Location-Aware and Personalized Collaborative Filtering for Web Service Recommendation

Jianxun Liu, Mingdong Tang, Member, IEEE, Zibin Zheng, Member, IEEE, Xiaoqing (Frank) Liu, Member, IEEE, Saixia Lyu

Abstract— Collaborative Filtering (CF) is widely employed for making Web service recommendation. CF-based Web service recommendation aims to predict missing QoS (Quality-of-Service) values of Web services. Although several CF-based Web service QoS prediction methods have been proposed in recent years, the performance still needs significant improvement. Firstly, existing QoS prediction methods seldom consider personalized influence of users and services when measuring the similarity between users and between services. Secondly, Web service QoS factors, such as response time and throughput, usually depends on the locations of Web services and users. However, existing Web service QoS prediction methods seldom took this observation into consideration. In this paper, we propose a location-aware personalized CF method for Web service recommendation. The proposed method leverages both locations of users and Web services when selecting similar neighbors for the target user or service. The method also includes an enhanced similarity measurement for users and Web services, by taking into account the personalized influence of them. To evaluate the performance of our proposed method, we conduct a set of comprehensive experiments using a real-world Web service dataset. The experimental results indicate that our approach improves the QoS prediction accuracy and computational efficiency significantly, compared to previous CF-based methods.

Index Terms—Web services, service recommendation, QoS prediction, collaborative filtering, location-aware

1 INTRODUCTION

WEB service is a software system designed to support interoperable machine-to-machine interaction over a network. With the prevalence of Service-Oriented Architecture (SOA), more and more Internet applications are constructed by composing Web services. As a consequence, number of Web services has increased rapidly over the last decade. Web service discovery has become a crucial and challenging task for users. In addition to functional requirements, users also want to find Web services that satisfy their personal non-functional requirements. Under this circumstance, service discovery that incorporates non-functional performance of Web services has aroused a great deal of interests in the services computing field [1],[2].

Quality-of-Service (QoS) is widely employed to represent the non-functional performance of Web services [3],[4]. QoS is usually defined as a set of non-functional Collaborative Filtering properties, such as response time, throughput, reliability, and so on. Due to the paramount importance of QoS in building successful service-oriented applications, QoS-based Web service discovery and selection has garnered much attention from both academia and industry [5], [6]. Typically, a user prefers to select a Web service with the best QoS performance, provided that a set of Web service candidates satisfying his/her functional requirements are discovered. In reality, however, it is neither easy nor practical for a user to acquire the QoS for all Web service candidates, due to the following reasons: (1) Web service QoS is highly depend on both users’ and Web services’ circumstances. Therefore, the observed QoS of the same Web service may be different from user to user. (2) Conducting real-world Web service evaluation for obtaining QoS of Web service candidates is both time-consuming and resource-consuming. It is thus impractical for a user to acquire QoS information by invoking all of the service candidates. And (3) some QoS properties (e.g., reputation and reliability) are difficult to be evaluated, since they require both long observation duration and a large number of invocations [7],[8]. These challenges call for more effective approaches to acquire service QoS information.

Copyright © 2015 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.
optimize the accuracy and time efficiency of QoS prediction, there is still room for improvement.

Previous CF-based Web service recommendation methods have rarely taken into account the peculiar characteristics of Web service QoS when making QoS predictions. QoS attributes of Web services such as response time and throughput highly depend on the underlying network conditions, which, however, are usually ignored by the previous work. In this study, we explored several influential factors of Web service QoS (such as user location, service location and QoS variation) and incorporate them into our QoS prediction method. Extended from its preliminary conference version [10], the contribution of this paper is three-fold:

- We proposed an enhanced measurement for computing QoS similarity between different users and between different services. The measurement takes into account the personalized deviation of Web services’ QoS and users’ QoS experiences, in order to improve the accuracy of similarity computation.
- Based on the above enhanced similarity measurement, we proposed a location-aware CF-based Web service QoS prediction method for service recommendation.
- We conducted a set of comprehensive experiments employing a real-world Web service dataset, which demonstrated that the proposed Web service QoS prediction method significantly outperforms previous well-known methods.

The remainder of this paper is organized as follows. Section 2 discusses the background and related work of service recommendation. Section 3 presents the motivation of this work. Section 4 gives an overview of our location-aware Web service recommendation method. Section 5 discusses location information representation, acquisition, and processing. Section 6 introduces similarity computation and similar neighbor selection. Section 7 describes both QoS prediction and Web service recommendation. Section 8 presents the experimental results and Section 9 concludes the paper.

2 RELATED WORK

Collaborative filtering is one of the most popular recommendation techniques, which has been widely used in many recommender systems. In this section, we give a brief survey of CF algorithms, and summarize recent work on CF-based Web service recommendation.

2.1 Collaborative Filtering (CF)

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating) [11]. Formally, a CF domain consists of a set of users $U$, a set of items $I$, and users’ ratings on items. The last is often represented by a user-item matrix $R$, where each entry $r(x,y)$ ($x \in U, y \in I$) represents user $x$’s rating on item $y$. The rating $r(x,y)$ is empty if user $x$ has not yet rated item $y$. Because the number of items that are collected and rated by a user is usually very small, the user-item matrix $R$ is likely to be very sparse. Under this formulation, the task of CF is to predict the values for specific empty entries (i.e., predict a user’s rating for an item). Table 1 is a simple example of a user-rating matrix, where predictions are computed for missing ratings.

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>A TOY EXAMPLE OF USER-ITEM MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User 1</td>
</tr>
<tr>
<td>Item 1</td>
<td>3</td>
</tr>
<tr>
<td>Item 2</td>
<td>-</td>
</tr>
<tr>
<td>Item 3</td>
<td>-</td>
</tr>
</tbody>
</table>

Many efforts have been made in recent years to improve the CF technique. McAuliffe and Herlocker [20] improved the user similarity computation by modifying the conventional Pearson’s Correlation Coefficient (PCC). They adjusted PCC by multiplying it with a weight which is dependent on the number of item ratings in common between the concerned users. The more the item ratings in common, the larger the weight would be. Liu et al. [21] explored the personalized influence of items when measuring the user similarity based on their commonly rated items. They argued that items with more users (i.e., more popular items) should have more effects on the user similarity than those with fewer users (i.e., less popular items). Their argumentation is based on the observation that, two users co-selecting a popular item (e.g., film) is less meaningful than co-selecting an unpopular item regarding whether they share common tastes. Adomavicius and Tuzhilin [22] incorporated additional contextual information (such as time and location) into recommender systems. For example, using the temporal context, a travel recommender system would provide a vacation recommendation in winter very different from the one provided in summer. They demonstrated that incorporating contextual information in essence would improve both the effectiveness and the efficiency of a recommender system.

2.2 Web Service Recommendation

Various recommendation techniques have recently been applied to Web service recommendation, such as the content-based [23],[24], link prediction-based [25],[26], and
CF-based [7],[8],[9]. CF has attracted the most attention for its simplicity and effectiveness. Shao et al. [7] proposed a user-based CF method for QoS-aware Web service recommendation. Zheng et al. [8], [9] combined both user-based and item-based CF algorithm to predict Web service QoS values. Their argued that, for every pair of active user and target Web service, both the QoS experience of the users similar to the active user and the QoS values of the services similar to the target service can be employed for QoS prediction. However, these previous approaches failed to exploit the characteristics of QoS in the similarity computation.

Based on the traditional CF approaches, several enhanced methods have been proposed to improve the prediction accuracy. Wu et al. [27] proposed an improved CF method by using data smoothing for the user-service QoS matrix. Qiu et al. [28] incorporated users’ reputation into CF for Web service QoS prediction and recommendation. Chen et al. [29] recognized the influence of user location in Web service QoS prediction and proposed a scalable CF method. The method groups users into a set of regions according to users’ IP addresses and QoS similarities. When identifying similar users for a target user, instead of searching the entire user set, the method searches the region set. Thus, the time efficiency of QoS prediction is improved. Measuring two users’ closeness by comparing their IP addresses, however, may not be accurate. Moreover, Chen et al. did not take into account the Web service location in the Web service QoS prediction.

Recently, Matrix Factorization (MF) has been successfully employed for accurate and scalable Web service QoS prediction [30], [31], [32]. However, these model-based CF methods may have difficulties in handling dynamics of the user-service interaction matrix. When new interactions between users and services occur, the MF model has to be recomputed from scratch to perform QoS prediction. Therefore, this work focuses on improving memory-based CF by exploiting the characteristics of Web services and service users.

3 Motivation

Now we give a further explanation regarding the motivation of our work. Section 3.1 discusses why QoS variation should be incorporated into user and service similarity measurement. Section 3.2 discusses why location information is helpful in improving QoS prediction.

3.1 Incorporating QoS Variation into User and Service Similarity Measurement

Previous QoS prediction methods assume that the co-invoked Web services have equal contribution weights when computing similarity between two users. We argue that the personalized characteristics (e.g., QoS variation) of both Web services and users should be incorporated into measuring the similarity among users and services.

Web service QoS factors, such as response time, availability and reliability, are usually user-dependent. From different Web services, we can derive different personalized characteristics, based on their QoS values, as perceived by a variety of users. Some Web services may have a very good QoS for all users. For example, the availability is always 100%. This is probable if the Web services are deployed in a high performance Cloud environment. If the QoS is good enough (as in this instance), a small variation of QoS values over all users is likely to be observed. Some Web services may have a very poor QoS for all users. For example, the availability is always below 50%. This is probable if the Web services are deployed in a network environment with poor performance and bandwidth. These Web services are also likely to have small variation of QoS values over different users. Many other Web services may have a relatively large variation of QoS over different users. For example, the availability varies from 50% to 100% for different users. These Web services are considered to be user-sensitive. The following example explains why Web services with different QoS variations could contribute differently when computing the similarity between service users.

Suppose user a and user b have both invoked Web service 1 and 2, and they perceived similar QoS values on them. Provided that the variation of QoS values of Web service 1 among all users is very small (i.e., the QoS of Web service 1 is almost the same to all users), similar perceived QoS however does not indicate that User a and User b are very similar. Therefore, the contribution of Web service 1, to the degree of similarity between User a and User b, is small. On the contrary, if Web service 2 is a user-sensitive service, (i.e., the QoS of Web service 2 varies from user to user), similar perceived QoS really indicates that the two users are similar. Therefore, the contribution of Web service 2 to the similarity between Users a and b is larger than Web service 1.

Likewise, variations of service QoS perceived by service users should also be taken into consideration when computing the similarity between Web services. There could be some service users locating in the hot spots of Internet and with high bandwidth who are likely to observe good performance on most services, and there also could be some service users locating in the less popular spots of Internet and with low bandwidth who are likely to observe poor performance on most services. These service users are considered less service-sensitive, and therefore should contribute less to the service similarity computation. The other users with more normal QoS variation should contribute more to the service similarity computation.

3.2 Incorporating Locations of Users and Services into Similar Neighbor Selection

Web services are deployed on the Internet. Thus, QoS of Web services (such as response time, reliability and throughput) is highly dependent on the performance of the underlying network [33]. If the network between a target user and a target Web service is of high performance, the probability that the user will observe high QoS on the target service will increase. There are several factors affecting the network performance between the target user and the target service. The most important factors include network distance and network bandwidth, which are highly rele-
want to locations of the target user and the target service. When the user and the service are located at different networks which are far away from each other on the Internet, network performance is likely to be poor due to both the transfer delay and the limited bandwidth of links between different networks. In contrast, when the user and the Web service are located in the same network, the user is more likely to observe high network performance. Therefore, the locations of user and service are crucial factors affecting QoS. Fig. 1 provides an example to illustrate why locations of two users can be exploited to improve both the accuracy and efficiency of QoS prediction. Similar examples can also be found to illustrate why service location is also very important for QoS prediction. For the sake of conciseness, we only focus on user location in this example.

Suppose Bob and Alice are two users located in different networks that are far from each other (see Fig. 1). Each observed similar QoS, such as response time and throughput, on two Web services, e.g., Service 1 and Service 2 (The two services might be deployed in some networks that have similar performance to Alice and Bob). According to conventional CF-based QoS prediction methods, the two users are somewhat similar. Thus, they are likely to observe similar QoS on other Web services (e.g., Service 3). However, provided Service 3 was deployed in the same network as Bob, thus being close to Bob but far away from Alice, it’s highly likely that the two users will observe quite different QoS values on Service 3. This is in contradiction with the expectation of conventional CF-based prediction methods. Actually, Alice and Bob are not really similar, but happen to have similar QoS experiences on a few Web services. Conventional QoS prediction methods mishandle this case. By taking locations of users into consideration, we can avoid choosing inappropriate neighbors for the target user, thus improving the accuracy of QoS prediction.

4 OVERVIEW OF OUR WEB SERVICE RECOMMENDATION METHOD

We consider the following scenarios where an active user is searching for high-quality Web services in a Web service discovery system or the system is recommending high-quality Web services to an active user. In these scenarios, predicting QoS values for Web services unknown to the active user is firstly required; then, Web services with satisfactory QoS can be identified and recommended to the user. This work focuses on predicting QoS values of Web services for recommendation. As shown in Figure 2, our Web service recommendation method consists of the following main ingredients:

(1) **User location information handler:** This module obtains location information of a user including the network and the country according to the user’s IP address. It also provides support for efficient user-querying based on location.

(2) **Service location information handler:** This handler acquires additional location information of Web services according to either their URLs or IP addresses. The location information includes the network and the country in which the Web service are located. It also provides functionalities for supporting efficient location-based Web service query.

(3) **Find similar users:** This module finds users who are similar to the active user by considering both the users’ QoS experiences and locations. For accurate user similarity measurement and scalable similar user selection, we propose a weighted user-based PCC via exploring QoS variation of Web services and incorporate user locations into similar user selection.

(4) **Find similar services:** In contrast to finding similar users, this module finds similar Web services for a target service, considering both QoS of Web services as well as service locations. A weighted service-based PCC for measuring similarity between services is proposed.

(5) **User-based QoS prediction:** After a certain number of similar users are identified for the active user, this function aggregates the QoS values they perceived on target Web services, and predicts the missing QoS values for the active user.

(6) **Service-based QoS prediction:** After a certain number of similar services are identified for a target Web service, this function aggregates their QoS values to predict the missing QoS values for the active user.

(7) **Hybrid QoS prediction:** This function combines the user-based QoS prediction and the service-based QoS prediction results, making final QoS predictions. The cold-start problem and data-sparsity problem in QoS predictions are also addressed in this module (details will be provided in Section 7.3). 

(8) **Recommender:** After predicting missing QoS values for all candidate Web services, this function recommends Web services with optimal QoS to the active user.


5 Location Information Representation, Acquisition, and Processing

This section discusses how to represent, acquire, and process location information of both Web services and service users, which lays a necessary foundation for implementing our location-aware Web service recommendation method.

5.1 Location Representation

We represent a user’s location as a triple \((IP_u, ASN_u, CountryID_u)\), where \(IP_u\) denotes the IP address of the user, \(ASN_u\) denotes the ID of the Autonomous System (AS)\(^1\) that \(IP_u\) belongs to, and \(CountryID_u\) denotes the ID of the country that \(IP_u\) belongs to.

Typically, a country has many ASs and an AS is within one country only. The Internet is composed of thousands of ASs that inter-connected with each other. Generally speaking, intra-AS traffic is much better than inter-AS traffic regarding transmission performance, such as response time \([34]\). Also, traffic between neighboring ASs is better than that between distant ASs. Therefore, the Internet AS-level topology has been widely used to measure the distance between Internet users \([34]\). Note that users located in the same AS are not always geographically close, and vice versa. For example, two users located in the same city may be within different ASs. Therefore, even if two users are located in the same city, they may look distant on the Internet if they are within different ASs. This explains why we choose AS instead of other geographic positions, such as latitude and longitude, to represent a user’s location.

Similarly, we model a Web service’s location as \((IP_s, ASN_s, CountryID_s)\), where \(IP_s\) denotes the IP address of the server hosting the service, \(ASN_s\) denotes the ID of the AS that \(IP_s\) belongs to, and \(CountryID_s\) denotes the ID of the country that \(IP_s\) belong to.

The above representation for locations of both users and Web services enables us accurately and easily measure closeness between both users and Web services. We will demonstrate this later in this section.

5.2 Location Information Acquisition

Acquiring the location information of both Web services and service users can be easily done. Because the users’ IP addresses are already known, to obtain full location information of a user, we only need to identify both the AS and the country in which he is located according to his IP address. A number of services and databases are available for this purpose (e.g., the Whois lookup service\(^2\)). In this work, we accomplished the IP to AS mapping and IP to country mapping using the GeoLite Autonomous System Number Database\(^1\). The database is updated every month, ensuring that neither the IP to AS mapping nor the IP to country mapping will be out-of-date.

Acquiring the location information of Web services is similar to acquiring the location information of users. Because the services’ URLs or DSNs are already known, only a prior DSN name to IP address translation is required. This is also easy to be implemented. Table 2 illustrates real IP to AS and IP to country mappings. Each row of the table consists of an IP address block and its associated AS number and country name.

<table>
<thead>
<tr>
<th>Start IP Address</th>
<th>End IP Address</th>
<th>AS Number</th>
<th>Country Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.56.0.0</td>
<td>4.67.63.255</td>
<td>AS863</td>
<td>Canada</td>
</tr>
<tr>
<td>4.67.64.0</td>
<td>4.67.67.255</td>
<td>AS9996</td>
<td>Japan</td>
</tr>
<tr>
<td>4.67.68.0</td>
<td>4.68.247.255</td>
<td>AS863</td>
<td>Canada</td>
</tr>
<tr>
<td>4.68.248.0</td>
<td>4.68.249.31</td>
<td>AS1148</td>
<td>Netherlands</td>
</tr>
<tr>
<td>4.68.294.32</td>
<td>4.71.36.3</td>
<td>AS863</td>
<td>Canada</td>
</tr>
<tr>
<td>4.71.36.4</td>
<td>4.71.36.7</td>
<td>AS1148</td>
<td>Netherlands</td>
</tr>
</tbody>
</table>

5.3 Location Information Processing

To efficiently determine which user is close to the target user, we group users according to their location information so that those within the same group are really close. Likewise, we group Web services according to their location information so that those within the same group are close to each other.

In work \([29]\), users are grouped mainly according to the similarity of their IP addresses. That is, if two users have close IP addresses, they are considered close in location. This seems reasonable but may, in reality, cause inaccuracies. Due to several factors, such as the shortage of IPv4 addresses and the wide application of provider-independent IP addresses, fragmentation of IP prefixes (i.e., IP address blocks allocated to ASs) is increasing \([35]\). Therefore, two IP addresses with close values do not necessarily belong to the same AS or country. Table 2 shows an example that the IP addresses possessed by a network (e.g., AS 863) are unnecessarily continuous. The IP prefixes that are close in value are unnecessarily belonged to the same network or country (e.g., 4.67.68.0 and 4.67.64.0 in Table 2). This indicates that identifying either near users or near Web services using only IP addresses could be problematic.

Different from work \([29]\), we cluster users within the same AS into a group. The group is represented by the AS number and identified as the AS-level group. The AS-level group may be very sparse since there are thousands of ASs in the Internet. We thus further cluster users within the same country into a group. The group is represented by the country ID and identified as the country-level group. Figure 3 illustrates the hierarchy of groups. In a similar matter, we also cluster Web services into groups. In our method, if two users or Web services are located in the same AS, we regard them as close. Likewise, if two users (or Web services) are located in the same country, they are regarded as close. However, the latter has less closeness than the former. In order to efficiently search for both close-by users and Web services, in our implementation, we employ a data structure of hash tables to map every AS number or country ID to the group it represents. Therefore, given the AS number or country ID of a user (or Web service), it costs very little time to retrieve close-by users (or Web services).

---

\(^1\) An autonomous system (AS) is either a single network or a group of networks within the Internet that is controlled by a common network administrator on behalf of a single administrative entity (such as a university and a business enterprise). Each autonomous system has a globally unique ID, named as autonomous system number (ASN).

\(^2\) http://www.whois.net
6 Similarity Computation and Similar Neighbor Selection

In this section, we first formally define notations for the convenience of describing our method and algorithms. We then present a weighted PCC for computing similarity between both users and Web services, which takes their personal QoS characteristics into consideration. Finally, we discuss incorporating locations of both users and Web services into the similar neighbor selection.

6.1 Notations and Definitions

The following defines notations which will be employed in the rest of this paper:

- \( U = \{u_1, u_2, \ldots, u_m\} \) is a set of Web service users, where \( u_j \) (1 ≤ j ≤ m) represents a service user and m is the total number of service users.

- \( I = \{i_1, i_2, \ldots, i_n\} \) is a set of Web service items with similar functionality, where \( i_j \) (1 ≤ j ≤ n) represents a Web service and n is the total number of Web services.

- \( I_u \) represents the subset of Web services invoked by user \( u \). \( I_u \) represents the subset of users that co-invoked Web service \( i \).

- \( R = \{r(u,i) | u \in U, i \in I\} \) is a user-item matrix, where \( r(u,i) \) represents a record of invocation QoS values, e.g., response time, throughput, etc.) generated by the interaction between user \( u \) and Web service \( i \). \( r(u,i) = \text{null} \) if \( u \) has not invoked \( i \) yet. For easy presentation, assume that \( r(u,i) \) only contains a single QoS value.

- \( A = \{a_1, a_2, \ldots, a_s\} \) is a set of AS numbers, where \( s \) is the total number of ASs detected from all users and Web services. Let \( U^a \) represent the subset of users located in the AS numbered \( a \). Similarly, let \( I^a \) represent the subset of Web services located in the AS numbered \( a \).

- \( C = \{c_1, c_2, \ldots, c_t\} \) is a set of country IDs, where \( t \) is the total number of countries detected from all users and Web services. Let \( U^c \) represent the subset of users located in the country with ID \( c \). Similarly, let \( I^c \) represent the subset of Web services located in the country with ID \( c \).

- \( \bar{r}(U^{a}, i) = \text{avg}\{r(u,i) \in R | u \in U^{a} \land r(u,i) \neq \text{null}\} \) is the average QoS value of Web service \( i \) observed by the subset of users \( U^{a} \). If no users in \( U^{a} \) has invoked \( i \), \( \bar{r}(U^{a}, i) = \text{null} \). Similarly, \( \bar{r}(U^{c}, i) \) is the average QoS value of Web service \( i \) observed by the subset of users \( U^{c} \). \( r(U, i) \) (also abbreviated as \( r(i) \)) is the average QoS value of Web service \( i \) observed by all users in \( U \).

- \( \bar{r}(U^{a}) = \{\bar{r}(U^{a}, i) | i \in I\} \) is the vector of the average QoS values of different Web services observed by the subset of users \( U^{a} \) (also referred to as the average QoS vector of \( U^{a} \)). In a similar manner, we define \( \bar{r}(U^{c}) \) and \( \bar{r}(I) \) as the vectors of the average QoS values of different Web services observed by all users in \( U^{c} \) and \( U \) respectively.

- \( \bar{r}(u,I^{a}) = \text{avg}\{r(u,i) \in R | i \in I^{a} \land r(u,i) \neq \text{null}\} \) is the average QoS value of Web services in \( I^{a} \) observed by \( u \). Similarly, \( \bar{r}(u,I^{c}) \) is the average QoS values of Web services in \( I^{c} \) observed by \( u \). \( \bar{r}(I^{a}) \) is the average QoS values of Web services in \( I \) perceived by \( u \).

- \( \bar{r}(I^{a}) = \{\bar{r}(I^{a}, i) | u \in U\} \) is the vector of the average QoS values observed by different users on all Web services in \( I^{a} \) (also referred to as the average QoS vector of \( I^{a} \)). In a similar manner, we define \( \bar{r}(I^{c}) \) and \( \bar{r}(I) \) as the vectors of the average QoS values observed by different users on all Web services in \( I^{c} \) and in \( I \) respectively.

6.2 Similarity Computation based on Weighted PCC

The Pearson Correlation Coefficient (PCC) has been applied in many recommendation systems to compute the similarity between both users and items. In user-based collaborative filtering, the standard PCC used to measure similarity between two users is computed as:

\[
PCC(u,v) = \frac{\sum_{i \in I_{u \cap I_v}} (r(u,i) - \bar{r}(u))(r(v,i) - \bar{r}(v))}{\sqrt{\sum_{i \in I_{u \cap I_v}} (r(u,i) - \bar{r}(u))^2 \sum_{i \in I_{u \cap I_v}} (r(v,i) - \bar{r}(v))^2}}
\]

where \( I_{u \cap I_v} \) is the set of Web services that are co-invoked by both user \( u \) and user \( v \), and \( \bar{r}(v) \) and \( \bar{r}(u) \) represent the average QoS values that user \( v \) and user \( u \) have perceived from all Web services they invoked, respectively. The similarity value calculated in Eq. (1) is within the continuous range of [-1, 1]. The larger the value is, the more similar two users are.

However, Eq. (1) fails to consider the personal influence of Web services on similarity measurement. That is, co-invoked Web services are always given equivalent weights in the measurement of similarity between users. As discussed in Section 3, we argue that Web services with more steady QoS values for all of their users should contribute less to the degree of similarity between users, and vice versa. Therefore, we developed a weighted PCC that incorporates the personality of Web services into similarity computation for users.

We introduce here a deviation-based weight to represent each Web service’s personal contribution when computing the similarity between users. The following steps are used for computing the weight of Web service \( i \) based on its QoS deviation:

- Step 1: QoS normalization. In this step, we transform each QoS value of service \( i, r(u,i) \), to a real number between 0 and 1 by comparing it with the maximum and minimum QoS values of \( i \). There are two cases to be considered. If the QoS criterion concerned is positive,
i.e., a larger QoS value indicates better QoS, Eq. (2) is used to normalize \( r(u,i) \); otherwise, if the QoS criterion is negative, i.e., a smaller QoS value indicates better QoS, Eq. (3) is used to normalize \( r(u,i) \).

\[
\begin{align*}
n(u,i) & = \frac{r(u,i) - \min r(i)}{\max r(i) - \min r(i)} & \text{if } \max r(i) - \min r(i) \\
n(u,i) & = \frac{\max r(i) - r(u,i)}{\max r(i) - \min r(i)} & \text{if } \max r(i) - \min r(i) \neq 0
\end{align*}
\]

where \( r(i) \) represents the set of QoS values of Web service \( i \).

- **Step 2**: Standard deviation computation based on normalized QoS values. For a Web service \( i \), its standard deviation after QoS normalization is computed as

\[
d_i = \begin{cases} 
\sqrt{\frac{\sum_{u \in U_i} (n(u,i)-\overline{n}(i))^2}{|U_i|}}, & \text{if } |U_i| \geq \theta \\
\sqrt{\frac{\sum_{u \in U_i} (n(u,i)-\overline{n}(i))^2}{|U_i| \times |U| / \theta}}, & \text{if } |U_i| < \theta
\end{cases}
\]

where \( \overline{n}(i) \) is the average QoS value of Web service \( i \), \( \theta \) is a threshold for the number of users that have invoked \( i \), i.e., \( U_i \). If \( U_i \) is very small, the standard deviation is likely to be overestimated by the original standard deviation computation formula. The \( \theta \) is used to address the above issue.

- **Step 3**: Weigh generation. For a Web service \( i \), its weight is straightforwardly obtained using

\[
w_i = d_i
\]

The value of \( w_i \) is always in the range \([0, 1]\).

After computing the weight of contribution for every Web service, we develop the following formula for computing similarity degree between users \( u \) and \( v \) by extending Eq. (1):

\[
\text{Sim}(u,v) = \frac{\sum_{u \in U_i, v \in U_j} w_i \frac{(r(u,i)-\overline{r}(i))(r(v,j)-\overline{r}(j))}{\sqrt{\sum_{u \in U_i} w_i (r(u,i)-\overline{r}(i))^2 \sum_{v \in U_j} w_j (r(v,j)-\overline{r}(j))^2}}}\
\]

Eq. (6) incorporates the personalized influence of Web services into user similarity measurement, and implies that Web services with larger weights will contribute more to the two users’ similarity.

As similar as user-based collaborative filtering, previous item-based collaborative filtering methods also often adopt the standard PCC to measure similarity between items. The formula for computing PCC between Web service \( i \) and \( j \) is

\[
PCC(i,j) = \frac{\sum_{u \in U_i \cap U_j} (r(u,i)-\overline{r}(i))(r(u,j)-\overline{r}(j))}{\sqrt{\sum_{u \in U_i \cap U_j} (r(u,i)-\overline{r}(i))^2 \sum_{u \in U_i \cap U_j} (r(u,j)-\overline{r}(j))^2}}
\]

where \( U_i \cap U_j \) is a subset of users that invoked both service \( i \) and \( j \), \( r(u,i) \) and \( r(u,j) \) represent the QoS values of service \( i \) and \( j \) respectively, as perceived by user \( u \). \( \overline{r}(i) \) and \( \overline{r}(j) \) represent the average QoS values of Web service \( i \) and \( j \) respectively. Similar to the user similarity measurement, we also develop a weighted PCC to measure the similarity between Web services. It takes into account the influence of user personality and computes the similarity between two Web services \( i \) and \( j \) as:

\[
\text{Sim}(i,j) = \sqrt{\frac{\sum_{u \in U_i \cap U_j} w_u (r(u,i)-\overline{r}(i))(r(u,j)-\overline{r}(j))}{\sqrt{\sum_{u \in U_i \cap U_j} w_u (r(u,i)-\overline{r}(i))^2 \sum_{u \in U_i \cap U_j} w_u (r(u,j)-\overline{r}(j))^2}}}
\]

where \( w_u \) is the contribution weight of user \( u \), which is defined to be the standard deviation of the normalized QoS values of \( u \). Eq. (8) implies that users with larger weights will contribute more to Web services’ similarity.

### 6.3 Similar Neighbor Selection

Similar neighbor selection is a very important step of CF. Selecting the neighbors right similar to the active user is necessary for accurate missing value prediction. In conventional user-based CF, the Top-K similar neighbor selection algorithm is often employed [8]. It selects \( K \) users that are most similar to the active user as his/her neighbors. Similarly, the Top-K similar neighbor selection algorithm can be employed to select K Web services that are most similar to the target Web service.

There are several problems involved, however, when applying the Top-K similar neighbor selection algorithm to Web service recommendation. Firstly, in practice, some service users have either few similar users or no similar users due to the data sparsity. Traditional Top-K algorithms ignore this problem and still choose the top \( K \) most ones. Because the resulting neighbors are not actually similar to the target user (service), doing this will impair the prediction accuracy. Therefore, removing those neighbors from the top \( K \) similar neighbor set is better if the similarity is no more than 0. Secondly, as previously mentioned, Web service users may happen to perceive similar QoS values on a few Web services. But they are not really similar. Considering the location-relatedness of Web service QoS, we incorporate the locations of both users and Web services into similar neighbor selection.

Let \( N(u) \) denote the resulting subset of similar neighbors for the active user \( u \). We propose a similar neighbor selection algorithm for constructing \( N(u) \) as follows:

- **Step 1**: Obtain the subset of users within the same AS numbered \( a \) as user \( u \), denoting them with \( U^a_u \). Use Eq. (6) to compute the similarity between user \( u \) and each other users \( v \in U^a_u \). Search \( U^a_i \) for the top \( K \) most similar users to \( u \) with similarity greater than 0, and use them to construct \( N(u) \). If fewer than \( K \) users are found, proceed to Step 2; else, return \( N(u) \) as the result.

- **Step 2**: Obtain the subset of users within the same country numbered \( c \) as \( u \), denoting them with \( U^c_u \). Use Eq. (6) to compute the similarity degrees between user \( u \) and each other users \( v \in U^c_u \). Search \( U^c_i \) for the top \( K \) most similar users to \( u \) with similarity greater than 0, and use them to construct \( N(u) \). If fewer than \( K \) users are found, proceed to Step 3; else, return \( N(u) \) as the result.

- **Step 3**: Find the subset of users in \( U \) that have invoked Web service \( i \), denoting them with \( U_i \). Use Eq. (6) to compute the QoS similarity degrees between \( u \) and \( v \) for every user \( v \) in \( U_i \). Search \( U_i \) for the top \( K \) most similar users to \( u \) with similarity greater than 0, and use them to construct \( N(u) \). If fewer than \( K \) users
are obtained, use the actual number of similar users (with similarity greater than 0) found to construct \( N(u) \).

From the above process, we can see that the algorithm first searches local users for similar users. It will search a larger range of users if not enough similar users are found at the previous step. Due to the observation that local users are likely to observe similar QoS on co-invoked Web services, this algorithm has a high probability of finding users similar to the active user in his/her local region. Therefore, the above algorithm is also likely to be more efficient than conventional CF algorithms, which always search the entire user set for similar users.

![Diagram](image_url)

**Fig. 4. Location-aware similar neighbor selection for a target user**

Figure 4 is an example illustrating how we select similar neighbors for a target user. Suppose that \( u_0 \) is the target user. Let the inner circle represent the range of users located in the same AS as \( u_0 \), the outer circle represent the range of users located in the same country as \( u_0 \), and the outerest rectangle represent the range of all users. Let \( K=5 \), and it indicates that five similar neighbors are need to be found for \( u_0 \). With our proposed location-aware neighbor selection algorithm, we first search for similar neighbors in the inner circle. Suppose that \( u_1, u_2, \) and \( u_3 \) are the only three similar neighbors in the inner circle, which are fewer than 5, we therefore turn to the outer circle to find similar neighbors for \( u_0 \). Suppose that \( u_4, u_5, u_6 \), and \( u_7 \) are the most similar five neighbors found in the country of \( u_0 \) and they all satisfy our similarity threshold, the neighbor selection algorithm will stop here and return them as the results. The user \( u_0 \), though was selected in the first round, will eventually be filtered because it is not among the top-5 most similar users to \( u_0 \) in the second-round search.

The similar neighbor selection algorithm for a target Web service is similar to the selection process described above. That is, we first search for similar services in the same AS where the target service is deployed, and then search in the same country where the target service is located, and finally turn to the entire service set when the number of similar services found is still fewer than \( K \).

## 7 QoS VALUE PREDICTION AND WEB SERVICE RECOMMENDATION

### 7.1 User-based QoS Value Prediction

In this subsection, we present a user-based location-aware CF method, named as ULACF. Traditional user-based CF methods usually adopt

\[
\hat{r}_u(v,i) = \hat{r}_u(v,i) + \frac{\sum_{u \in N(u)} \text{Sim}(u,v) \times (r(v,i) - \bar{r}(v))}{\sum_{u \in N(u)} \text{Sim}(u,v)}
\]  

(9)

for missing value predictions. This equation, however, may be inaccurate for Web service QoS value prediction for the following reasons. Web service QoS factors such as response time and throughput, which are objective parameters and their values vary largely. In contrast, user ratings used by traditional recommender systems are subjective and their values are relatively fixed [29]. Therefore, predicting QoS values based on the average QoS values perceived by the active user (i.e., \( \bar{r}(u) \)) is flawed. Moreover, Eq. (9) does not distinguish local and remote users that are similar to the active user. Intuitively, given two users that have the same estimated similarity degree to the target user, the user closer to the target user should be placed more confidence in QoS prediction than the other. Based on the above consideration, we use Eq. (10) to compute the predicted QoS value for the active user based on the QoS experience of his/her similar users

\[
\hat{r}_u(v,i) = \frac{\sum_{u \in N(u)} \text{Conf}(u,v) \times \text{Sim}(u,v) \times r(v,i)}{\sum_{u \in N(u)} \text{Conf}(u,v) \times \text{Sim}(u,v)}
\]  

(10)

where \( \text{Conf}(u,v) \) is the confidence of \( \text{Sim}(u,v) \) in the view of \( u \). This work computes the value of \( \text{Conf}(u,v) \) by considering whether \( u \) and \( v \) are located in the same AS or country, or neither of them. In detail, for different user \( v \) in \( N(u) \), \( \text{Conf}(u,v) \) is computed as follows (Suppose that user \( u \) has AS number \( a \) and country ID \( c \)):

1. If \( v \in U^a \) (\( u \) and \( v \) are within the same AS \( a \)), \( \text{Conf}(u,v) \) is computed with the PCC similarity between the QoS vector of \( u \) and the average QoS vector of \( U^a \), i.e.,

\[
\text{Conf}(u,v) = \text{PCC}(r(u), \bar{r}(U^a))
\]  

(11)

where \( \bar{r}(U^a) \) is the average QoS vector of \( U^a \), as defined in Section 6.1.

2. If \( v \in U^a \land v \in U^c \) (\( u \) and \( v \) are within the same country \( c \)), \( \text{Conf}(u,v) \) is computed with the PCC similarity between the QoS vector of \( u \) and the average QoS vector of \( U^c \), i.e.,

\[
\text{Conf}(u,v) = \text{PCC}(r(u), \bar{r}(U^c))
\]  

(12)

where \( \bar{r}(U^c) \) is the average QoS vector of \( U^c \), as defined in Section 6.1.

3. Otherwise, \( \text{Conf}(u,v) \) is computed with the PCC similarity between the QoS vector of \( u \) and the average QoS vector of \( U \), i.e.,

\[
\text{Conf}(u,v) = \text{PCC}(r(u), \bar{r}(U))
\]  

(13)

where \( \bar{r}(U) \) is the average QoS vector of \( U \), as defined in Section 6.1.

Usually, the confidence of \( \text{Sim}(u,v) \) will be larger when \( u \) and \( v \) are in the same AS or country than when they are in different countries, since nearby users are likely to have more similar QoS experiences than those far-
way from each other, as we indicated in our experiments.

Employing Eq. (10) to predict missing QoS values for active user \( u \) and target service \( i \) may fail when \( N(u) \) is empty or no users in it has invoked service \( i \). In this case, we will use \( \bar{r}(U^+, i) \), \( \bar{r}(U^-, i) \), or \( \bar{r}(i) \) to estimate \( r(u, i) \).

Considering that the closer users’ experiences are likely more reliable, the three values should be in order of decreasing priority. Only when the prior value is null, will the posterior value be used for \( \hat{r}_\lambda(u, i) \). If \( \bar{r}(U^+, i) \), \( \bar{r}(U^-, i) \), and \( \bar{r}(i) \) are all null (when Web service \( i \) is never invoked), we set \( \hat{r}_\lambda(u, i) = null \).

### 7.2 Item-based QoS Value Prediction

In this subsection, we present an item-based location-aware CF method, named as ILACF. Based on the similar consideration as ULACF’s, we use Eq. (11) to compute the predicted QoS value for a service based on the QoS values of its similar services

\[
\hat{r}_r(u, i) = \frac{\sum_{j \in N(u)} \text{Conf}(i, j) \times \text{Sim}(i, j) \times r(u, i)}{\sum_{j \in N(u)} \text{Conf}(i, j) \times \text{Sim}(i, j)}
\]  

(14)

where \( \text{Conf}(i, j) \) is the confidence of \( \text{Sim}(i, j) \) to Web service \( i \). Similarly, we compute the value of \( \text{Conf}(i, j) \) by considering whether \( i \) and \( j \) are located in the same AS or country, or neither of them. In detail, for different service \( j \) in \( N(u) \), \( \text{Conf}(i, j) \) is computed as follows (Suppose that Web service \( i \) has AS number \( a \) and country ID \( c \)):

(1) If \( j \in I^+ (i \) and \( j \) are within the same AS \( a \), \( \text{Conf}(i, j) \) is computed with the PCC similarity between the QoS vector of \( i \) and the average QoS vector of \( I^+ \), i.e.,

\[
\text{Conf}(i, j) = \text{PCC}(r(i), \bar{r}(I^+))
\]  

(15)

where \( \bar{r}(I^+) \) is the average QoS vector of \( I^+ \), as defined in Section 6.1.

(2) If \( j \notin I^+ (i \) and \( j \) are within the same country \( c \), \( \text{Conf}(i, j) \) is computed with the PCC similarity between the QoS vector of \( i \) and the average QoS vector of \( I^+ \), i.e.,

\[
\text{Conf}(i, j) = \text{PCC}(r(i), \bar{r}(I^+))
\]  

(16)

where \( \bar{r}(I^+) \) is the average QoS vector of \( I^+ \), as defined in Section 6.1.

(3) Otherwise, \( \text{Conf}(i, j) \) is computed with the PCC similarity between the QoS vector of \( u \) and the average QoS vector of \( I \), i.e.,

\[
\text{Conf}(i, j) = \text{PCC}(r(i), \bar{r}(I))
\]  

(17)

where \( \bar{r}(I) \) is the average QoS vector of \( I \), as defined in Section 6.1.

Employing Eq. (14) to predict missing QoS values for \( u \) and \( i \) may fail when \( N(i) \) is empty or no services in it has been invoked by \( u \). In this case, we will use \( \bar{r}(U^+, i) \), \( \bar{r}(U^-, i) \), or \( \bar{r}(i) \) to estimate \( r(u, i) \). Considering that the closer services’ QoS values are likely more reliable, the three values should also be in order of decreasing priority. Only when the prior value is null, will the posterior value be used for prediction. If \( \bar{r}(U^+, i) \), \( \bar{r}(U^-, i) \), and \( \bar{r}(i) \) are all null (when \( u \) has never invoked any service), set \( \hat{r}_\lambda(u, i) = null \).

### 7.3 Integrating QoS Predictions

Due to the sparsity of the user-item matrix, to make the missing value prediction as accurate as possible, it’s better to fully explore the information of similar users as well as similar services. Therefore, we develop a hybrid location-aware CF, named as HLACF, which integrated the user-based QoS prediction with the item-based QoS prediction. The following four cases will be considered in integrating QoS predictions:

(1) Case 1: Neither \( \hat{r}_\lambda(u, i) \) nor \( \hat{r}_r(u, i) \) are null. In this case, we integrate the user-based prediction with item-based prediction as:

\[
\hat{r}(u, i) = \lambda \hat{r}_\lambda(u, i) + (1 - \lambda) \hat{r}_r(u, i)
\]  

(18)

where \( \hat{r}(u, i) \) is the final prediction value and \( \lambda \) is a tunable parameter to balance the user-based prediction with the item-based prediction. The parameter \( \lambda \) makes this prediction adaptable to different environments.

(2) Case 2: \( \hat{r}_\lambda(u, i) \) is null while \( \hat{r}_r(u, i) \) is not. Let \( \hat{r}(u, i) = \hat{r}_r(u, i) \).

(3) Case 3: \( \hat{r}_\lambda(u, i) \) is null while \( \hat{r}_r(u, i) \) is not. Let \( \hat{r}(u, i) = \hat{r}_\lambda(u, i) \).

(4) Case 4: Both \( \hat{r}_\lambda(u, i) \) and \( \hat{r}_r(u, i) \) are null. This only occurs when both the active user and the target service are new and have never experienced any invocations. In this case, the missing QoS values will not be predicted. However, this case is very rare.

The above description implies that HLACF can address the data sparsity and cold start problems very well. In traditional CF-based recommendation methods, the data sparsity problem occurs because there are insufficient data in the user-item matrix for CF to measure similarity and identify similar neighbors accurately. However, in HLACF, the data sparsity problem can be greatly relieved, because it exploits not only the QoS information of similar users and similar services, but also the additional location information of users and services. Therefore, even no similar users and services are found by CF because of data sparsity, the QoS data of local users or services can be employed for QoS value prediction. The cold start problem can be viewed as a special case of the data sparsity problem, being caused by the introduction of new users or new items that have insufficient data. The cold start problem can also be well addressed in our method, since it’s easy to find users or services close to a new user or services. The performance of our method under data sparsity was verified in experience, and will be presented later.

### 7.4 Web Service Recommendation

The QoS values can be used for different Web service recommendation scenarios after predicting the missing QoS values for an active user. For instance, when an active user is searching for Web services with specified functionality, the predicted QoS values can help the users discover the Web service with optimal QoS value from a set of candidate services. The QoS prediction method can also identify a set of high-quality Web services, and di-
rectly recommend them to an active user for selection.

8 Experiments

We have conducted a set of experiments to evaluate the performance of our QoS prediction method. We also have conducted experiments to verify the relation between users’ (or Web services’) locality and QoS similarity. More specifically, we addressed the following questions:

- Is there a correlation between location closeness and QoS similarity for either Web services or users? How strong is it?
- Do the parameters $\lambda$ and top-K influence the prediction accuracy? The parameter $\lambda$ determines how much the hybrid QoS prediction method relies on the user-based prediction or the service-based prediction, each contributes to the prediction accuracy.
- How does the data density affect the performance of the QoS prediction? What is the performance of our method under different data sparseness conditions?
- How much better is our approach when compared with other CF-based QoS prediction methods? We compared our approach with several previous, well-known methods, in both prediction accuracy and prediction time.

All experiments were developed with Matlab. They were performed on an HP desktop computer with the following configuration: Intel Core i3 3.20GHz CPU, 2GB RAM with the Windows 7 operating system.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Country Name</th>
<th>Number of Users</th>
<th>Proportion of Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>United States</td>
<td>161</td>
<td>47.49%</td>
</tr>
<tr>
<td>2</td>
<td>Germany</td>
<td>41</td>
<td>12.09%</td>
</tr>
<tr>
<td>3</td>
<td>Japan</td>
<td>16</td>
<td>4.72%</td>
</tr>
<tr>
<td>4</td>
<td>Canada</td>
<td>12</td>
<td>3.54%</td>
</tr>
<tr>
<td>5</td>
<td>Poland</td>
<td>12</td>
<td>3.54%</td>
</tr>
</tbody>
</table>

8.1 Dataset

During our experiments, we adopted a real-world Web service dataset, WSDataPrep dataset 2 [36], published in www.wsdd.com. This dataset contained the QoS records of service invocations on 5825 Web services from 339 users. The dataset can be transformed into a user-service matrix. Each item of the user-service matrix is a pair of values: response time (also called Round Trip Time, RTT) and throughput (TP). Therefore, the original user-service matrix can be decomposed into two simpler matrices: RTT matrix and TP matrix. We used either the RTT matrix or the TP matrix to compute both the user and the service similarities.

This dataset also contained both the IP addresses of all users and the URLs of all Web services. Through analysis, we found that all 339 users were distributed within 137 ASs and 31 countries. Among the 5825 Web services, 5102 Web services are distributed within 1021 ASs and 74 countries. The AS numbers and country names of the other 723 services is unknown because we either failed to transform their URLs into IP addresses or failed to obtain their AS numbers and country names. Table 3 and Table 4 show the top five countries and top five ASs respectively, of users in the dataset. Table 5 and Table 6 show the top five countries and top five ASs respectively, of Web services in the dataset.

8.2 Correlation between Location Closeness and QoS Similarity

In this subsection, we present experimental results on the relation between the location closeness and QoS similarity for both users and Web services. The QoS similarity both between users and between Web services is computed with PCC. To correctly evaluate this relationship, we developed the following two series of experiments:

1) For a user, we first identified its top K similar neighbors based on the QoS similarity measurement. We then calculated the proportion of the user’s similar neighbors that are within the same AS or country of the user. A higher proportion indicates a stronger correlation between location closeness and QoS similarity with respect to users. Because most ASs and countries have only a few users (as mentioned previously), in this experiment we therefore choose the top 5 ASs and countries with the most users, performing calculation for each user in them and taking the average proportion as the result. In a similar manner, we also tested the correlation between location closeness and QoS similarity for Web services.
every pair of users within the same AS or country, which is referred to as Local User Similarity (LUS), denoted by either A-LUS (AS-based) or C-LUS (Country-based). On the other hand, we computed the average QoS similarity between every pair of users across different ASs or countries, which is referred to as Global User Similarity (GUS). Again, depending whether AS or country is regarded, GUS is denoted by A-GUS or C-GUS. Likewise, we also computed both the local service similarity (denoted by either A-LSS or C-LSS) and the global service similarity (denoted by either A-GSS or C-GSS). If each type of local similarity is significantly greater than the corresponding global similarity, the location closeness is considered highly correlated with the QoS similarity. Figure 5 illustrates the results of the first series of experiments described above. The results present how probable the top K similar neighbors of a user are within either the same AS or country as the user. The results also present how probable the top K similar neighbors of a Web service are within either the same AS or country as the Web service. Note that similarity between either users or services is computed using each QoS attribute separately (for simplicity). As a result, we obtained four kinds of QoS similarities:

- User similarity based on RTT (denoted rtt-user);
- User similarity based on TP (denoted tp-user);
- Service similarity based on RTT (denoted rtt-service);
- Service similarity based on TP (denoted tp-service).

We can see from Figure 5(a) that the top K similar neighbors of either a user or a Web service are highly likely to be located in the same country as the user or the service. For example, when K=10, taking tp-service into consideration, on average, more than 90% of the top K similar neighbors of a Web service are located in the same country as the service. Figure 5(b) illustrates the proportion of the top K similar neighbors of either a user or a Web service being located in the same AS as the user or the service. The proportion is also quite high when K is small. As K increases, however, the proportion with respect to user (e.g., rtt-user and tp-user) decreases significantly. This decrease can be explained from Table 4, which indicates that even the 2nd most-user AS has no more than 10 users, not to say the other low-ranked ASs. Hence, with a larger K (e.g., K=10), a user is likely to have not enough similar neighbors in his AS, which indeed will reduce the proportion calculated.

experiments. For simplicity, we again used both RTT and TP separately to calculate the local user (or service) similarity and the global user (or service) similarity respectively. We compared the local user similarity with the global one, regarding both AS and country. We also compared the local service similarity with the global one, regarding both AS and country. We can see in Table 7 that every type of local similarity is significantly greater than its corresponding global similarity. The results therefore justify the relation between the location closeness and the QoS similarity.

From the above results, we can conclude that our location-aware QoS prediction method has a solid basis, because of the strong relation between the locations of users (or Web services) and the Web services’ QoS perceived by the users.

<table>
<thead>
<tr>
<th>TABLE 7</th>
<th>COMPARISON BETWEEN LOCAL SIMILARITY AND GLOBAL SIMILARITY OF BOTH USERS AND SERVICES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C-LUS,</td>
</tr>
<tr>
<td></td>
<td>C-GUS,</td>
</tr>
<tr>
<td>RTT</td>
<td>0.6357,</td>
</tr>
<tr>
<td></td>
<td>0.4153,</td>
</tr>
<tr>
<td>TP</td>
<td>0.7786,</td>
</tr>
<tr>
<td></td>
<td>0.4303,</td>
</tr>
</tbody>
</table>

8.3 Prediction Accuracy Evaluation

The Mean Absolute Error (MAE) is often used in collaborative filtering methods to measure the prediction accuracy. MAE is defined as

\[ MAE = \frac{1}{N} \sum_{i,j} |r(u,i) - \hat{r}(u,i)| \]  

(19)

where \( r(u,i) \) denotes the actual QoS values of Web service \( i \), observed by service user \( u \), \( \hat{r}(u,i) \) represents the predicted QoS values, and \( N \) denotes the number of predicted values. Because different Web service QoS factors have distinct value ranges, we also used the Normalized Mean Absolute Error (NMAE) metric to measure the prediction accuracy. NMAE is defined as

\[ NMAE = \frac{MAE}{\sum_{u,i} |r(u,i)|/N} \]  

(20)

A smaller NMAE value represents higher prediction accuracy.

We compared our method with several well-known prediction methods. These methods include user-based method using PCC (UPCC) [14], item-based method using PCC (IPC) [17], hybrid CF method WSRec [8], location-aware method RegionKNN [29], and Latent Factor Model (LFM) [37].

The 339 service users were divided into two groups: 20 test (active) users and 319 training users. The test users are randomly selected from the top 5 ASs with most users, and all the other users are considered training users. Based on training users and testing users, both the RTT matrix and the TP matrix were divided into a training
tion, we randomly remove entries from both the training matrix and the test matrix to reduce the data density to 20%. The entries removed from the test matrix were used to evaluate the prediction quality. Empirically, we set the parameters used in our method as $K=10$, $\theta=20$, and $\lambda=0.9$. Each experiment was performed 10 times, with an average value taken as the result. To get a reliable error estimate, we evaluate the prediction accuracy by using the average MAE and NMAE value generated from all test users.

Table 8 displays the prediction accuracy of various methods. ULACF represents the proposed user-based location-aware CF method. ILACF represents the proposed item-based location-aware CF method. HLACF represents the hybrid location-aware QoS prediction method. Let $rttmae$ and $rtnmae$ respectively represent the MAE and NMAE performance of RTT, and $tpmae$ and $tpnmae$ respectively represent the MAE and NMAE performance of TP. From Table 8, we can see that HlACF has significantly smaller MAE and NMAE values than the other methods, indicating better prediction performance. The slightly better performance of HLACF compared to ULACF and ILACF also validates the usefulness of combining user-based CF with item-based CF.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>UPCC</th>
<th>ULACF</th>
<th>IPCC</th>
<th>ILACF</th>
<th>WSRec</th>
<th>RegionKNN</th>
<th>LFM</th>
<th>HlACF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$rttmae$</td>
<td>0.5972</td>
<td>0.4162</td>
<td>0.8223</td>
<td>0.5620</td>
<td>0.5220</td>
<td>0.4905</td>
<td>0.5193</td>
<td>0.4214</td>
</tr>
<tr>
<td>$rtnmae$</td>
<td>1.0465</td>
<td>0.7479</td>
<td>1.5836</td>
<td>1.0999</td>
<td>0.9384</td>
<td>0.8814</td>
<td>0.9128</td>
<td>0.7410</td>
</tr>
<tr>
<td>$tpmae$</td>
<td>56.1644</td>
<td>26.9475</td>
<td>49.2627</td>
<td>47.3596</td>
<td>41.4691</td>
<td>37.2570</td>
<td>29.3586</td>
<td>26.7992</td>
</tr>
<tr>
<td>$tpnmae$</td>
<td>2.1308</td>
<td>0.5614</td>
<td>1.0262</td>
<td>0.9866</td>
<td>0.9826</td>
<td>0.7749</td>
<td>0.6106</td>
<td>0.5981</td>
</tr>
</tbody>
</table>

TABLE 9

<table>
<thead>
<tr>
<th>Metrics</th>
<th>UPCC</th>
<th>ULACF</th>
<th>IPCC</th>
<th>ILACF</th>
<th>WSRec</th>
<th>RegionKNN</th>
<th>LFM</th>
<th>HlACF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$rttmae$</td>
<td>0.0549</td>
<td>0.0221</td>
<td>0.0261</td>
<td>0.0278</td>
<td>0.0816</td>
<td>0.0089</td>
<td>0.0045</td>
<td>0.0105</td>
</tr>
<tr>
<td>$rtnmae$</td>
<td>0.0540</td>
<td>0.0218</td>
<td>0.0279</td>
<td>0.0081</td>
<td>0.0825</td>
<td>0.0093</td>
<td>0.0047</td>
<td>0.0108</td>
</tr>
</tbody>
</table>

8.4 Prediction Time Evaluation

In addition to the prediction accuracy, another advantage of our method is its high efficiency of QoS prediction. This indicates that our method is more scalable than traditional CF methods when applied to large-scale service recommender systems. The reason is that, in most cases we can limit similar neighbor searching to a small subset of users (or Web services), especially when $K$ is small. Instead, traditional CF methods have to search the entire user set or Web service set to locate similar neighbors. We conducted an experiment to evaluate the prediction time of our method, and compare it with some existing methods. Given $K=10$ and matrix density=20%, we randomly choose 20 test users from the matrix and perform QoS predictions for these users using different methods. We compute the average prediction time for each missing QoS value of the test users.

Table 9 compares the average prediction time for a missing QoS value produced by various methods such as UPCC, IPCC, WSRec, RegionKNN, LFM, ULACF, ILACF and HlACF. We can see that our proposed methods (including ULACF, ILACF and HlACF) are very efficient and significantly outperform the traditional CF methods such as UPCC, IPCC and WSRec. RegionKNN and LFM are also very efficient. But RegionKNN needs considerable time to build and maintain user clusters, and LFM needs much time to train its model when new QoS values are added to the matrix.

8.5 Impact of $\lambda$

Different datasets have different data correlation characteristics. Parameter $\lambda$ determines how much the hybrid location-aware prediction relies on either the user-based prediction, ULACF, or the item-based prediction, ILACF. Parameter $\lambda$ makes the prediction feasible for various environments. If $\lambda=0$, only ILACF is involved in QoS prediction. If $\lambda=1$, only ULACF is involved in QoS prediction. In other cases, ULACF and ILACF are combined to predict the missing QoS values for active users.

To study the influence of $\lambda$ on our hybrid location-aware CF method, we conducted an experiment, in which we set $K=10$ and varied the value of $\lambda$ from 0 to 1, with a step value of 0.1. Figure 6(a) illustrates the influence of $\lambda$ on the prediction accuracy of both RTT and TP. Our method achieved the best prediction accuracy when $\lambda=0.9$ for both the RTT and the TP prediction. This indicates that ULACF is likely more accurate than ILACF for the Web service dataset we employed.

8.6 Impact of $K$

We are also interested in whether $K$ has significant influence on the prediction accuracy when employing the Top-K similar neighbor selection algorithm. To evaluate the impact of $K$, we conducted an experiment on our proposed method with $\lambda=0.9$. The experimental results are shown in Fig. 6(b). We can see, for the RTT matrix, with the increase of $K$, the prediction accuracy grows initially, then decreases slightly; as for the TP matrix, the prediction accuracy first grows rapidly with the increase of $K$, then reaches a steady status. These indicate that $K$ is no need to give a large value for obtaining optimal performance in our method. This could be caused by that, when a large $K$ is used, many of the similar neighbors selected are from regions other than just the target user’s country. Therefore, those that are not really similar to the target user or service will likely to be involved.

![Fig. 6](image-url)
8.7 Impact of Data Spareness

We investigated the impact of data spareness on the prediction performance from two aspects: prediction accuracy and coverage. The prediction coverage is defined as the percent of the missing QoS values in the user-service matrix that we can compute a prediction. We set $\lambda=0.9$, $K=10$ or $20$, and vary the matrix density from 5% to 30% with a step of 5%. Fig. 7 reports the experimental results of our method (HLACF) on both the TP and the RTT matrices. We can see that the data spareness indeed has significant influence on the prediction accuracy.

Prediction coverage is also an important metric for evaluating a QoS prediction algorithm. Traditional CF is likely to suffer from low prediction coverage when the data used for prediction is very sparse. However, we improve the traditional CF by using not only similar neighbors but also local neighbors to predicting QoS values for an active user. We conducted an experiment to evaluate the impact of data spareness on the prediction coverage, in which, our proposed methods (including ULACF, ILACF and HLACF) were compared with the traditional CF methods such as UPCC and IPCC. Fig. 8 reports the experimental results. We find that, our methods can always achieve nearly 100% prediction coverage, when the matrix density varies from 5% to 30%. By contrast, the traditional CF methods have significantly lower prediction coverage, especially when $K$ is small.

9 CONCLUSION AND FUTURE WORK

This paper presents a personalized location-aware collaborative filtering method for QoS-based Web service recommendation. Aiming at improving the QoS prediction performance, we take into account the personal QoS characteristics of both Web services and users to compute similarity between them. We also incorporate the locations of both Web services and users into similar neighbor selection, for both Web services and users. Comprehensive experiments conducted on a real Web service dataset indicate that our method significantly outperforms previous CF-based Web service recommendation methods.

In the future, we will take more detailed location information into consideration for QoS prediction, such as the Internet’s AS topology. We will also consider incorporating the time factor into QoS prediction, and plan to obtain bigger datasets for evaluating our methods.

ACKNOWLEDGMENT

The authors appreciate the anonymous reviewers for their valuable feedback on this manuscript. The work was supported by the National Natural Science Foundation of China (No. 61100054, 61272063 and 61402168).

REFERENCES

Jianxun Liu received PhD degree in computer science from Shanghai Jiao Tong University in 2003. He is now a professor of Department of Computer Science and Engineering, Hunan University of Science and Technology. His current research interests include workflow management systems, services computing, and cloud computing. He has published about 80 academic papers in international journals or conference proceedings.

Mingdong Tang is an associate professor in the School of Computer Science and Engineering at Hunan University of Science and Technology, Xiangtan, China. He received his Ph.D. degree in Computer Science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2010. His current research interests include services computing, cloud computing, and social networks. He is a member of IEEE and ACM.

Zibin Zheng is an associate research fellow at Shenzhen Research Institute, The Chinese University of Hong Kong. He received Outstanding Ph.D. Thesis Award of The Chinese University of Hong Kong at 2012. ACM SIGSOFT Distinguished Paper Award at ICSE2010, Best Student Paper Award at ICWS2010, and IBM Ph.D. Fellowship Award at 2010. His research interests include service computing and cloud computing.

Xiaqing (Frank) Liu is currently a professor and a director of the McDonnell Douglas Foundation software engineering laboratory at the Missouri University of Science and Technology. He received the PhD degree in computer science from Texas A&M University, College Station, in 1995. His research interests include software engineering, service computing, requirements engineering, collaborative systems, and software engineering applications in many domains.

Saixia Lyu is a master student in the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. He received his B.S. degree in Information and Computing Sciences from Hunan University of Science and Technology, Xiangtan, China, in 2013. His research interests include service recommendation and selection.