SOCA (2016) 10:135–149
DOI 10.1007/s11761-014-0168-4

ORIGINAL RESEARCH PAPER

A Web service QoS prediction approach based on time- and location-aware collaborative filtering

Chengyuan Yu · Linpeng Huang

Received: 11 February 2014 / Revised: 17 October 2014 / Accepted: 23 October 2014 / Published online: 31 October 2014

Abstract In QoS-based Web service recommendation, predicting quality of service (QoS) for users will greatly aid service selection and discovery. Collaborative filtering (CF) is an effective method for Web service selection and recommendation. CF algorithms can be divided into two main categories: memory-based and model-based algorithms. Memory-based CF algorithms are easy to implement and highly effective, but they suffer from a fundamental problem: inability to scale-up. Model-based CF algorithms, such as clustering CF algorithms, address the scalability problem by seeking users for recommendation within smaller and highly similar clusters, rather than within the entire database. However, they are often time-consuming to build and update. In this paper, we propose a time-aware and location-aware CF algorithms. To validate our algorithm, this paper conducts series of large-scale experiments based on a real-world Web service QoS data set. Experimental results show that our approach is capable of addressing the three important challenges of recommender systems–high quality of prediction, high scalability, and easy to build and update.

Keywords Web service · Qos prediction · Time-aware · Location-aware · Collaborative filtering algorithm

1 Introduction

In Web service-oriented environments, Web services are in essence loosely coupled, hosted by different providers, and located in different location. With the exponential growth of Web services, there are many Web services with identical or similar functionalities, but different QoS [1]. Thus, Web services with identical or similar functionalities would be identified by QoS, which has become a critical issue in services computing [2].

On the other hand, the demands of the service consumers vary significantly. It is not possible to fulfill all consumer expectations from the service provider perspective, and hence, a balance needs to be made via a negotiation process. By the end of the negotiation process, provider and consumer reach an agreement. In SOA terms, this agreement is referred to as a service-level agreement (SLA). SLA has been widely used in business systems, e.g., Amazon [3]. SLAs contain service-level objectives (SLOs) and numerical QoS objectives, which the service needs to fulfill. QoS promises are typically defined as legally binding SLAs.

A SLA between a service provider and its customers will assure customers that they can get the service they pay for and will obligate the service provider to achieve its service promises. For cases of violations, SLAs often define monetary penalties. Hence, providers have a strong interest in reducing the number of SLA violations for their services. Thus, a prior identification of SLA violations has become a very important research topic, e.g., [4]. It predicts SLA violations by comparing the predictions of QoS values with existing customer SLAs. The QoS performance of Web services observed from the users' perspective is usually quite different from that declared by the service providers in SLA, mainly due to the following reasons: (1) QoS performance of Web services is highly related to invocation time, since the service status (e.g., workload and number of clients) and the network environment (e.g., congestion) change over time.

C. Yu (✉) · L. Huang
Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
e-mail: ycy8525@sjtu.edu.cn

L. Huang
e-mail: huang-lp@cs.sjtu.edu.cn

For example, in throughput data set 3,[1] which was published in references [5], when user 141 invokes service 4,499 at time interval 20, the throughput is 39.953053. But when the same user invokes the same service at different time interval 45, the throughput becomes 6.022647. The first throughput is six times larger than the second. (2) Different users may observe quite different QoS performance when invoking the same Web service at the same time. For example, in respond time data set 3 (see footnote 1), when user 56 or 57 invokes service 353 at time interval 37, the respond time is 20.0. Both user 56 and 57 belong to AS[2] AS11318. But when user 53 who belongs to AS AS1742 invokes service 353 at time interval 37, the respond time is 0.0010. Therefore, predicting the QoS values is a critical issue for a prior identification of SLA violations.

Leitner's [4] approach predicts SLA violations by comparing the predictions of SLOs values with existing customer SLAs. The SLO may be composed of one or more QoS measurements that are combined to produce the SLO achievement value. Therefore, the approach for predicting the QoS value with high accuracy and high scalability is really important for identification of SLA violations, e.g., Leitner's approach [4].

Based on the above analysis, predicting the QoS value is becoming more and more essential for service-oriented applications designers to make service selection, service composition, and identification of SLA violations.

CF [6–9] is an effective method for Web service selection and recommendation. For example, collaborative filtering has been widely used in commercial recommendation systems [10]. CF would automatically predict the QoS values of a target Web service for an active service user by employing historical QoS information from other similar service users, who have similar historical QoS values on the service set, in which every service is similar to the target service. In a typical CF scenario, a recommender system consisting of N users and M services, the relationship between users and services is denoted by an $N \times M$ matrix, called the user-service matrix. Every entry $q_{n,m}$ in this matrix represents a vector of QoS values (e.g., response time and failure rate), which is observed by the user n on the service m. If user n has not invoked the service m before, then $q_{n,m} = 0$.

Researchers have devised a number of collaborative filtering algorithms, which could be divided into two main categories: memory-based and model-based algorithms. Memory-based CF algorithms use the entire or a sample of the user-service database to generate a prediction. Every user is part of a group of people with similar interests. By identifying the so-called neighbors of a new user (or active user), a prediction of preferences on new services for him or her can be produced. Model-based CF algorithms design and develop models (such as machine learning and data mining algorithms) to recognize complex patterns based on the training data and then make intelligent predictions for test data or real-world data, based on the learned models.

Memory-based CF algorithms are easy to implement and highly effective but suffer from a fundamental problem: inability to scale-up. Algorithms based on memory-based approaches often cannot cope well with the large numbers of users and services. Model-based CF algorithms, such as clustering CF algorithms, address the scalability problem by seeking users for recommendation within smaller and highly similar clusters, rather than within the entire database. However, they are often time-consuming to build and update.

This paper proposes a time-aware and location-aware collaborative filtering algorithm, which combines the strengths of memory-based approaches and model-based approaches and overcomes shortcomings of both approaches. QoS performance of Web services is highly related to locations, since the user-observed QoS performance of Web services is greatly influenced by the network distance and the Internet connections between users and Web services. Therefore, our approach divides the users set and services set into many clusters according to the location information. Then, it tries to seek similar users and services within smaller and highly similar clusters, rather than within the entire database. This enables our approach to be scalable. Since our approach classifies users and services according the location information, adding and deleting users or services will only update these clusters, which contain those users and services, while other clusters will not be affected. This enables our approach to build and update clusters quickly.

QoS performance of Web services is highly related to invocation time, since the service status (e.g., workload and number of clients) and the network environment (e.g., congestion, etc.) change over time. Therefore, time factor is employed by our approach to improve the prediction accuracy. QoS performance of Web services is also highly related to locations, since the user-observed QoS performance of Web services is greatly influenced by the network distance and the Internet connections between users and Web services. Therefore, location factor is also employed by our approach to improve the prediction accuracy.

The rest of this paper is organized as follows: Sect. 2 introduces related work. Section 3 presents a motivating scenario. Section 4 describes the time-aware and location-aware collaborative filtering algorithm. Section 5 presents experiments and results. Section 6 concludes this paper.

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

[2] An autonomous system (AS) is either a single network or a group of networks that is controlled by a common network administrator on behalf of a single administrative entity (such as a university and a business enterprise). An autonomous system has a globally unique number.

2 Related work

With the exponential growth of Web applications, there are lots of investigations focusing on different kinds of issues. For example, Web service selection [1,20], Web service composition [21,22], failure prediction [23], and reliability prediction [24]. In this section, we briefly present some of the research literatures related to collaborative filtering, recommender systems, and QoS prediction.

Collaborative filtering methods are widely adopted in commercial recommender systems [12,25]. Researchers have devised a number of collaborative filtering algorithms, which could be divided into two main categories: memory-based and model-based algorithms [11]. The most often analyzed examples of memory-based collaborative filtering include user-based approaches [26] and item-based approaches [27,28]. Memory-based CF methods are easy to implement and highly effective, but they suffer from a potential challenge, which is shown as follow:

Scalability Memory-based CF methods require computation that grows with both the number of users and the number of services. When the number of users or services is too large, calculating similarity values for each pair of users or services is time-consuming.

Much research effort focuses on improving the prediction accuracy of memory-based CF algorithms, but seldom focuses on scalability. For example, WSRec is a hybrid method. WSRec is a user-based and item-based collaborative filtering algorithm to make QoS prediction [13]. But WSRec suffers from serious scalability problem. In order to address the scalability problem, our approach divides the users set and services set into many clusters according to the location information and tries to seek similar users and services within smaller and highly similar clusters.

Chen et al. [16] and LACF [17] also take location factor into account. Chen is the first to recognize the influence of user location in Web services QoS prediction, and he also proposes a novel method. However, services' location information is not employed. Tang et al. [17] proposes a method of location-aware collaborative filtering to recommend Web services to users by incorporating locations of both users and services. LACF is a clustering CF algorithm. Clustering CF algorithms are trade-offs between scalability and prediction performance [29]. In order to further improve the prediction accuracy, our approach employs time factor, since QoS performance of Web services is highly related to invocation time.

In order to improve the prediction accuracy, different kinds of factors are taken into account when the missing QoS values are predicted by collaborative filtering algorithm. Figure 1 shows those factors that have been employed by CF. Jiang et al. [14] take into account the influence of personalization of Web service items when computing degree of

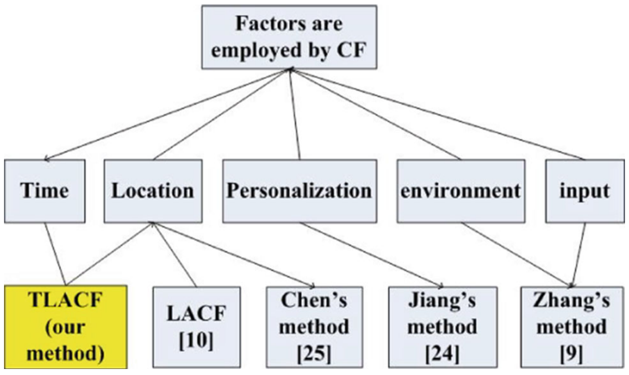

Fig. 1 The factors have been employed by CF

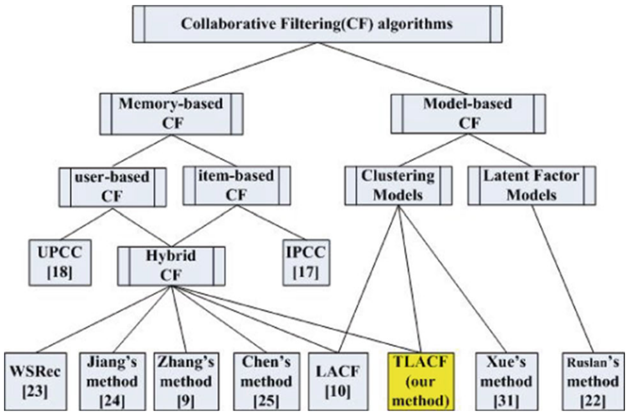

Fig. 2 The relationship between our approach and others

similarity between users. The method based on the observation that more popular services or services with more stable QoS from user to user should contribute less to user similarity measurement. Zhang et al. [15] take the environment factor and user input factor into account, where environment factor refers to those environmental features, which have effect on the QoS of Web service, e.g., bandwidth, and input factor refers to those input features, which have effect on QoS of Web service, e.g., input data size. All above works focus on improving the prediction accuracy of memory-based CF algorithms, but none of them focus on scalability.

Model-based approaches employ machine learning techniques to fit a predefined model based on the training data sets. Model-based approaches include several types: the clustering models [18] and the latent factor models [19]. Cluster models have better online scalability than memory-based collaborative filtering methods because they compare the user to a controlled number of segments rather than an entire customer base. But general clustering computation is time-consuming to build and update. Other model-based approaches also suffer from the same problems.

A relationship Fig. 2 and a comparison Table 1 have been given in this section. The relationship Fig. 2 shows the placement of our approach among others, and the comparison

Table 1 Comparison of our approach with other approaches

	Missing data	Similarity weight	Scalability	Easy to build and update	Time-aware	Location-aware
UPCC [11]	No	No	No	Yes	No	No
IPCC [12]	No	No	No	Yes	No	No
WSRec [13]	No	Yes	No	Yes	No	No
Jiang et al. [14]	No	No	No	Yes	No	No
Zhang et al. [15]	Yes	No	No	Yes	No	No
Chen et al. [16]	No	No	Yes	Yes	No	Yes
LACF [17]	No	No	Yes	Yes	No	Yes
TLACF	Yes	Yes	Yes	Yes	Yes	Yes
Xue et al. [18]	Yes	No	Yes	No	No	No
Salakhutdinov and Mnih [19]	No	No	Yes	No	No	No

Table 1 shows the differences between our approach and other approaches. Missing data will be described in Sect. 4.6. Similarity weight has been employed to reduce the influence of a small number of common users or services. In other words, similarity weight has been employed to emphasize high similarity and to punish low similarity. Similarity weight will be described in Sects. 4.4 and 4.5.

Our approach divides the users set and services set into many clusters and tries to seek similar users and services within smaller and highly similar clusters. Thus, our approach is scalable. Since our approach classifies users and services just according to location information, the time complexity of our clustering computation is $O(N + M)$ and the time complexity of updating clusters is $O(1)$, where N is the number of users and M is the number of services. The time complexity of general model-based approaches to build models, such as singular value decomposition (SVD), is $O(N^3 + M^3)$ [30]. The time complexity of SVD updating is $O(k^2 \times M + k^2 \times N)$, where k is the number of factors [31]. Thus, the clusters can be built and updated quickly by our approach. General model-based CF algorithms design and develop models by machine learning algorithms, data mining algorithms, etc., which are difficult to implement, while our approach classifies users and services just according to location information; thus, the implementation of our approach is easier. In other words, our approach would overcome the disadvantage of general model-based approaches. Since Clustering CF algorithms are trade-offs between scalability and prediction performance [29], in order to improve the prediction accuracy, our approach employs time factor, since QoS performance of Web services is highly related to invocation time.

Therefore, our approach combines the strengths of memory-based approaches and model-based approaches and overcomes shortcomings of both approaches. Our approach is efficient in predicting the missing QoS values as analyzed in Sect. 4.8. Moreover, our approach significantly improves prediction accuracy.

3 A motivating scenario

CF is an effective method for Web service selection and recommendation. Researchers have devised a number of collaborative filtering algorithms, which could be divided into two main categories: memory-based and model-based algorithms [11].

The memory-based CF methods are notably deployed into commercial systems, since they are easy to implement and highly effective [16]. But when the number of existing users and services grow tremendously, traditional memory-based CF algorithms will suffer from serious scalability problems, with computational resources going beyond practical or acceptable levels. It is $O(NM)$ in the worst case, where N is the number of users and M is the number of services, since it examines N users and up to M services for each customer [10].

Model-based CF methods, such as clustering CF algorithms, have better online scalability than memory-based CF methods, but they are often time-consuming to build and update [18]. Cluster models have better online scalability because they compare the user to a controlled number of segments rather than an entire user base.

Based on the above discussion, it is certainly helpful to have a novel method with better online scalability, which is easy to implement, easy to build, and easy to update. In this paper, location information has been employed to enable our approach to be scalable, easy to build, and easy to update. And time information has been employed to further improve the prediction accuracy.

Nevertheless, there are some problems that need to be solved to make accurate and efficient QoS prediction using location and time information: (1) How to represent user or service location and acquire location information? (2) How to employ and represent time information? (3) How to combine location and time information with collaborative filtering to predict QoS values? and (4) How do we design experiments for performance and efficiency evaluation?

4 The QoS prediction algorithm

4.1 Notations and definitions

The following are important notations used in the rest of this paper:

$U = \{u_1, u_2, \ldots, u_n\}$ is a set of service users, where n refers to the total number of service users registered in the recommendation system.

$S = \{s_1, s_2, \ldots, s_m\}$ is a set of Web services, where m refers to the total number of Web services collected by the recommendation system.

$A = \{a_1, a_2, \ldots, a_e\}$ is a set of ASNs,[3] where e refers to the total number of ASs detected from all users and services. Let a_{u_i} denotes ASN to which the user u_i's IP belongs. Similarly, a_{s_j} denotes ASN to which the service s_j's IP belongs.

$A_{u_i} = \{u|u \in U, u_i \in U, a_{u_i} \in A, a_u \in A, a_{u_i} = a_u\}$. A_{u_i} is a subset of the set U. A_{u_i} consists of those users who's IP and the user u_i's IP belong to the same ASN. If user $u \in A_{u_i}$ and $u_i \in A_{u_i}$, then user u's IP and user u_i's IP belong to the same ASN.

$A_{s_j} = \{s|s \in S, s_j \in S, a_{s_j} \in A, a_s \in A, a_{s_j} = a_s\}$. A_{s_j} is a subset of the set S. A_{s_j} consists of those services who's IP and the service s_j's IP belong to the same ASN. If service $s \in A_{s_j}$ and $s_j \in A_{s_j}$, then service s's IP and service s_j's IP belong to the same ASN.

$C = \{c_1, c_2, \ldots, c_f\}$ is a set of country IDs, where f refers to the total number of countries detected from all users and services. Let c_{u_i} denotes country ID to which the user u_i's IP belongs. Similarly, c_{s_j} denotes country ID to which the service s_j's IP belongs.

$C_{u_i} = \{u|u \in U, u_i \in U, c_{u_i} \in C, c_u \in C, c_{u_i} = c_u\}$. C_{u_i} is a subset of the set U. C_{u_i} consists of those users who's IP and the user u_i's IP belong to the same country. If user $u \in C_{u_i}$ and $u_i \in C_{u_i}$, then user u's IP and user u_i's IP belong to the same country.

$C_{s_j} = \{s|s \in S, s_j \in S, c_{s_j} \in C, c_s \in C, c_{s_j} = c_s\}$. C_{s_j} is a subset of the set S. C_{s_j} consists of those services who's IP and the service s_j's IP belong to the same country. If service $s \in C_{s_j}$ and $s_j \in C_{s_j}$, then service s's IP and service s_j's IP belong to the same country.

$T = \{t_1, t_2, \ldots, t_r\}$ is a set of time interval, where r refers to the total number of time interval. For example, 1 day has 24 h with a time interval lasting for 15 min, then $r = 96$.

[3] ASN denotes globally unique number of the AS to which the IP belongs. Note that even if users are within a same AS, it does not definitely mean that they are close geographically, and vice versa. Generally speaking, intra-AS traffic is much better than inter-AS traffic regarding transmission performance such as response time. Therefore, even if two users are located in the same city, they may seem far away from each other in terms of network distance if their computers are within different ASs.

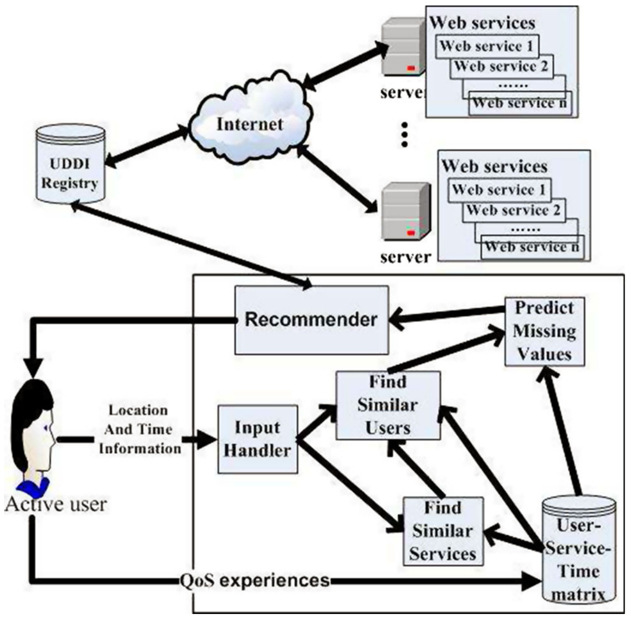

Fig. 3 A framework for Web service recommender system

$M = \{q_{u_i,s_j,t_k} | 1 \le i \le n, 1 \le j \le m, 1 \le k \le r\}$ is the user-service-time matrix, where q_{u_i,s_j,t_k} is a vector of QoS attribute values acquired from u_i invoking s_j at t_k. If u_i has no experiences on s_j at t_k, $q_{u_i,s_j,t_k} = \emptyset$.

The approach proposed by this paper address the scalability problem by seeking users and services for recommendation within smaller and highly similar clusters, rather than within the entire database.

The approach proposed by this paper divides the users set U and the services set S into many clusters according to the location information. Then, it attempts to seek topK users with the highest degree of similarities to active user u_i within smaller and highly similar clusters A_{u_i} or C_{u_i}, rather than within the entire users set U. And it also attempts to seek topK services with the highest degree of similarities to target service s_j within smaller and highly similar clusters A_{s_j} or C_{s_j}, rather than within the entire services set S.

Both A_{u_i} and C_{u_i} are user clusters. C_{u_i} classifies users at the level of countries, which is a very coarse-grained level. A_{u_i} classifies users at the level of ASN, which is a more fine-grained level. Both A_{s_j} and C_{s_j} are service clusters. C_{s_j} classifies services at the level of countries. And A_{s_j} classifies services at the level of ASN.

4.2 Collaborative framework for Web service recommender system

In this section, we present the collaborative framework for QoS prediction of Web services. Figure 3 shows the system architecture, which mainly includes the following procedures: (1) An active user provides the location and time information for the system. (2) The Input Handler processes the input data. (3) The Find Similar Services find similar services from the user-service-time matrix. (4) The Find Similar Users find similar users from the user-service-time matrix. (5) The Predict Missing Values predicts the missing QoS values for the active user using our approach. (6) The Recommender employs the predicted QoS values to recommend optimal Web services to the active user.

4.3 Overview of algorithm

In this section, an abstract description of time-aware and location-aware QoS predicting algorithm and a simple execution process of our algorithm will be given. Let u_{i_1} be an active user and s_{j_1} be a target service and t_{k_1} be the time interval. When the active user invokes the target service at t_{k_1}, the missing QoS values need to be predicted for u_{i_1}. The time-aware and location-aware QoS predicting algorithm comprises the following three sub-procedures.

1. Calculate the average similarity between target service s_{j_1} and service s_{j_2} at time interval t_{k_1}, where $s_{j_2} \in A_{s_{j_1}} - \{s_{j_1}\}$ or $s_{j_2} \in C_{s_{j_1}} - \{s_{j_1}\}$. Select the topK services with highest average similarity to the target service which will be used to calculate the user average similarity.
2. Calculate the user average similarity based on the data of services selected in the first step.
3. Predict QoS for active user based on the result of the first and second steps.

The simple execution process of our algorithm is shown in Fig. 4. And what is worth mentioning, the set of properties (i.e., availability, throughput, reliability, execution and duration) that can be evaluated by traditional collaborative filtering algorithm can also be evaluated by our algorithm.

4.4 Determine the services used for prediction

One critical step in the collaborative filtering algorithm is to compute the similarity between services and then to select the most similar services. Computation of the average similarity between service s_{j_1} and service s_{j_2} could be divided into three steps. The first step is to isolate the users who have invoked both of them. The second step is to calculate the similarity between service s_{j_1} and service s_{j_2} at time interval t_k by formula (3), where $t_k \in t_D$. The third step is to calculate the average similarity between service s_{j_1} and service s_{j_2} at time interval t_{k_1} by formula (1).

Calculating the average similarity between service s_{j_1} and s_{j_2} at time interval t_{k_1} is as formula (1):

$$\mathrm{AVGsim}_{t_{k_1},d}(s_{j_1}, s_{j_2}) = \frac{\sum\limits_{t_k \in t_D} \mathrm{sim}(s_{j_1,t_k}, s_{j_2,t_k})}{|t_D|} \quad (1)$$

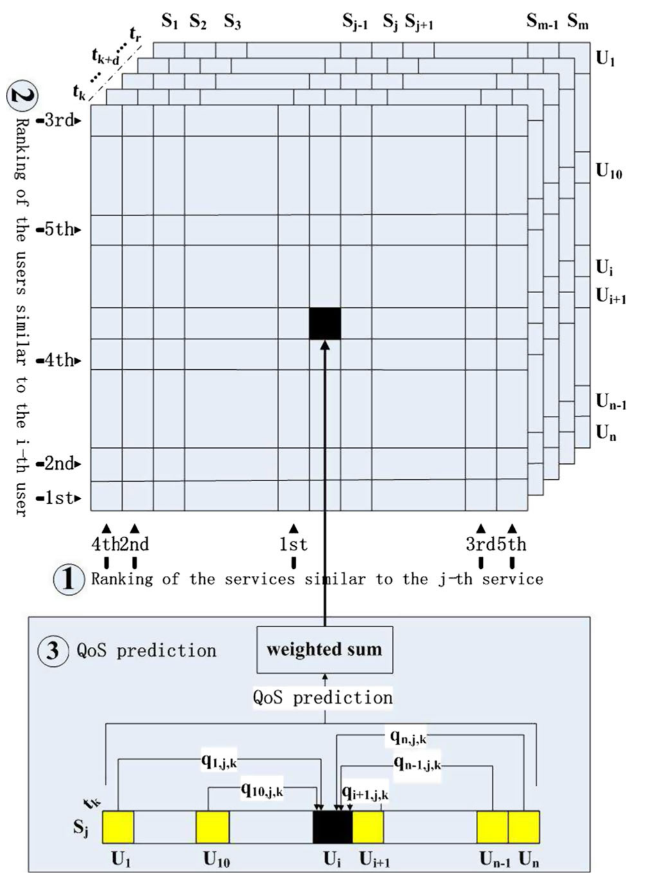

Fig. 4 The prediction generation process is illustrated for five neighbors

where $\mathrm{AVGsim}_{t_{k_1},d}(s_{j_1}, s_{j_2})$ denotes the average similarity between service s_{j_1} and s_{j_2} at time interval t_{k_1}. $\mathrm{AVGsim}_{t_{k_1},d}(s_{j_1}, s_{j_2})$ ranges from $[-1, 1]$ with a larger value indicating that services s_{j_1} and s_{j_2} are more similar. When $\mathrm{AVGsim}_{t_{k_1},d}(s_{j_1}, s_{j_2})$ is equal to 1, common users have almost same QoS on service s_{j_1} and s_{j_2} at time interval set t_D. The negative value indicates that common users always have contrary QoS experiences. Time interval set t_D is defined as follow:

$$t_D = \begin{cases} \{t_k | k_1 - d \le k \le k_1 + d, 1 \le k_1 - d, k_1 + d \le r\} \\ \{t_k | k_1 - d \le k \le r, 1 \le k_1 - d, k_1 + d \succ r\} \\ \{t_k | 1 \le k \le k_1 + d, 1 \succ k_1 - d, k_1 + d \le r\} \\ \{t_k | 1 \le k \le r, 1 \succ k_1 - d, k_1 + d \succ r\} \end{cases} \quad (2)$$

where r refers to the total number of time interval. A tunable parameter d is used to reduce the search depth. For example, when $d = 1$ and $t_{k_1} = t_2$, we can get $t_D = \{t_1, t_2, t_3\}$ and $\mathrm{AVGsim}_{t_2,d}(s_{j_1}, s_{j_2}) = \frac{\mathrm{sim}(s_{j_1,t_1}, s_{j_2,t_1}) + \mathrm{sim}(s_{j_1,t_2} + s_{j_2,t_2}) + \mathrm{sim}(s_{j_1,t_3}, s_{j_2,t_3})}{3}$. When $d = 2$ and $t_{k_1} = t_2$, we can get $t_D = \{t_1, t_2, t_3, t_4\}$ and $\mathrm{AVGsim}_{t_2,d}(s_{j_1}, s_{j_2}) = \frac{\mathrm{sim}(s_{j_1,t_1}, s_{j_2,t_1}) + \mathrm{sim}(s_{j_1,t_2}, s_{j_2,t_2}) + \mathrm{sim}(s_{j_1,t_3}, s_{j_2,t_3}) + \mathrm{sim}(s_{j_1,t_4}, s_{j_2,t_4})}{4}$. Therefore d is used to limit the search depth.

$\text{sim}(s_{j1,t_k}, s_{j2,t_k})$ represents the similarity between service s_{j1} and s_{j2} at time interval t_k. There are a number of different methods to compute the similarity between services. For example, cosine-based similarity [28], Pearson correlation coefficient (PCC) [28], and adjusted-cosine similarity [28]. PCC often overestimates the similarities of services, which are actually not similar but happen to have similar QoS on a few common users [32]. To address this problem, this paper employs a similarity weight to reduce the influence of the small number of similar common users. An enhanced PCC for $\text{sim}(s_{j1,t_k}, s_{j2,t_k})$ is defined as:

$$\text{sim}(s_{j1,t_k}, s_{j2,t_k}) = \frac{\sum_{u\in U_{s_{j1},s_{j2},t_k}} (q_{u,s_{j1},t_k} - \overline{q_{s_{j1},t_k}})(q_{u,s_{j2},t_k} - \overline{q_{s_{j2},t_k}})}{\sqrt{\sum_{u\in U_{s_{j1},s_{j2},t_k}} (q_{u,s_{j1},t_k} - \overline{q_{s_{j1},t_k}})^2} \times \sqrt{\sum_{u\in U_{s_{j1},s_{j2},t_k}} (q_{u,s_{j2},t_k} - \overline{q_{s_{j2},t_k}})^2}} \times \frac{\left|U_{s_{j1},s_{j2},t_k}\right|}{\left|U_{\text{flag}}\right|} \quad (3)$$

From this definition, $\text{sim}(s_{j1,t_k}, s_{j2,t_k})$ ranges from $[-1, 1]$ with a larger value, indicating that services s_{j1} and s_{j2} are more similar at time interval t_k. When $\text{sim}(s_{j1,t_k}, s_{j2,t_k})$ is equal to 1, common users have almost same QoS on service s_{j1} and s_{j2} at time interval t_k. The negative value indicates that common users always have contrary QoS experiences at time interval t_k.
Where $U_{s_{j1},s_{j2},t_k} \subseteq U_{\text{flag}}$. U_{s_{j1},s_{j2},t_k} is the set of common users of service s_{j1} and s_{j2} at time interval t_k. $\frac{\left|U_{s_{j1},s_{j2},t_k}\right|}{\left|U_{\text{flag}}\right|}$ is a similarity weight. When $\left|U_{s_{j1},s_{j2},t_k}\right|$ is small, the similarity weight $\frac{\left|U_{s_{j1},s_{j2},t_k}\right|}{\left|U_{\text{flag}}\right|}$ will devalue the similarity estimation between the services. q_{u,s_{j1},t_k} is a vector of QoS attribute values acquired from user u invoking s_{j1} at time interval t_k. $\overline{q_{s_{j1},t_k}}$ represents the vector of average QoS values of service s_{j1} at time interval t_k. q_{u,s_{j2},t_k} is a vector of QoS attribute values acquired from u invoking s_{j2} at time interval t_k.$\overline{q_{s_{j2},t_k}}$ represents the vector of average QoS values of service s_{j2} at time interval t_k.

U_{flag} is a user cluster to which the active user u_{i_1} belongs. U_{flag} is used to decrease the time complexity of calculating the similarity between two services. U_{flag} is defined as:

$$U_{\text{flag}} = \begin{cases} A_{u_{i_1}}, & \text{if flag} = 0 \text{ and } |A_{u_{i_1}}| \geq \text{TH} \\ C_{u_{i_1}}, & \text{if (flag} = 1 \text{ or (flag} = 0 \\ & \text{and } |A_{u_{i_1}}| < \text{TH)) and } |C_{u_{i_1}}| \geq \text{TH} \\ U, & \text{if flag} = 2 \text{ or ((flag} = 1 \text{or (flag} = 0 \\ & \text{and } |A_{u_{i_1}}| < \text{TH)) and } |C_{u_{i_1}}| < \text{TH)} \end{cases} \quad (4)$$

where there is a parameter flag which is used to set the scope of U_{flag}. For example, when flag $= 0$ and $|A_{u_{i_1}}| \geq$ TH, the common users set $U_{\text{flag}} = A_{u_{i_1}}$. In this case, the time complexity of calculating the similarity between two services is $O(|A_{u_{i_1}}|)$, rather than $O(|U|)$, according to Eq. (3). TH is a threshold. Since it will not be reliable when there are too few common users to generate similarity values between two services [29], a variable TH is employed.

We can get the average similarity of the two services by the Eqs. (1), (2) and (3).

4.5 User similarity calculation

Similar services selection is a very important step for making accurate prediction, since dissimilar services will lead to inaccurate missing value prediction. Traditional Top-K algorithm sorts the services according to the similarity and simply returns the topK most similar services as topK services. In practice, some services are not similar to the target services. Traditional Top-K algorithm ignores this problem and still chooses the topK most similar services to predict the missing value. This will greatly influence prediction accuracy. Then, similar to [13], a similar service will be removed from the set of the topK similar services if its similarity is equal to or smaller than 0.

$S_{s_{j1},\text{topK}}$ would be gotten by traditional Top-K algorithm. And the scope of $S_{s_{j1},\text{topK}}$ is defined as follows:

$$S_{s_{j1},\text{topK}} \subseteq \begin{cases} A_{s_{j1}}, & \text{if flag } = 0 \text{ and } \left|A_{s_{j1}}\right| \geq \text{topK} \\ C_{s_{j1}}, & \text{if (flag } = 1 \text{ or} \\ & (\text{flag } = 0 \text{ and } \left|A_{s_{j1}}\right| < \text{topK})) \\ & \text{and } \left|C_{s_{j1}}\right| \geq \text{topK} \\ S, & \text{if flag } = 2 \text{ or} \\ & (\text{flag } = 1 \text{ and } \left|C_{s_{j1}}\right| < \text{topK}) \\ & \text{or(flag } = 0 \text{ and } \left|A_{s_{j1}}\right| < \text{topK} \\ & \text{and } \left|C_{s_{j1}}\right| < \text{topK}) \end{cases} \quad (5)$$

where the cardinality of the set $S_{s_{j1},\text{topK}}$ is equal to topK. There is a parameter flag, which is used to set the scope of $S_{s_{j1},\text{topK}}$. For example, when flag $= 0$ and $\left|A_{s_{j1}}\right| \geq$ topK, our approach attempts to seek similar services within smaller and highly similar clusters $A_{s_{j1}}$, rather than within the entire services set S. Therefore, $S_{s_{j1},\text{topK}} \subseteq A_{s_{j1}}$. When flag $= 1$ and $\left|C_{s_{j1}}\right| \geq$ topK, our approach attempts to seek similar services within similar clusters $C_{s_{j1}}$, rather than within the entire services set S. When flag $= 2$, it will seek within the entire services set S directly. Thus, flag can limit similar services search to a much smaller scope.

Similar to [13], those similar services will be removed from $S_{s_{j1},\text{topK}}$ if their similarity is equal to or smaller than 0. Then, a new set of similar services $S'_{s_{j1},\text{topK}'}$ can be gotten, where $S'_{S_{j1},\text{topK}'} \subseteq S_{s_{j1},\text{topK}}$, and topK$'$ are the cardinality of the set $S'_{S_{j1},\text{topK}'}$, and topK$' \leq$ topK.

This set collecting topK′ services with most trusting referential value composes a new matrix M_{topK',t_D}. The matrix M_{topK',t_D} is defined as follow:

$$M_{\mathrm{topK}',t_D} = \{q_{u_i,s_j,t_k} | u_i \in U, s_j \in S'_{s_{j_1},\mathrm{topK}'}, t_k \in t_D\}$$

Time interval set t_D is defined in formula (2).

Then, the average similarity between users can be calculated based on the new matrix M_{topK',t_D}. Calculating the average similarity between users is as formula (6):

$$\mathrm{AVGsim}_{t_{k_1},d}(u_{i_1}, u_{i_2}) = \frac{\sum_{t_k \in t_D} \mathrm{sim}(u_{i_1,t_k}, u_{i_2,t_k})}{|t_D|} \tag{6}$$

where $\mathrm{AVGsim}_{t_{k_1},d}(u_{i_1}, u_{i_2})$ denotes average similarity between user u_{i_1} and u_{i_2} at time interval t_{k_1}. $\mathrm{AVGsim}_{t_{k_1},d}(u_{i_1}, u_{i_2})$ ranges from [−1, 1] with a larger value indicating that users u_{i_1} and u_{i_2} are more similar. When $\mathrm{AVGsim}_{t_{k_1},d}(u_{i_1}, u_{i_2})$ is equal to 1, both users have almost same QoS on common services at time interval set t_D. The negative value indicates that both users always have contrary QoS experiences on common services at time interval set t_D.

The definition of time interval set t_D is given in Eq. (2).

$\mathrm{sim}(u_{i_1,t_k}, u_{i_2,t_k})$ represents the similarity between service u_{i_1} and u_{i_2} at time interval t_k. Just like the $\mathrm{sim}(s_{j_1,t_k}, s_{j_2,t_k})$, an enhanced PCC for $\mathrm{sim}(u_{i_1,t_k}, u_{i_2,t_k})$ is defined as:

$$\mathrm{sim}(u_{i_1,t_k}, u_{i_2,t_k}) = \frac{\sum_{s \in S_{u_{i_1},u_{i_2},t_k}} (q_{u_{i_1},s,t_k} - \overline{q_{u_{i_1},t_k}})(q_{u_{i_2},s,t_k} - \overline{q_{u_{i_2},t_k}})}{\sqrt{\sum_{s \in S_{u_{i_1},u_{i_2},t_k}} (q_{u_{i_1},s,t_k} - \overline{q_{u_{i_1},t_k}})^2} \times \sqrt{\sum_{s \in S_{u_{i_1},u_{i_2},t_k}} (q_{u_{i_2},s,t_k} - \overline{q_{u_{i_2},t_k}})^2}} \times \frac{\left|S_{u_{i_1},u_{i_2},t_k}\right|}{\left|S'_{s_{j_1},\mathrm{topK}'}\right|} \tag{7}$$

From this definition, $\mathrm{sim}(u_{i_1,t_k}, u_{i_2,t_k})$ ranges from [−1, 1] with a larger value, indicating that users u_{i_1} and u_{i_2} are more similar at time interval t_k. When $\mathrm{sim}(u_{i_1,t_k}, u_{i_2,t_k})$ is equal to 1, both users have almost same QoS on common services at time interval t_k. The negative value indicates that both users always have contrary QoS experiences on common services at time interval t_k.

where $S_{u_{i_1},u_{i_2},t_k} \subseteq S'_{s_{j_1},\mathrm{topK}'}$. $S_{u_{i_1},u_{i_2},t_k}$ is the set of common services which is invoked by user u_{i_1} and u_{i_2} at time interval t_k. $\frac{\left|S_{u_{i_1},u_{i_2},t_k}\right|}{\left|S'_{s_{j_1},\mathrm{topK}'}\right|}$ is a similarity weight. When $\left|S_{u_{i_1},u_{i_2},t_k}\right|$ is small, the similarity weight $\frac{\left|S_{u_{i_1},u_{i_2},t_k}\right|}{\left|S'_{s_{j_1},\mathrm{topK}'}\right|}$ will devalue the similarity estimation between the users. $q_{u_{i_1},s,t_k}$ is a vector of QoS attribute values acquired from user u_{i_1} invoking service s at time interval t_k. $\overline{q_{u_{i_1},t_k}}$ represents the vector of average QoS values of user u_{i_1} on service set $S'_{s_{j_1},\mathrm{topK}'}$ at time interval t_k. $q_{u_{i_2},s,t_k}$ is a vector of QoS attribute values acquired from user u_{i_2} invoking service s at time interval t_k. $\overline{q_{u_{i_2},t_k}}$ represents the vector of average QoS values of user u_{i_2} on service set $S'_{s_{j_1},\mathrm{topK}'}$ at time interval t_k. We can get the average similarity between the two users by the Eqs. (2), (6), and (7) and new matrix M_{topK',t_D}.

The final prediction is based on performance of the left topK users, $U_{u_{i_1},\mathrm{topK}} = \{u_{y_1}, u_{y_2}, \ldots, u_{y_{\mathrm{topK}}}\}$. The parameter flag is also used to set the scope of $U_{u_{i_1},\mathrm{topK}}$. The scope of $U_{u_{i_1},\mathrm{topK}}$ is defined as follow:

$$U_{u_{i_1},\mathrm{topK}} \subseteq \begin{cases} A_{u_{i_1}}, & \text{if flag} = 0 \text{ and } \left|A_{u_{i_1}}\right| \geq \mathrm{topK} \\ C_{u_{i_1}}, & \text{if (flag} = 1 \text{ or} \\ & \text{(flag} = 0 \text{ and } \left|A_{u_{i_1}}\right| < \mathrm{topK})) \\ & \text{and } \left|C_{u_{i_1}}\right| \geq \mathrm{topK} \\ S, & \text{if flag} = 2 \text{ or} \\ & \text{(flag} = 1 \text{ and } \left|C_{u_{i_1}}\right| < \mathrm{topK}) \\ & \text{or (flag} = 0 \text{ and } \left|A_{u_{i_1}}\right| < \mathrm{topK} \\ & \text{and } \left|C_{u_{i_1}}\right| < \mathrm{topK}) \end{cases} \tag{8}$$

4.6 Missing data prediction

Some user $\in U_{u_{i_1},\mathrm{topK}}$ have not invoked the target service at target time interval. These missing data will have great impact on prediction. Simply ignoring these missing data would reduce the prediction accuracy. The general method calculates the missing data by average value of neighbors. Neighbors are those users whose ordinal number of the rows are near the user's ordinal number of the row. Thus, it is probable that the neighbors are not similar users. Therefore, the average value is not accurate, for it ignores the difference among users. This paper employs a novel method to calculate the missing data. It takes time and location factor into account to calculate the missing data. For examples, to calculate the missing data q_{u_i,s_j,t_k}:

$$q_{u_i,s_j,t_k} = \begin{cases} \frac{\sum_{t \in (t_D - t'_D)} q_{u_i,s_j,t}}{|t_D - t'_D|}, & \text{if } t_D - t'_D \neq \emptyset \\ \frac{\sum_{u \in (U_{u_i,\mathrm{topK}} - U_{u_i,\mathrm{topK}}'')} q_{u,s_j,t_k}}{|U_{u_i,\mathrm{topK}} - U_{u_i,\mathrm{topK}}''|}, & \text{if } t_D - t'_D = \emptyset \\ & \text{and } U_{u_i,\mathrm{topK}} - U_{u_i,\mathrm{topK}}'' \neq \emptyset \\ \mathrm{DV} & \text{if } U_{u_i,\mathrm{topK}} - U_{u_i,\mathrm{topK}}'' = \emptyset \\ & \text{and } t_D - t'_D = \emptyset \end{cases}$$

t_D would be calculated by Eq. (2). $t'_D = \{t | t \in t_D, q_{u_i,s_j,t} = \emptyset\}$. $U_{u_i,\mathrm{topK}}$ would be calculated by Eq. (8). $U_{u_i,\mathrm{topK}}'' = \{u | u \in U_{u_i,\mathrm{topK}}, q_{u,s_j,t_k} = \emptyset\}$. DV denotes default value. Empirically, assuming some default value for the missing data can improve the CF prediction performance [29]. In this paper, DV is a variable. DV has different values in different applications. For example, Chee et al. [33] use the average value as default voting to extend each user's rating history. Breese et al. [11] use a neutral or somewhat negative preference for the unobserved ratings. Our approach tries to calculate the missing data by the same user's QoS experiences on the service at other time or similar users' QoS experiences on the service at the same time. For example, $q_{u_i,s_j,t_k'}$ or q_{u_i',s_j,t_k} would be used to calculate the missing data q_{u_i,s_j,t_k}.

4.7 Prediction making

After computing the degree of similarity between the active user and all other users, a user similarity vector has been gotten. By the traditional Top-K algorithm, we can get Top-K similar users set $U_{u_{i_1},\mathrm{topK}}$.

Similar users selection is a very important step for making accurate prediction, since dissimilar users will lead to inaccurate missing value prediction for the active user. In practice, some users are not similar to active user. Traditional Top-K algorithms ignore this problem and still choose the topK most similar users to predict the missing value. This will greatly influence prediction accuracy. Then, similar to [13], a similar user will be removed from the set of the topK similar users if its similarity is equal to or smaller than 0. Then, a new set of similar users $U'_{u_{i_1},\mathrm{topK}'''}$ can be gotten, where $U'_{u_{i_1},\mathrm{topK}'''} \subseteq U_{u_{i_1},\mathrm{topK}}$ and topK$'''$ is the cardinality of the set $U'_{u_{i_1},\mathrm{topK}'''}$ and $\mathrm{topK}''' \le \mathrm{topK}$.

Our approach computes the prediction on the target service s_{j_1} for the active user u_{i_1} at time interval t_{k_1} by computing the weighted sum of the QoS attribute values given by the similar users on the target service s_{j_1} at time interval t_{k_1}. Each QoS attribute value is weighted by the corresponding average similarity $\mathrm{AVGsim}_{t_{k_1},d}(u_{i_1},u_{i_2})$ between users u_{i_1} and u_{i_2}.

After the above step, $q_{u_{i_1},s_{j_1},t_{k_1}}$ could be predicted by similar users set $U'_{u_{i_1},\mathrm{topK}'''}$.

$$q_{u_{i_1},s_{j_1},t_{k_1}} = \frac{\sum_{u\in U'_{u_{i_1},\mathrm{topK}'''}} \mathrm{AVGsim}_{t_{k_1},d}(u_{i_1},u)\times q_{u,s_{j_1},t_{k_1}}}{\sum_{u\in U'_{u_{i_1},\mathrm{topK}'''}} \mathrm{AVGsim}_{t_{k_1},d}(u_{i_1},u)} \tag{9}$$

$q_{u_{i_1},s_{j_1},t_{k_1}}$ could be predicted by Eq. (9).

4.8 Complexity analysis

The main computation of our approach is seeking the topK services with the highest degree of similarities to target service s_{j_1} and the topK users with the highest degree of similarities to active user u_{i_1}.

Based on the formulas (1), (2), (3), and (4), the worst-cast time complexity of calculating the average similarity between two services is $O(n \times (d+1))$, where n refers to the total number of users registered in the recommendation system and d is a parameter for our approach. Thus, the worst-cast time complexity of seeking the topK services with the highest degree of average similarities to target service s_{j_1} is $O(n \times (d+1) \times m)$, where m refers to the total number of services registered in the recommendation system. Based on the formulas (5), (6) and (7), the time complexity of calculating the average similarity between two users is $O(\mathrm{topK} \times (d+1))$, where topK is the cardinality of the set $S_{s_{j_1},\mathrm{topK}}$. Thus, the worst-cast time complexity of searching the topK users with the highest degree of average similarities to active user u_{i_1} is $O(\mathrm{topK} \times (d+1) \times n)$. Since $\mathrm{topK} \le m$, the worst-cast time complexity of our approach is $O(n \times (d+1) \times m)$.

The best time complexity of calculating the average similarity between two services is $O\left(\left|A_{u_{i_1}}\right| \times (d+1)\right)$, where $\left|A_{u_{i_1}}\right|$ is cardinality of the set $A_{u_{i_1}}$ and u_{i_1} is the active use. Thus, the best time complexity of seeking the topK services with the highest degree of average similarities to target service s_{j_1} is $O\left(\left|A_{u_{i_1}}\right| \times (d+1) \times \left|A_{s_{j_1}}\right|\right)$, where $\left|A_{s_{j_1}}\right|$ is cardinality of the set $A_{s_{j_1}}$. Since the time complexity of calculating the average similarity between two users is $O(\mathrm{topK} \times (d+1))$, the best time complexity of seeking the topK users with the highest degree of average similarities to active user u_{i_1} is $O\left(\mathrm{topK} \times (d+1) \times \left|A_{u_{i_1}}\right|\right)$. Thus, the best time complexity of our approach is $O\left(x \times (d+1) \times \left|A_{u_{i_1}}\right|\right)$, where $x = \mathrm{MAX}\left(\mathrm{topK}, \left|A_{s_{j_1}}\right|\right)$.

When numbers of existing users and services grow tremendously, i.e., Amazon.com has more than 29 million customers and several million catalog items, $\mathrm{topK} \ll \left|A_{u_{i_1}}\right|$ and $\mathrm{topK} \ll \left|A_{s_{j_1}}\right|$. When flag $= 0$, we can get $S_{s_{j_1},\mathrm{topK}} \subseteq A_{s_{j_1}}$ and $U_{u_{i_1},\mathrm{topK}} \subseteq A_{u_{i_1}}$ according to formulas (5) and (8). In this case, the actual complexity of our approach is almost equal to the best time complexity. And what is worth mentioning, when $d \ll r$, i.e., $d = 2$, the accuracy of our approach is rather high according to the experiment results, where r refers to the total number of time interval. Then, the final performance of the algorithm is approximately $O\left(\left|A_{u_{i_1}}\right| \times \left|A_{s_{j_1}}\right|\right)$, where $\left|A_{u_{i_1}}\right| \ll n$ and $\left|A_{s_{j_1}}\right| \ll m$.

This complexity analysis shows that our approach is very efficient and can be applied to large-scale systems.

5 Experiments

In this paper, the QoS prediction quality has been employed to evaluate recommendation quality, since it is a key component for service recommendation system. The normalized mean absolute error (NMAE) metric has been used to measure the prediction quality. We conduct several experiments and then compare our approach with several state-of-the-art collaborative filtering prediction methods to evaluate the new method proposed by this paper. Specifically, these experiments adopt a real-world Web service data set in order to improve the credibility of the results.

All experiments were implemented and deployed with JDK6.0, Eclipse Helios Service Release 2, Apache HttpComponents[4] and the tool provided by Team Cymru.[5]

All experiments were run on a win7-based PC with Intel Core i3-3220 CPU having a speed of 3.3 GHz and 4 GB of RAM.

Our experiments could be divided into two phases. The first phase consists of three experiments, which are described, respectively, in Sects. 5.3, 5.4, and 5.5. The second phase consists of two experiments, which are described, respectively in Sects. 5.6, and 5.7. The purpose of the first phase is to analyze the impact of flag, *d*, and topK on prediction accuracy. The purpose of the second phase is to evaluate the prediction performance and efficiency through comparison between our approach and some other well-known prediction methods.

In the following, Sect. 5.1 gives the description of our experimental data set and introduces the common Hypothesis for all experiments and shows how we get the NSA from IP, Sect. 5.2 defines the evaluation metrics, Sects. 5.3, 5.4, and 5.5 study the impact of parameter flag, *d* and topK, respectively, Sects. 5.6 and 5.7 compares the prediction quality and efficiency of our approach with other competing methods, respectively, finally, the discussion of the evaluation is given in Sect. 5.8.

5.1 Data set

To evaluate our approach in the real world, this paper adopts a real-world Web service data set WSDream data set 3,[6] which was published in Ref. [5]. In order to evaluate our approach, we get the users' IP and services' URL from Zheng who is the second author of the paper [5]. And the users' IP and Web Services' URL of WSDream data set 3 can be found in our Web site.[7] We convert the Web services' URL to the equivalent IP address by java class java.net.InetAddress. And we accomplish IP to ASN mapping and IP to country mapping by Team Cymru. The tool has been published in Web site.[8] Apache HttpComponents has been used to automatically accomplish IP to ASN mapping and IP to country mapping.

The original data set 3 contains QoS records of service invocations on 4,532 Web services from 142 users at 64 time interval, which are transformed into a user-service-time matrix. Every time interval lasts for 15 min. Then, the data set is a 142 × 4,532 × 64 user-service-time matrix, and each item is a pair values (RT, TP). RT denotes response time and TP denotes throughput. The original user-service-time matrix will be decomposed into two simpler matrices: 142 × 4,532 × 64 user-service-time RT matrix and 142 × 4,532 × 64 user-service-time TP matrix.

Forty users' IP addresses have not been provided by the author of the paper [5]. And we fail to translate or map 777 services' URL to IP addresses or ASN. Thus, the new user-service-time RT matrix and TP matrix are 102 × 3,755 × 64 user-service-time RT matrix and 102 × 3,755 × 64 user-service-time TP matrix, respectively. The new RT matrix and TP matrix will be used to compute similarity and measure prediction quality in following experiments.

We find that all 102 users are distributed within 68 ASs and 20 countries and all 3,755 services are distributed within 861 ASs and 68 countries.

In the following experiment, the 102 users are divided into two groups: 82 service users randomly selected as training service users and the rest as test service users. The RT matrix is divided into the RT-training matrix and the RT-test matrix, and so is the TP matrix. For the training matrix, we randomly remove entries to make the matrix sparser with density 30 %. The value of DV is 0 for convenience. And the value of TH is 50. Each experiment is performed 100 times, and their average values are taken as results.

5.2 Evaluation metrics

Measures for evaluating the quality of a recommender system can be mainly categorized into two classes:

1. Statistical accuracy metrics evaluate the accuracy of a system by comparing the numerical prediction QoS values with the actual QoS values acquired by invoking actual service. Mean absolute error (MAE) [34] between actuality and predictions is a widely used metric. MAE is a measure of the deviation of predictions from their actual QoS values. For each actuality-prediction pair $\langle Aq_{u_i,s_j,t_k}, Pq_{u_i,s_j,t_k}\rangle$, this metric treats the absolute error between them equally. The MAE is computed by first summing these absolute errors of the N corresponding actuality-prediction pairs and then computing the average. Formally,

$$\text{MAE} = \frac{\sum_{u_i,s_j,t_k} \left|Aq_{u_i,s_j,t_k} - Pq_{u_i,s_j,t_k}\right|}{N}$$

where Aq_{u_i,s_j,t_k} denotes actual QoS values of Web service s_j observed by user u_i at time interval t_k and Pq_{u_i,s_j,t_k} denotes the predicted QoS values of Web service s_j observed by user u_i at time interval t_k, and N denotes the number of predicted values. The lower the MAE, the higher the recommendation system accuracy.

[4] http://hc.apache.org/.

[5] http://www.team-cymru.org/Services/ip-to-asn.html#dns.

[6] http://www.wsdream.net/wsdream/dataset.html.

[7] http://radlab.sjtu.edu.cn/?p=154.

[8] http://whois.cymru.com/.

Since different QoS properties of Web services have distinct value ranges, the normalized mean absolute error (NMAE) [34] metric is also used to measure the prediction accuracy. NMAE is defined as:

$$\text{NMAE} = \frac{\text{MAE}}{\left(\sum_{u_i,s_j,t_k} Aq_{u_i,s_j,t_k}\right)/N}$$

Lower NMAE value represents higher prediction accuracy about recommendation system.

2. Decision support accuracy metrics evaluate how effective a prediction system is at helping a user select high-quality Web services from the set of Web services. These metrics assume the prediction process as a binary operation either Web services are predicted (good) or not (bad). With this observation, whether a Web service has a prediction score of 3 or 7 on a ten-point scale is irrelevant if the user only chooses to consider predictions of 8 or higher. The most commonly used decision support accuracy metrics is ROC sensitivity [35].

In this paper, NMAE is employed to measure the prediction quality, since it is most commonly used and easiest to interpret directly.

5.3 Impact of parameter flag

In the definition of formulas (4), (8), and (5), there is a parameter flag which is used to set the scope of common users, similar users, and similar services, respectively. To study the impact of parameter flag, two experiments have been implemented. We set topK $= 5, 10, 15, \ldots, 45$, respectively, $d = 0$ and vary the value of flag from 0 to 2 with a step value of 1. And the results is shown in Figs. 5 and 6.

flag $= 0$ means that common users and the topK similar users would be located in the AS to which the active user belongs and the topK similar services would be located in the AS to which the target service belongs. flag $= 1$ means that common users and the topK similar users would be located in the Country to which the active user belongs and the topK similar services would be located in the Country to which the target service belongs. flag $= 2$ means that the location factor will not be taken into account when calculating the average similarity between two services and identifying the topK similar users and services.

According to Fig. 6, when topK $\geq$ 30, the prediction accuracy of our approach tends to decrease with the increase of topK, which may be due to the fact that most ASs have only a very small number of users and services in the data set; hence, users or services within them must have similar neighbors from other ASs, especially when topK becomes large. We find out that this phenomenon does exist and certainly

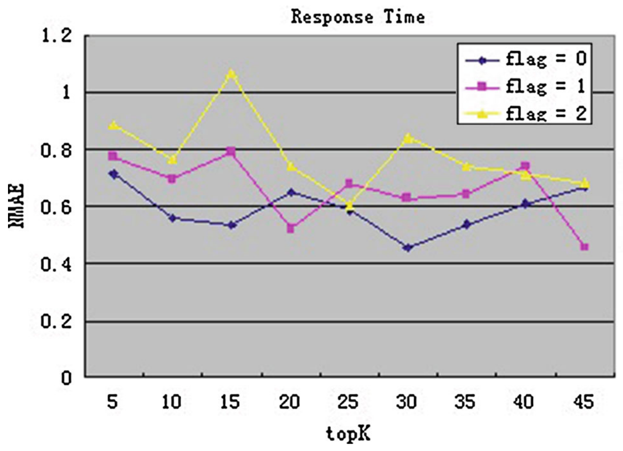

Fig. 5 Impact of the parameter flag for response time ($d = 0$)

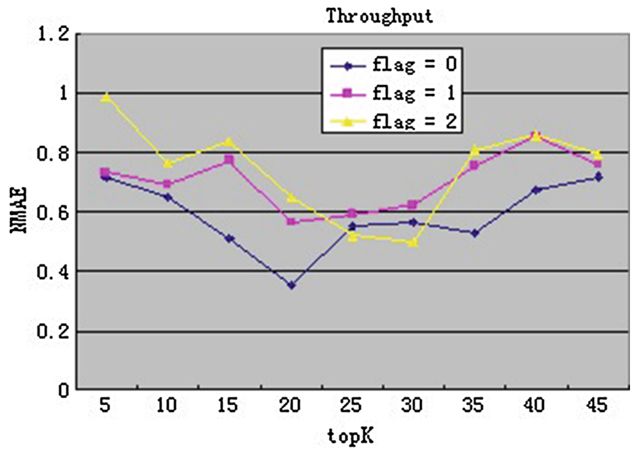

Fig. 6 Impact of the parameter flag for throughput ($d = 0$)

reduces the correlation between location closeness and QoS similarity. But according to Figs. 5 and 6, we can conclude that the prediction accuracy of our approach is improved with the decrease in the parameter flag. In other words, the algorithm which takes the location factor into account would improve prediction accuracy.

5.4 Impact of parameter d

In the definition of formula (2), there is a parameter d which is used to reduce the search depth. To study the impact of parameter d, two experiments have been implemented. We set topK $= 5$ and flag $= 0$, respectively, and vary the value of d from 0 to 40 with a step value of 2. And the results are shown in Fig. 7. According to Fig. 7, we can conclude that the prediction accuracy of our approach oscillates along with the increase of the value of parameter d. The reasons are as follows. On the one hand, since QoS performance of Web services is highly related to invocation time, there are three main factors which may decrease the prediction accuracy of

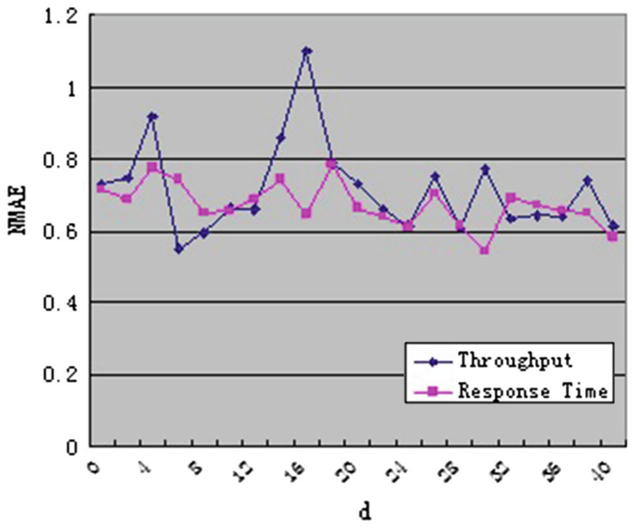

Fig. 7 Impact of the parameter d (flag = 0)

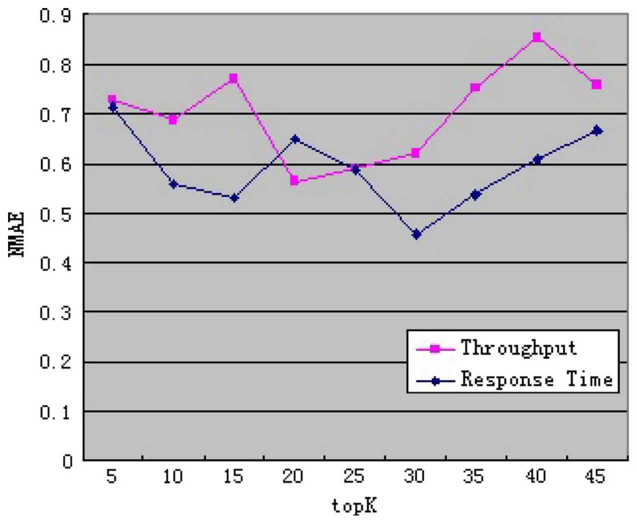

Fig. 8 Impact of the parameter topK (flag = 0 and $d = 0$)

our approach. First, the average similarity of services which are extremely similar to target service at and only at the time interval t_{k_1} would decrease with the increase of the parameter d. Second, the average similarity of services which are not similar to target service at the time interval t_{k_1} would increase with the increase of the parameter d. Third, most of highly similar services and users are in the same time interval or nearby time intervals with the active user and target service. On the other hand, the average similarity of services which are extremely similar to target service at the time interval t_{k_1} would increase with the increase of the parameter d, which would improve the prediction accuracy according to Eq. (9). What is worth mentioning, when the value of parameter d is 0, our approach has the minimal similar user or service search depth and higher prediction accuracy compared with other methods.

According to the analysis above, we can conclude that the prediction accuracy of our approach would be improved with the increase of the parameter d, and it also would be decreased with the increase of the parameter d, whether the prediction accuracy of our approach would be improved or decreased is determined by the values of the data set and the current value of the parameter d. Thus, the variation curves of prediction accuracy of our approach would be different for different data sets. This can explain why for certain values of d the NMAE value for throughput is completely inconsistent with that of the response time in Fig. 7.

5.5 Impact of parameter topK

To study the impact of parameter topK, two experiments have been implemented. We set $d = 0$ and flag = 0, respectively, and vary the value of topK from 5 to 45 with a step value of 5. And the results are shown in Fig. 8. As shown in Fig. 8, we observe that as topK increases, the NMAE decreases (prediction accuracy increases), but when topK surpasses a threshold, the NMAE increases (prediction accuracy decreases) with further increase of the value of topK. This observation indicates that too few similar users and services are not enough to characterize the features of active user and target service, while too many similar users and services will increase the number of users and services which are actually not similar and the number of users and services which are similar but located in other ASs.

5.6 Compare our method with other well-known prediction methods

In order to study the prediction performance, we compare the prediction accuracy of the following methods:

1. UPCC [11] employs the average QoS value of the users on other Web services to predict the missing value. And UPCC is a well-known prediction method.
2. IPCC [12] employs the average QoS value of the Web service item observed by other users to predict the missing value for the active users. IPCC is also a well-known prediction method.
3. WSRec [13] is a hybrid algorithm combing UPCC and IPCC. Similar to WSRec, our approach is also a hybrid algorithm combing UPCC and IPCC. But WSRec suffers from serious scalability problem. In order to overcome the disadvantage of WSRec, our approach divides the users set and services set into many clusters according to the location information and tries to seek similar users and services within smaller and highly similar clusters.
4. LACF [17] is a location-aware hybrid CF. LACF also divides the users set and services set into many clusters

Table 2 NMAE performance comparison

	Throughput topK = 5	Throughput topK = 10	Response time topK = 5	Response time topK = 10
UPCC	0.986	0.960	1.040	0.932
IPCC	1.090	1.130	0.960	0.820
WSRec	0.850	0.943	0.813	0.801
LACF	0.813	0.826	0.789	0.729
TLACF	0.717	0.649	0.715	0.558

according to the location information, and it is also scalable. LACF is a clustering CF algorithm. Clustering CF algorithms are trade-offs between scalability and prediction performance [29]. In order to further improve the prediction accuracy, our approach employs time factor, since QoS performance of Web services is highly related to invocation time.

Table 3 Comparison of predicting time of different approaches (second)

	UPCC	IPCC	WSRec	LACF	TLACF
Throughput	0.0163	0.0305	0.0478	0.0023	0.0027
Response time	0.0167	0.0341	0.0537	0.0029	0.0034

Since UPCC and IPCC have been widely accepted and adopted, they have been picked up for comparison in a number of CF-based approaches. Thus, they are also picked up for comparison in this paper. Since our approach is a clustering CF algorithm and clustering CF algorithms are trade-offs between scalability and prediction performance, our approach employs time factor to improve the prediction accuracy. In order to show the improvement of prediction accuracy, our approach should be compared with a clustering CF algorithm and a hybrid algorithm. Since WSRec is a well-known hybrid algorithm, it is picked up for comparison in this paper. Since LACF is a location-aware hybrid CF and it also divides the users set and services set into many clusters according to the location information, LACF is picked up for comparison in this paper.

The original 102 × 3,755 × 64 user-service-time RT matrix and TP matrix are converted into 102 × 3,755 user-service RT matrix and TP matrix, respectively. $q_{u_i,s_j} = q_{u_i,s_j,t_k}$ where $t_k \in \{1, 2, \ldots, 64\}$. In order to simulate the real-world situation, t_k will be randomly selected from set $\{1, 2, \ldots, 64\}$. For any two entries of 102 × 3,755 user-service RT matrix, $q_{u_{i_1},s_{j_1}} = q_{u_{i_1},s_{j_1},t_{k_1}}$ and $q_{u_{i_2},s_{j_2}} = q_{u_{i_2},s_{j_2},t_{k_2}}$, if $u_{i_1} \neq u_{i_2}$ or $s_{j_1} \neq s_{j_2}$, it is possible that $t_{k_1} \neq t_{k_2}$. And so does 102 × 3,755 user-service TP matrix.

The 102 service users are divided into two groups: 82 users randomly selected as training service users and the rest as test service users. The 102 × 3,755 user-service RT matrix is divided into the RT-training matrix and the RT-test matrix, and so is the 102 × 3,755 user-service TP matrix. For the training matrix, we randomly remove entries to make the matrix sparser with density 30 %. Each experiment is performed 100 times, and their average values are taken as results.

Table 2 shows the prediction accuracy comparison of different methods when topK = 5 or topK = 10. TLACF denotes the time-aware and location-aware collaborative filtering algorithm with parameter $d = 0$ and flag $= 0$. TLACF is proposed by these papers.

It can be seen from Table 2 that TLACF has significantly smaller NMAE values, which indicates better prediction performance. Under all the different experimental settings, our approach consistently outperforms other methods.

When topK = 5, it also can be observed from Table 2 that the average percent of improvement in prediction accuracy of our approach against UPCC, IPCC, WSRec, and LACF are 29.27, 29.87, 13.85, and 10.60 %, respectively.

The results shown in this section indicate that the prediction accuracy would be improved, when time factor has been taken into account.

5.7 Prediction efficiency evaluation

Another advantage of our approach is its efficiency. In order to study the efficiency of our approach, the prediction approach proposed by this paper TLACF is also compared with UPCC, IPCC, WSRec, and LACF.

Since time factor has been employed to improve the prediction accuracy in our approach, the efficiency of our approach would be lower than LACF. Therefore, we need to compare the efficiency of our approach with that of LACF to study how much the efficiency has been lost by employing time factor. Moreover, the efficiency of our approach should also be compared with UPCC, IPCC, and WSRec to study how much the efficiency has been improved by employing location factor.

Given settings: topK = 10, density = 30 %, flag = 0 and $d = 0$. Each experiment is performed 100 times and their average values are taken as results. Table 3 compares the average predicting time (unit second) of different methods.

It can be seen from Table 3 that our approach has significantly smaller predicting time than UPCC, IPCC, and WSRec, which indicates that our approach is an efficient approach. Both LACF and our approach TLACF are clustering CF algorithms and classify users and services just according to location information. Since our approach TLACF takes time factor into account, the predicting time of our approach is slightly larger than LACF.

When topK = 10, it also can be observed from Table 3 that the average percent of improvement in efficiency of our approach against UPCC, IPCC, WSRec, and LACF are 448, 967, 1,575, and −14.76 %, respectively. −14.76 % denotes that the efficiency of our approach is lower than LACF.

The results shown in this section indicate that the efficiency would be improved, when location factor has been taken into account.

5.8 Discussion

From those experimental results, we make some important observations.

First, our approach is efficient in predicting the missing QoS values based on the analysis in Sect. 4.8 and the experiment result shown in Sect. 5.7. The improvement is significantly large, since our approach divides the users set and services set into many clusters according to the location information, and then, it tries to seek similar users and services within smaller and highly similar clusters, rather than within the entire database.

The second observation is that the prediction accuracy of our approach is significantly improved. QoS performance of Web services is highly related to invocation time, since the service status (e.g., workload and number of clients) and the network environment (e.g., congestion) change over time. Therefore, time factor would be employed to improve the prediction accuracy. Our experimental results shown in Sect. 5.6 support that claim. QoS performance of Web services is also highly related to locations, since the user-observed QoS performance of Web services is greatly influenced by the network distance and the Internet connections between users and Web services. Our experimental results shown in Sect. 5.3 support that claim.

The third observation is that our approach is easy to build and update. Since our approach classifies users and services only according to the location information, adding and deleting users or services will only update these clusters which contain those users and services, while other clusters will not be affected. This enables our approach to build and update clusters quickly.

Therefore, our approach is capable of addressing the three important challenges of recommender systems–high quality of prediction, high scalability, and easy to build and update.

6 Conclusion

Collaborative filtering is an effective method for Web service selection and recommendation. Memory-based CF algorithms are easy to implement and highly effective, but they suffer from a fundamental problem: inability to scale-up. Model-based CF algorithms, such as clustering CF algorithms, address the scalability problem by seeking users for recommendation within smaller and highly similar clusters, rather than within the entire database. However, they are often time-consuming to build and update.

In this paper, we propose a time-aware and location-aware collaborative filtering algorithm which combines the strengths of memory-based approaches and model-based approaches and overcomes shortcomings of both approaches. To validate our algorithm, this paper conducts series of large-scale experiments based on a real-world Web service QoS dataset. Experimental results show that our approach is capable of addressing the three important challenges of recommender systems—high quality of prediction, high scalability, and easy to build and update.

In future work, we will take other QoS factors and relationships among QoS factors into consideration, study how to incorporate them into QoS prediction and how to alleviate the data sparsity problem. Predicting SLO values will be also taken into account.

Acknowledgments The work described in this paper was supported by the National Natural Science Foundation of China under Grant Nos. 91118004, 61232007 and the Innovation Program of Shanghai Municipal Education Commission (No. 13ZZ023).

References

1. Yu T, Zhang Y, Lin K-J (2007) Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Trans Web. doi:10.1145/1232722.1232728
2. Zhang L-J, Zhang J, Cai H (2007) Services computing. Springer and Tsinghua University Press, Berlin
3. Amazon: Service level agreement for ec2. http://aws.amazon.com/ec2-sla/
4. Leitner P, Ferner J, Hummer W, Dustdar S (2013) Data-driven and automated prediction of service level agreement violations in service compositions. Distrib Parallel Datab 31(3):447–470. doi:10.1007/s10619-013-7125-7
5. Zhang Y, Zheng Z, Lyu MR (2011) WSPred: A time-aware personalized QoS prediction framework for web services. In: Proceedings of IEEE symposium on software reliability engineering. doi:10.1109/ISSRE.2011.17
6. Shardanand U, Maes P (1995) Social information filtering: algorithms for automating 'Word of Mouth'. In: CHI '95 Proceedings

of the SIGCHI conference on human factors in computing systems. doi:10.1145/223904.223931
7. Hill W, Stead L, Rosenstein M, Furnas G (1995) Recommending and evaluating choices in a virtual community of use. In: CHI '95 Proceedings of the SIGCHI conference on human factors in computing systems. doi:10.1145/223904.223929
8. Konstan J, Miller B, Maltz D, Herlocker J, Gordon L, Riedl J (1997) GroupLens: applying collaborative filtering to usenet news. Commun ACM 40:77–87. doi:10.1145/245108.245126
9. Rich E (1979) User modeling via stereotypes. Cogn Sci 3:329–354
10. Linden G, Smith B, York J (2003) Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. doi:10.1109/MIC.2003.1167344
11. Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In:L UAI
12. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) GroupLens: An open architecture for collaborative filtering of netnews. In: Proceedings of CSCW '94. doi:10.1145/192844.192905
13. Zheng Z, Ma H, Lyu MR, King I (2009) WSRec: a collaborative filtering based web service recommendation system. In: Proceedings of IEEE international conference on web services (ICWS 09). doi:10.1109/ICWS.2009.30
14. Jiang Y, Liu J, Tang M, Liu XF (2011) An effective web service recommendation method based on personalized collaborative filtering. In: Proceedings of IEEE international conference on web services (ICWS 11). doi:10.1109/ICWS.2011.38
15. Zhang L, Zhang B, Liu Y, Gao Y, Zhu Z (2010) A web service QoS prediction approach based on collaborative filtering. In: Proceedings of IEEE Asia-Pacific services computing conference. doi:10.1109/APSCC.2010.43
16. Chen X, Liu X, Huang Z, Sun H (2010) RegionKNN: a scalable hybrid collaborative filtering algorithm for personalized Web service recommendation. In: Proceedings of IEEE international conference on web services (ICWS 10). doi:10.1109/ICWS.2010.27
17. Tang M, Jiang Y, Liu J, Liu X (2012) Location-aware collaborative filtering for QoS-based service recommendation. In: IEEE international conference on web services (ICWS). doi:10.1109/ICWS.2012.61
18. Xue G, Lin C, Yang Q, Xi W, Zeng H, Yu Y, Chen Z (2005) Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of SIGIR'05. doi:10.1145/1076034.1076056
19. Salakhutdinov R, Mnih A (2008) Probabilistic matrix factorization. In: Proceedings of NIPS'08, vol 20, pp 1257–1264
20. El Haddad J, Manouvrier M, Rukoz M (2010) Tqos: Transactional and qos-aware selection algorithm for automatic web service composition. IEEE Trans Serv Comput. doi:10.1109/TSC.2010.5
21. Alrifai M, Skoutas D, Risse T (2010) Selecting skyline services for qos-based web service composition. In: Proceedings of WWW'10. doi:10.1145/1772690.1772693
22. Alrifai M, Risse T (2009) Combining global optimization with local selection for efficient QoS-aware service composition. In: Proceedings of WWW'09. doi:10.1145/1526709.1526828
23. Bird C, Nagappan N, Gall H, Murphy B, Devanbu P (2009) Putting it all together: using socio-technical networks to predict failures. In: Proceedings of ISSRE'09. doi:10.1109/ISSRE.2009.17
24. Zheng Z, Lyu M (2010) Collaborative reliability prediction of service-oriented systems. In: Proceedings of ICSE'10. doi:10.1145/1806799.1806809
25. Ma H, King I, Lyu MR (2007) Effective missing data prediction for collaborative filtering. In: SIGIR. doi:10.1145/1277741.1277751
26. Jin R, Chai JY, Si L (2004) An automatic weighting scheme for collaborative filtering. In: SIGIR. doi:10.1145/1008992.1009051
27. Deshpande M, Karypis G (2004) Item-based top-n recommendation. ACM Trans Inf Syst 22:143–177. doi:10.1145/963770.963776
28. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: WWW 2001. doi:10.1145/371920.372071
29. Su X, Khoshgoftaar TM (2009) A survey of collaborative filtering techniques. Adv Artif Intell. doi:10.1155/2009/421425
30. Sarwar et al (2002) Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Fifth international conference on computer and information science
31. Berry M et al (1995) Using linear algebra for intelligent information retrieval. SIAM Rev 37(4):573–595
32. McLaughlin MR, Herlocker JL (2004) A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In: SIGIR. doi:10.1145/1008992.1009050
33. Chee SHS, Han J, Wang K (2001) RecTree: an efficient collaborative filtering method. In: Proceedings of the 3rd international conference on datawarehousing and knowledge discovery. doi:10.1007/3-540-44801-2_15
34. Goldberg K, Roeder T, Gupta D, Perkins C (2001) Eigentaste: a constant time collaborative filtering algorithm. Inf Retr 4(2):133C151. doi:10.1023/A:1011419012209
35. Sarwar BM, Konstan JA, Borchers A, Herlocker J, Miller B, Riedl J (98) Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system. In: Proceedings of CSCW '98. doi:10.1145/289444.289509