS-aware Web service recommendation approach. The basic idea is to predict Web service QoS values and recommend the best one for active users

TABLE 3
MAE Comparison on Response Time (Smaller Value Means Better Prediction Accuracy)

	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
CBRec	740.25	720.41	703.25	664.18	658.30	652.38
RegionKNN	638.21	624.51	623.90	573.85	560.13	556.75
LoRec	**584.32**	**561.27**	**557.95**	**542.11**	**523.33**	**506.86**

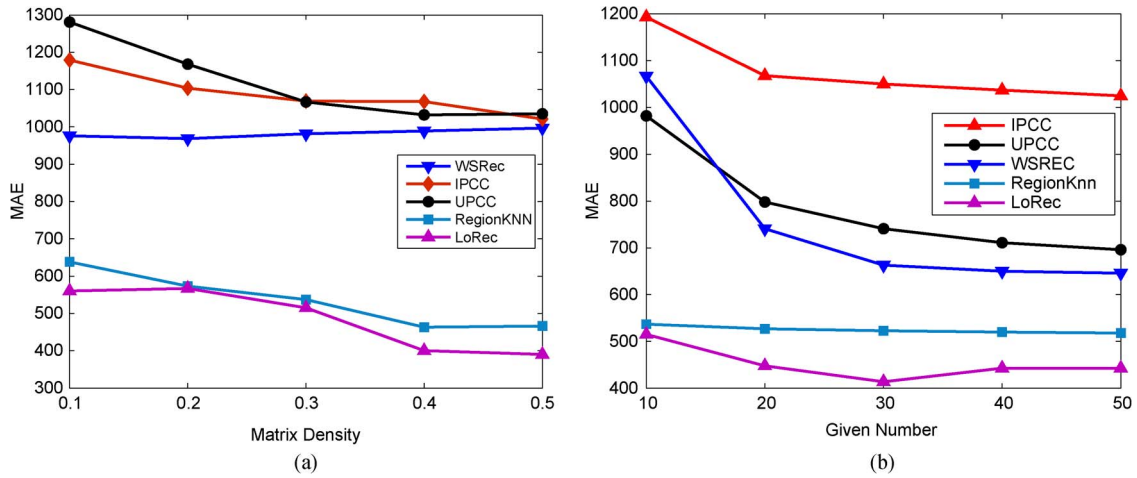


Fig. 2. Training matrix density's impact on prediction accuracy.

based on historical Web service QoS records. We combine prediction results generated from service regions and user regions, which achieves better results than existing approaches. We also find that the combination result is much better than the result from any single method, either the prediction generated from user regions or the one generated from Web service regions. This is because these two methods analyze the problem from different aspects and the combination of them counteracts the error of individual methods.

In our future work, we will consider several aspects to further improve the proposed Web service recommendation approach. In terms of the clustering method, we will consider probabilistic ones like EM to improve the scalability of LoRec. EM only requires one scan of the database with limited memory. For 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 recommended services is another task to investigate. Our future work also includes investigating the correlation between different QoS properties, and detecting malicious users with inaccurate QoS information.

Appendix A
Experiment on Parameter Impacts

A.1 Parameter Impact on Clustering

In phase one, users are clustered into regions based on similarity and physical location. Two thresholds λ and μ_u determine the number of regions that are created. As mentioned in Section 4.1, only regions with similarity higher than μ_u and sensitivity less than λ can be aggregated into one region.

To study the single impact of μ_u on prediction accuracy, we set given $= 20$, $\mu_w = 0.1$, $\omega = 0.1$, $\lambda = 0.2$ and topk $= 10$ for QoS prediction. We vary μ_u from 0.1 to 0.9 with a step of 0.1. Fig. 3a shows the relation between μ_u and prediction

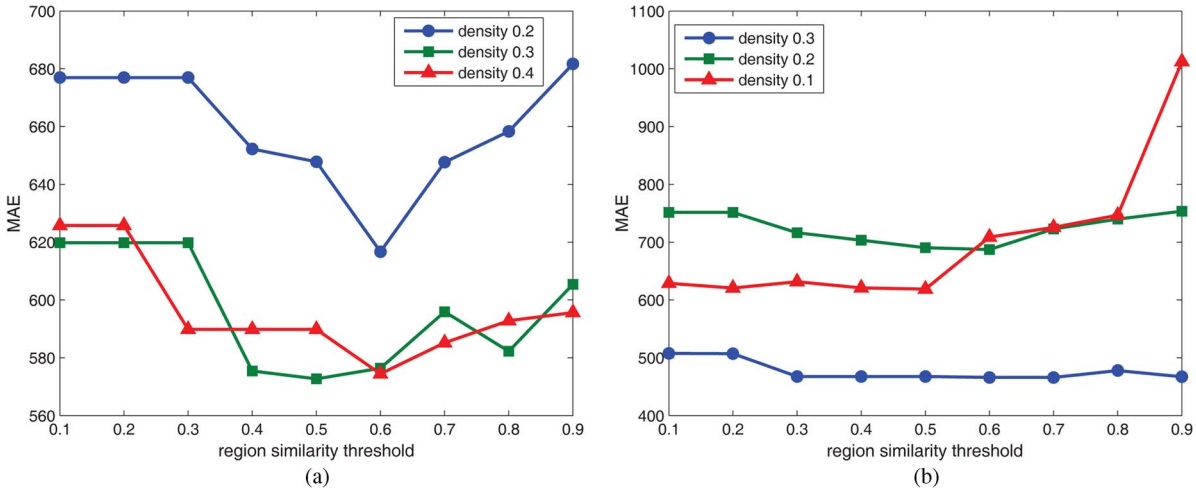


Fig. 3. Impact of μ_u on prediction accuracy.

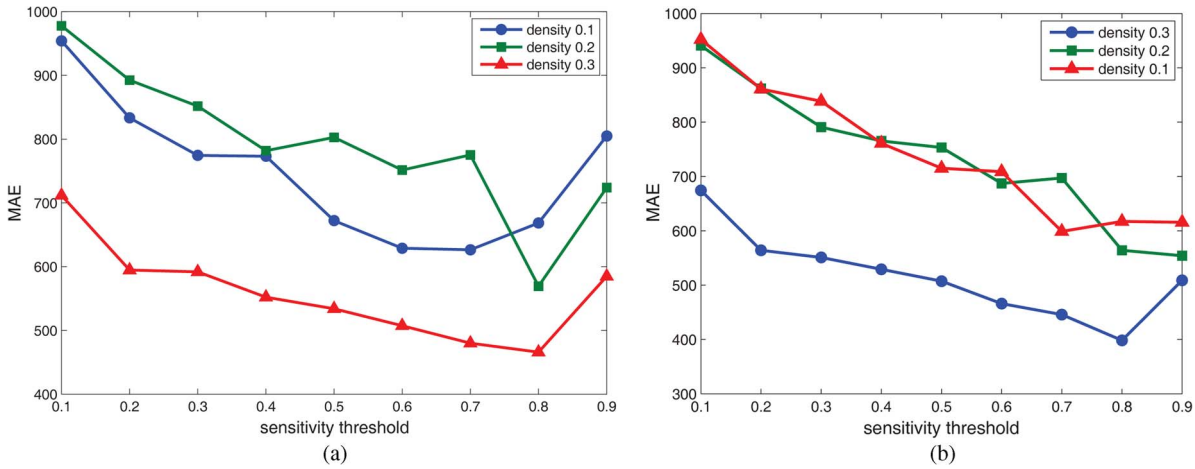


Fig. 4. Impact of λ on prediction accuracy.

accuracy with three training matrices density 20 percent, 30 percent, and 40 percent. The prediction quality enhances as μ_u grows at first, because higher μ_u helps obtain a set of coherent regions, and better prediction is obtained from very similar users. However, when μ_u grows beyond a certain value (0.6 in this experiment), the prediction quality fluctuates. We can see that the fluctuation is more severe with a sparse data set than with a dense data set. We find similar results with different λ values. As Fig. 3b shows, we set $\lambda = 0.6$ and keep other parameter settings the same. We employ three matrices with density 10 percent, 20 percent, and 30 percent respectively. We can see that the performance with density 10 percent matrix dramatically fluctuates, while the performance of others mildly changes. This is because when it is difficult to find very similar users to generate user based predictions, the final prediction results will only come from service based predictions.

To investigate the single impact of λ on prediction quality, we employ three data sets with density 10 percent, 20 percent, and 30 percent respectively. Each data set contains 2700 training users and 300 active users. We set given $= 20$, $\mu_w = 0.1$, $\omega = 0.1$, and topk $= 10$ for QoS prediction. Figs. 4a and 4b show the results with $\mu_u = 0.1$ and $\mu_u = 0.6$ respectively. Higher λ allows similarly sensitive regions to be aggregated into one region, and achieves better prediction result. Note that the optimal value of λ is related to the sensitivity of the original regions at the outset. For the full data set in the experiment, if we treat each user as a region, 4.67 percent are with sensitivity around 0.8 and 81.3 percent are with sensitivity less than 0.1.

A.2 Impact of Topk

Topk 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 30 percent, and set $\lambda = 0.3$, $\mu_u = 0.8$, $\mu_w = 0.2$, and $\omega = 0.1$. After the clustering phase, we obtain 42 user regions. To study the impact of neighborhood size, we vary topk from 5 to 40 with a step of 5. Fig. 5 shows the result with the given number from 10 to 30. The trends of the three curves are similar, which show that MAE decreases sharply with an increasing neighborhood size at the beginning, and then stays around a certain value. As topk grows, more regions that are not very similar will be considered in QoS prediction, and these regions make little contribution or even add noise to the final result.

A.3 Impact of ω

Different data sets have different data characteristics. Parameter ω makes our prediction method more flexible and adaptable to different data sets. If $\omega = 1$, we make prediction mainly based on user information, and if $\omega = 0$, we only consider valuable information from Web services. In other cases, we leverage both similar users and services to predict missing values for active users.

To study the impact of ω on our collaborative filtering method, we use data sets with 2700 training users and 300 active users. We set Topk $= 10$, $\mu_w = 0.1$ and $\mu_u = 0.6$. We vary ω value from 0.1 to 1 with a step of 0.1. As Fig. 6a shows, the first experiment employs three training matrices with density 10 percent, 20 percent, and 30 percent respectively, and each active user provides 20 records to the recommender system. It is obvious that ω has an impact on the prediction accuracy especially when the matrix is not that sparse. The result indicates that the prediction accuracy is very stable with matrix of 10 percent density.

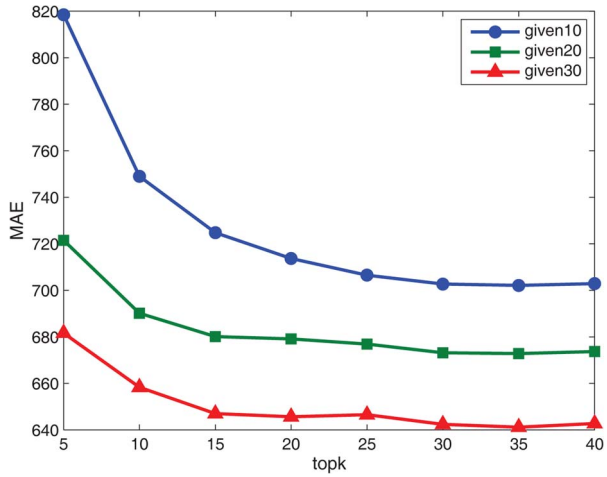


Fig. 5. Impact of topk on prediction accuracy.

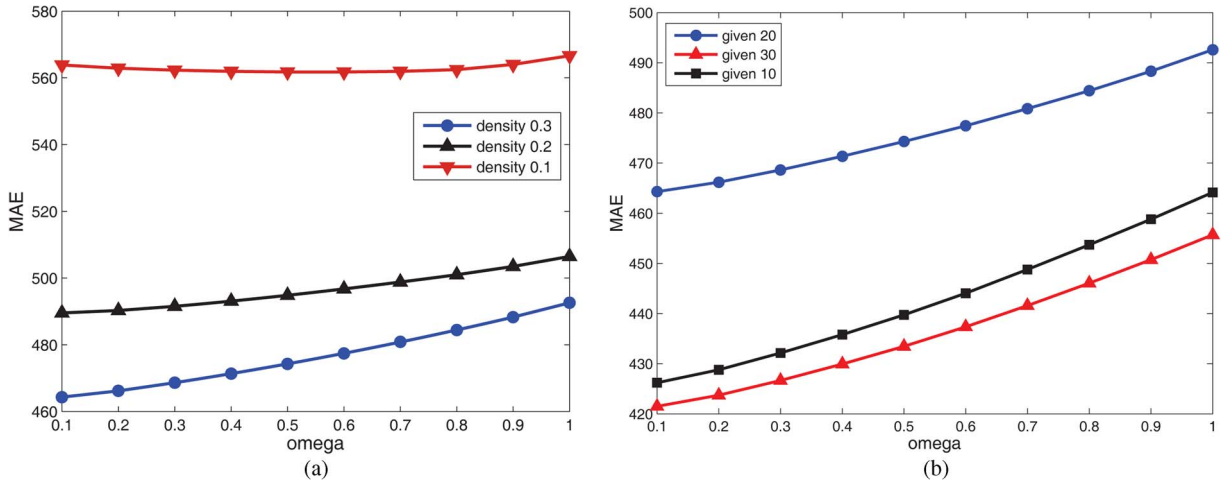


Fig. 6. Impact of ω.

However, for a dense data set, a better prediction accuracy is achieved with smaller ω, which means more information provided by similar Web services is used.

Another experiment is to study the impact of ω with different given number. As Fig. 6b shows, we employ the training matrix with 30 percent density, and set the given number 10, 20, and 30. Similarly, a better prediction result is achieved when we employ more information from similar Web services.

Acknowledgment

This work was supported by the National Basic Research Program of China (973 Project 2014CB347701), the National Natural Science Foundation of China (Project 61100078), the Shenzhen Basic Research Program (Project JCYJ20120619153834216), and the Research Grants Council of the Hong Kong Special Administrative Region, China (No. N_CUHK405/11 of the NSFC/RGC Joint Research Scheme, No. 415212 of the General Research Fund). Z. Zheng is the corresponding author.

References

[1] M. Alrifai and T. Risse, ''Combining Global Optimization with Local Selection for Efficient QoS-Aware Service Composition,'' in *Proc. 18th Int'l Conf. WWW*, 2009, pp. 881-890.

[2] L. Barakat, S. Miles, and M. Luck, ''Efficient Correlation-Aware Service Selection,'' in *Proc. IEEE 19th ICWS*, 2012, pp. 1-8.

[3] J.S. Breese, D. Heckerman, and C. Kadie, ''Empirical Analysis of Predictive Algorithms for Collaborative Filtering,'' in *Proc. 14th Annu. Conf. UAI*, 1998, pp. 43-52.

[4] R. Burke, ''Hybrid Recommender Systems: Survey and Experiments,'' *User Model. User-Adapt. Interact.*, vol. 12, no. 4, pp. 331-370, Nov. 2002.

[5] J. Canny, ''Collaborative Filtering with Privacy via Factor Analysis,'' in *Proc. 25th Int'l ACM SIGIR Conf.*, 2002, pp. 238-245.

[6] X. Chen, X. Liu, Z. Huang, and H. Sun, ''RegionKNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation,'' in *Proc. IEEE 8th ICWS*, 2010, pp. 9-16.

[7] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, ''Personalized QoS-Aware Web Service Recommendation and Visualization,'' *IEEE Trans. Serv. Comput.*, vol. 6, no. 1, pp. 35-47.1st Quart., 2013.

[8] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, ''Planetlab: An Overlay Testbed for Broad-Coverage Services,'' *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3-12, July 2003.

[9] M. Deshpande and G. Karypis, ''Item-Based Top-n Recommendation Algorithms,'' *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143-177, Jan. 2004.

[10] J.E. Haddad, M. Manouvrier, and M. Rukoz, ''TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition,'' *IEEE Trans. Serv. Comput.*, vol. 3, no. 1, pp. 73-85, Jan./Mar. 2010.

[11] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, ''An Algorithmic Framework for Performing Collaborative Filtering,'' in *Proc. 22nd Int'l ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, 1999, pp. 230-237.

[12] T. Hofmann, ''Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis,'' in *Proc. 26th Int'l ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, 2003, pp. 259-266.

[13] T. Hofmann, ''Latent Semantic Models for Collaborative Filtering,'' *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 89-115, Jan. 2004.

[14] S.-Y. Hwang, E.-P. Lim, C.-H. Lee, and C.-H. Chen, ''Dynamic Web Service Selection for Reliable Web Service Composition,'' *IEEE Trans. Serv. Comput.*, vol. 1, no. 2, pp. 104-116, Apr./June 2008.

[15] R. Jin, J. Chai, and L. Si, ''An Automatic Weighting Scheme for Collaborative Filtering,'' in *Proc. 27th Int'l ACM SIGIR Conf.*, 2004, pp. 337-344.

[16] G. Kang, J. Liu, M. Tang, X. Liu, B. Cao, and Y. Xu, ''AWSR: Active Web Service Recommendation Based on Usage History,'' in *Proc. IEEE 19th ICWS*, 2012, pp. 186-193.

[17] K. Karta, ''An Investigation on Personalized Collaborative Filtering for Web Service Selection,'' Honours Programme Thesis, Univ. Western Australia, Brisbane, Qld., Australia, 2005.

[18] G. Linden, B. Smith, and J. York, ''Amazon.com Recommendations: Item-to-Item Collaborative Filtering,'' *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76-80, Jan./Feb. 2003.

[19] H. Ma, I. King, and M.R. Lyu, ''Effective Missing Data Prediction for Collaborative Filtering,'' in *Proc. 30th Int'l ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, 2007, pp. 39-46.

[20] C.D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[21] M.R. McLaughlin and J.L. Herlocker, ''A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience,'' in *Proc. 27th Int'l ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, 2004, pp. 329-336.

[22] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, ''GroupLens: An Open Architecture for Collaborative Filtering of Netnews,'' in *Proc. Conf. CSCW*, 1994, pp. 175-186.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, ''Item-Based Collaborative Filtering Recommendation Algorithms,'' in *Proc. 10th Int'l WWW Conf.*, 2001, pp. 285-295.

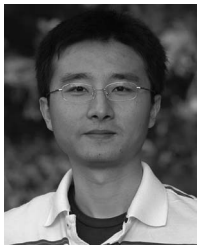
[24] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, ''Personalized QoS Prediction for Web Services via Collaborative Filtering,'' in *Proc. 5th ICWS*, 2007, pp. 439-446.

[25] R.M. Sreenath and M.P. Singh, ''Agent-Based Service Selection,'' *J. Web Semantics*, vol. 1, no. 3, pp. 261-279, Apr. 2003.

[26] N. Thio and S. Karunasekera, ''Automatic Measurement of a QoS Metric for Web Service Recommendation,'' in *Proc. Australian Softw. Eng. Conf.*, 2005, pp. 202-211.
[27] J. Wang, A.P. de Vries, and M.J. Reinders, ''Unifying User-Based and Item-Based Collaborative Filtering Approaches by Similarity Fusion,'' in *Proc. 29th Int'l ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, 2006, pp. 501-508.
[28] S. Wang, Q. Sun, and F. Yang, ''Towards Web Service Selection Based on QoS Estimation,'' *Int'l J. Web Grid Serv.*, vol. 6, no. 4, pp. 424-443, Nov. 2012.
[29] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., Amsterdam, The Netherlands: Elsevier, 2005.
[30] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen, ''Scalable Collaborative Filtering Using Cluster-Based Smoothing,'' in *Proc. 28th Int'l ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, 2005, pp. 114-121.
[31] J. Yin, S. Deng, Y. Li, and Z. Wu, ''Collaborative Web Service QoS Prediction with Location-Based Regularization,'' in *Proceedings of the 19th International Conference Web Services (ICWS'12)*, 2012, pp. 464-471.
[32] Q. Yu, ''Decision Tree Learning from Incomplete QoS to Bootstrap Service Recommendation,'' in *Proc. IEEE 19th ICWS*, 2012, pp. 194-201.
[33] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. New York, NY, USA: Springer-Verlag, 2007, Tsinghua University Press.
[34] Y. Zhang and Y. Fang, ''A Fine-Grained Reputation System for Reliable Service Selection in Peer-to-Peer Networks,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 8, pp. 1134-1145, Aug. 2007.
[35] Z. Zheng, H. Ma, M.R. Lyu, and I. King, ''Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization,'' *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 289-299, July/Sept. 2013.
[36] Z. Zheng, H. Ma, M.R. Lyu, and I. King, ''QoS-Aware Web Service Recommendation by Collaborative Filtering,'' *IEEE Trans. Serv. Comput.*, vol. 4, no. 2, pp. 140-152, Apr./June 2011.
[37] Z. Zheng, X. Wu, Y. Zhang, M.R. Lyu, and J. Wang, ''QoS Ranking Prediction for Cloud Services,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1213-1222, June 2013.
[38] J. Zhu, Y. Kang, Z. Zheng, and M.R. Lyu, ''A Clustering-Based QoS Prediction Approach for Web Service Recommendation,'' in *Proc. 15th IEEE Int'l Symp. Obj./Compon./Serv.-Oriented Real-Time Distrib. Comput. Workshops*, Apr. 2012, pp. 93-98.

Xi Chen received her MS degree in Computer Science and Engineering from Beihang University, Beijing, China, in 2011 and BS degree in Information and Computing Science from Beijing Information Technology Institute, Beijing, China, in 2008. She now works in Schlumberger Technologies (Beijing) Ltd. She received the Google Excellence Scholarship 2010 and served as reviewer for international journals and conferences including TSC, JWSR, and SCC. Her research interests include services computing, cloud computing and data mining.

Zibin Zheng is an associate research fellow at the Shenzhen Research Institute, The Chinese University of Hong Kong. He received his PhD degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, in 2010. He received Outstanding Thesis Award of CUHK at 2012, ACM SIGSOFT Distinguished Paper Award at ICSE2010, Best Student Paper Award at ICWS2010, and IBM PhD Fellowship Award 2010-2011. His research interests include cloud computing, service computing, and software engineering. He is a member of the IEEE.



Qi Yu received the PhD degree in Computer Science from Virginia Polytechnic Institute and State University (Virginia Tech), USA. He is an assistant professor at the College of Computing and Information Sciences of Rochester Institute of Technology, USA. His current research interests lie in the areas of service computing, data mining, and machine learning. His publications have mainly appeared in well-known journals (e.g., *VLDB Journal*, *IEEE TKDE*, and *ACM TWEB*) and conference proceedings (e.g., ICSOC and ICWS). He is a guest editor of the *IEEE TSC* special issue on service query models and efficient selection. He frequently serves as a program committee member on service computing and database conferences (e.g., DASFAA, IEEE Cloud, WISE, CollaborateCom, ICSOC, and IRI). He is also a reviewer for various journals (e.g., the *VLDB Journal*, *ACM TOIT*, and *ACM TWEB*). He is a member of the IEEE.

Michael R. Lyu received the BS degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, in 1981; the MS degree in Computer Engineering from University of California, Santa Barbara, USA, in 1985; and the PhD degree in Computer Science from the University of California, Los Angeles, USA, in 1988. He is currently a Professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, China. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks, Web technologies, multimedia information processing, and E-commerce systems. Dr. Lyu is an IEEE Fellow, an AAAS Fellow, and a Croucher Senior Research Fellow for his contributions to software reliability engineering and software fault tolerance. He received IEEE Reliability Society 2010 Engineering of the Year Award.