Location-Aware Collaborative Filtering for QoS-Based Service Recommendation

Mingdong Tang, Yechun Jiang, Jianxun Liu
School of Computer Science & Engineering
Hunan University of Science & Technology
Xiangtan, Hunan, China
{tangmingdong, jiangyechun1987, ljx529}@gmail.com

Xiaoqing (Frank) Liu
Department of Computer Science
Missouri University of Science & Technology
Rolla, Missouri, USA
fliu@mst.edu

***Abstract*— Collaborative filtering is one of widely used Web service recommendation techniques. In QoS-based Web service recommendation, predicting missing QoS values of services is often required. There have been several methods of Web service recommendation based on collaborative filtering, but seldom have they considered locations of both users and services in predicting QoS values of Web services. Actually, locations of users or services do have remarkable impacts on values of QoS factors, such as response time, throughput, and reliability. In this paper, we propose a method of location-aware collaborative filtering to recommend Web services to users by incorporating locations of both users and services. Different from existing user-based collaborative filtering for finding similar users for a target user, instead of searching entire set of users, we concentrate on users physically near to the target user. Similarly, we also modify existing service similarity measurement of collaborative filtering by employing service location information. After finding similar users and services, we use the similarity measurement to predict missing QoS values based on a hybrid collaborative filtering technique. Web service candidates with the top QoS values are recommended to users. To validate our method, we conduct series of large-scale experiments based on a real-world Web service QoS dataset. Experimental results show that the location-aware method improves performance of recommendation significantly.**

Keywords- *Web Services Recommendation; QoS Prediction; Collabrative Filtering; Location Aware*

I. Introduction

The past several years have witnessed an increasing abundance of open services on the Web. This calls for more effective approaches for Web service discovery, which is a critical issue in services computing [1]. Recently, there has been an increasing interest in employing Collaborative filtering (CF) to recommend Web services to service users [3,4,5,7]. CF has been widely used in commercial recommendation systems [8]. CF algorithms can be divided into two categories: memory-based and model-based [9]. Depending on characterizing relationships between users or product items, memory-based CF has two kinds of approaches: user-based approaches [9] and item-based approaches [15]. The user-based approach recommends to a user product items collected by other users sharing similar tastes; while the item-based approach recommends to a user those items similar to the ones the user preferred in the past.

Since number of Web services with similar functionalities is rising, it's quite important to recommend services considering their non-functional properties, i.e., Quality of Service (QoS) [2]. To carry out a QoS-aware service recommendation, predicting missing QoS values of service candidates is often required. CF can be used to estimate QoS values of a target service for an active user, either based on QoS experiences of the other users similar to the active user or based on QoS records of the other services similar to the target service. Although prior work shows that CF algorithms are effective in predicting QoS values, they seldom investigate why users with similar historic QoS experiences will observe similar QoS values on same Web service in the future, which is to be addressed in this paper.

QoS of Web services is mainly comprised of performance factors that include availability, response time, reliability, throughput, and etc. Values of these QoS factors are usually highly dependent on the network distance between services and service users, i.e. the locations of services and users, which are not fully incorporated in the existing CF recommendation algorithms. Motivated by this, we propose a location aware collaborative filtering method to predict missing QoS values and recommend Web services to users. The rest of this paper is organized as follows: Section II introduces related work and our contributions. Section III presents a motivating scenario. Section IV gives an overview of our method. Section V and Section VI describe our location-aware CF method in detail. Section VII presents experimental results. Section VIII concludes this paper.

II. Related Work and Our Contributions

Numerous research works have been done on collaborative filtering, recommendation systems and Web service discovery, here we only concentrate on QoS-aware Web service recommendation and QoS prediction.

The first work making QoS prediction using collaborative filtering technique was conducted by Shao et al. [3]. They proposed a user-based CF algorithm to predict QoS values. Zheng, et al. [4] proposed a hybrid user-based and item-based CF algorithm to recommend Web services, and carried out a series of large-scale experiments based on real Web services dataset. They also developed an enhanced Pearson Correlation Coefficient (PCC) measurement for user similarity computation, which addressed the problem that PCC often overestimates similarities of users of services who are actually not similar but happen to have similar QoS experience on a few co-invoked Web services.

Both [3] and [4] are based on the observation that if two users have similar QoS experiences on same services, i.e. the services they used have similar QoS values, the two users are similar. However, they failed to recognize and exploit the characteristics of QoS in similarity computation.

In order to improve their accuracy of QoS prediction of Web services, several new enhanced methods are proposed [5,10,11]. Jiang, et al. [10] proposed that the influence of personalization of Web service items should be taken into account when computing degree of similarity between users. That is, more popular services or services with more stable QoS from user to user should contribute less to user similarity measurement. Zhang et al. [11] suggested that it was better to combine users' QoS experiences, environment factor and user input factor to predict Web services QoS values. But how to obtain environment factor and user input factor were not discussed. Chen et al. [5] were the first to recognize the influence of user location in Web services QoS prediction and proposed a novel method. The method group users into a hierarchy of regions according to users' locations and their QoS records, so that the users in a region are similar. When identifying similar users for a target user, instead of searching the entire set of users, the method only searches the regions that the target user belongs to. However, it does not consider service location in recommending Web services.

Compared with prior related works, this paper makes the following major contributions. First, we consider not only locations of users but also of services, and present a hybrid location-aware QoS prediction method. We show that both user location and service location can be used in improving performance and accuracy of QoS prediction significantly. Second, we are the first to use large-scale real Web services QoS dataset to show that there exists a relationship between QoS similarity and user (service) location. That is, we solve the following problem: why should users be similar if they have similar QoS values on the same subset of services? What is behind it? We believe that if two users really have similar QoS experiences, they are highly likely located in same or adjacent areas on the Internet. That partly explains why they are similar. Similarly, we also solve the problem: why are services similar if they have similar QoS for the same subset of users?

III. A Motivating Scenario

In this section, we present a scenario to illustrate the motivation of our work. Assume that there is a Web service recommendation system, which has a Web service repository and many service users. Suppose that services and users are distributed all over the world. Fig. 1 depicts such a scenario. The top part of the diagram represents the Web services recommendation system, and the bottom part represents its underlying network. The dash curves link users and services to their corresponding locations in the underlying network. Suppose that s1 and s2 are weather forecast services, while s3 and s4 are document conversion services such as converting Word (.doc) files to PDF.

Since weather forecast services are related to region, it's highly likely for a user to select the services located in his region. In this scenario, for example, users in China tend to select s1 while users in USA tend to select s2. This indicates that users within the same region are in nature similar in terms of preference as far as region-related services are concerned. When considering region-unrelated services, such as s3, s4, because Internet application performance such as response time, and throughput, are largely dependent on network distance between users and applications (mainly because of transfer delay), it's also reasonable for users to select services near to his region. In other word, users in the same region tend to observe high performance on services running on servers nearby but poor performance on services running on servers faraway. For example, u1 and u2 are likely to observe similar performance when invoking s1, s2, s3, s4, because they are located in the same network. This leads to the observation that users are usually similar regarding performance if they are with close locations.

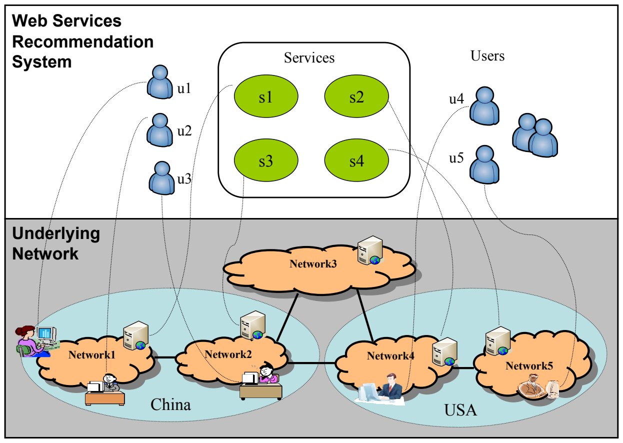

Figure 1. A motivating scenario

Based on the above observation, it's certainly helpful to recommend a user services that were liked by other users located near to him in terms of preference or performance. Therefore, it's highly valuable to take location information into account for QoS prediction, when finding similar users or services, since it not only reduces searching space but also can exclude users who are really not similar but happen to have similar QoS experiences with a few common services.

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

IV. Overview of Our Method

Fig. 2 is an overview of our Web service recommendation method. Suppose that the active user's interest is known, and a service list matching his functional interest is identified. We focus on predicting missing QoS values of the service candidates, which is critical in QoS based service recommendation.

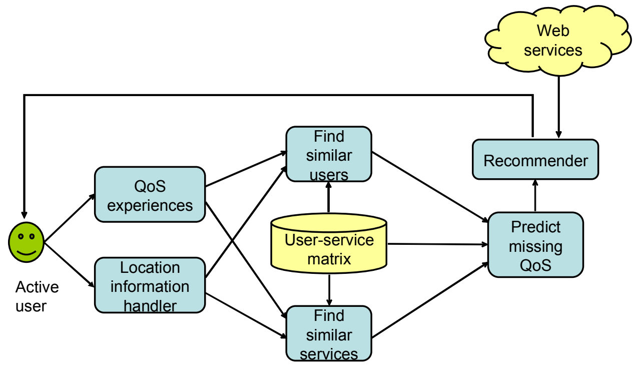

Figure 2. Overview of our recommendation method

In the method, we first acquire historical QoS data and the location information of the active user. A location information handler deals with the location information of both the active user and the target service whose QoS values are missing to the active user. The user-service matrix records every user's QoS experiences on Web services he invoked. To find similar users, user similarity measurement will be computed based on the historical QoS data of the users who are located close to the active user, determined by the location information handler. Likewise, services similarity measurement is computed based on the QoS records of the services which are located close to the target service, also determined by the location information handler.

After finding similar users and similar services for the active user and target service respectively, both user-based CF and item-based CF algorithm are used to predict the missing QoS values of the target service. We synthesize them into a hybrid algorithm to improve the prediction accuracy.

After predicted QoS values of all service candidates are computed, we recommend Web services with top QoS values to the active user.

V. Location Information Representation, Acquisition, and Processing

We represent a user' location as a triple (*IP_u, ASN_u, $CountryID_u$*), where *IP_u* denotes IP address of the user host, *ASN_u* denotes number of the Autonomous System (AS)[1] that *IP_u* belongs to, and *$CountryID_u$* denotes the ID of the country that *IP_u* belong to. Note that users are within a same AS does not definitely mean 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 [6]. 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. That's one reason why we choose AS instead of state (province) or city to represent user location. Similarly, we model a Web service's location as (*IP_s, ASN_s, $CountryID_s$*).

It's easy to acquire and construct the above location information. Because the users' IP addresses are already known, to obtain full location information of a user, we further need to identify the AS and the country he belongs to according to his IP address. There are a lot of available services and tools online for this purpose, e.g. the *Whois* lookup service[2]. In this work we accomplish IP to ASN mapping and IP to country mapping by using the dataset publicly published by Internet topology measurement projects, e.g. the Skitter project[3] and the RouteViews project[4]. Both Skitter and RouteViews data are free to access and frequently updated, which ensures that the IP to ASN or IP to country mapping won't be out-of-date. For the location information of services, its acquisition is similar to users. Since the services' URLs or DSNs are already known, only a prior DSN name to IP address translation is required, which is also straightforward.

The key issue in handling location information is how to measure user similarity or service similarity regarding their locations. In this aspect, our method is also different from the work by Chen et al. [5]. In their method, location similarity is computed based on IP similarity. That is, if two users have similar IP addresses, they are deemed as physically close. This seems to be reasonable but may cause inaccuracies in reality. Due to several factors, such as IPv4 address shortage and multi-homing, IP prefixes (i.e., IP address blocks assigned to networks) are constantly divided into finer granularities [14]. Therefore, two IP addresses with similar values do not necessarily belong to the same network or AS.

TABLE I. Examples of IP to ASN and IP to Country mapping

Start IP Address	End IP Address	AS Number	Country Name
4.56.0.0	4.67.63.255	AS863	Canada
4.67.64.0	4.67.67.255	AS9996	Japan
4.67.68.0	4.68.247.255	AS863	Canada
4.68.248.0	4.68.249.31	AS1148	Netherlands
4.68.294.32	4.71.36.3	AS863	Canada
4.71.36.4	4.71.36.7	AS1148	Netherlands

Table I shows some examples of real IP to ASN and country mapping. Each row of the table contains an IP address block, AS, and country it is assigned to. We can see that the IP addresses possessed by a network (e.g. AS863) are unnecessarily continuous, and the IP addressed with similar values are unnecessarily belonged to the same network or country (e.g. 4.67.68.0 and 4.67.64.0). In this situation, as a contrast, it's better to use ASN and country name to identify location-based similar users. If two users with different IP addresses are within a same AS, they are similar in terms of location; likewise, if two users with different IP addresses and different ASNs are within a same

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

[3] http://www.caida.org

[4] http://www.routeviews.org

country, they are still similar. However, the latter should have a less similarity than the former.

For the convenience of searching a user's or a service's location information and finding physically nearby users, we use a dada structure of hash tables to organize and store the location information of all users and services. Due to limitations of space, we omit the detail of the data structure in this paper.

VI. The QoS Prediction Algorithm

A. *Notations and Definitions*

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

$U = \{u_1, u_2, ..., u_m\}$ is a set of service users, where *m* is total number of service users registered in the recommendation system.

$S = \{s_1, s_2, ..., s_n\}$ is a set of services, where *n* is total number of services collected by the recommendation system.

$M = \{r(u_i, s_j) \mid 1 \le i \le m, 1 \le j \le n\}$ is the user-service matrix, where $r(u_i, s_j)$ is a vector of QoS attribute values acquired from u_i invoking s_j. $r(u_i, s_j) = \phi$ if u_i has no experiences on s_j.

$A = \{a_1, a_2, ..., a_k\}$ is a set of ASNs, where *k* is total number of ASs detected from all users and services. Let U_{a_i} denote the subset of users located in an AS with number a_i. Similarly, S_{a_i} denotes the subset of services located in an AS with number a_i.

$C = \{c_1, c_2, ..., c_l\}$ is a set of country IDs, where *l* is total number of countries detected from all users and services. Let U_{C_i} denote the subset of users located in a country with ID c_i. Similarly, S_{c_i} denotes the subset of services located in a country with ID c_i.

$\bar{r}(U_{ai}, s) = avg\{r(u,s) \mid u \in U_{a_i} \wedge r(u,s) \ne \phi\}$ is a vector of average QoS values of service *s* observed by users in U_{a_i}. Similarly, $\bar{r}(U_{c_i}, s)$ is a vector of average QoS values of service *s* observed by users in U_{c_i} and $\bar{r}(U, s)$ (sometimes abbreviated as $\bar{r}(s)$) is a vector of average QoS values of service *s* observed by all users.

$\bar{r}(u, S_{ai}) = avg\ \{r(u,s) \mid s \in S_{a_i} \wedge r(u,s) \ne \phi\}$ is a vector of average QoS values of services in S_{a_i} observed by *u*. $\bar{r}(u, S_{ci})$ is a vector of average QoS values of services in S_{c_i} observed by *u*. $\bar{r}(u,S)$ (sometimes abbreviated as $\bar{r}(u)$) is a vector of average QoS values of services in S observed by *u*.

B. *Overview of Algorithm*

Here we give a high-level description of our QoS predicting algorithm. Let *u* be an active user and *s* be a target service, of which missing QoS values need to be predicted for *u*. Our QoS predicting algorithm comprises the following three sub-procedures.

1) User similarity computing and user-based QoS predicting. The steps are:

Step 1: Search all users *u'* located in the same AS as *u*, and compute the similarity between each *u* and *u'* regarding their historical QoS experiences. If *u* has similar neighbors, goto Step 4.

Step 2: Similarly, search all users *u'* located in the same country as *u* and compute their similarity between *u'* and *u*. If *u* has similar neighbors, goto Step 4.

Step 3: For all users *u'* (irrespective of locations), compute the QoS similarity between *u* and *u'*. If *u* has similar neighbors, goto Step 4.

Step 4: Let *U'* denote the similar neighbors of *u*. If *U'* is not empty, based on their QoS experiences on *s*, predict the missing QoS values, which is denoted by $\hat{r}_u(u,s)$. Otherwise, set $\hat{r}_u(u,s)$ to be empty.

2) Service similarity computing and service-based QoS predicting. The steps are similar to above, as shown below.

Step 1: Find all services *s'* located in the same AS as *s*, and measure the similarity between each *s* and *s'* regarding their QoS records. If *s* has similar neighbors, goto Step 4.

Step 2: Similar to Step 1, find all services *s'* located in the same country as *s* and compute the similarity between *s'* and *s*. If *s* has similar neighbors, goto Step 4.

Step 3: For all services *s'* (irrespective of locations), compute the similarity between *s'* and *s* regarding their QoS records. If *s* has similar neighbors, goto Step 4.

Step 4: Let *S'* denote the similar neighbors of *s*. If *S'* is not empty, based on their QoS records concerned with *u*, predict the missing QoS values of *s*, denoted by $\hat{r}_s(u,s)$. Otherwise, set $\hat{r}_s(u,s)$ to be empty.

3) Integrating user-based QoS predicting and service-based QoS predicting. The final predicting QoS values are computed as follows:

Case 1: If $\hat{r}_u(u,s)$ and $\hat{r}_s(u,s)$ are both not empty, synthesize them as described later in this section.

Case 2: If either $\hat{r}_u(u,s)$ or $\hat{r}_s(u,s)$ is empty, choose the one that is not empty as the result.

Case 3: Both $\hat{r}_u(u,s)$ and $\hat{r}_s(u,s)$ are empty. This may occur when the user-service matrix is sufficiently sparse or the active user has invoked few services or the target service has few users. The last two are so-called cold-start problem. We will discuss how to solve this problem at the end of this section.

We now further address the following key issues: (1) How to compute user similarity and service similarity? (2) How to select similar neighbors for the active user and the target services? (3) How to integrate $\hat{r}_u(u,s)$ and $\hat{r}_s(u,s)$ to predict the missing QoS values? And (4) how to solve the cold-start problem?

C. *Similarity Computation*

Pearson Correlation Coefficient (PCC) is used in many recommendation systems to compute the similarity. User-based collaborative filtering adopts PCC to compute similarity between two users *w* and *u*. Its computational formula is as Formula (1):

$$sim(w,u)=\frac{\sum_{s\in I}\left(r(w,s)-\bar{r}(w)\right)\left(r(u,s)-\bar{r}(u)\right)}{\sqrt{\sum_{s\in I}\left(r(w,s)-\bar{r}(w)\right)^2}\sqrt{\sum_{s\in I}\left(r(u,s)-\bar{r}(u)\right)^2}} \quad (1)$$

where $sim(w,u)$ denotes degree of similarity between user *w* and user *u*, I is the set of Web service items that are commonly invoked by user *w* and user *u*. $r(w,s)$ and $r(u,s)$ denotes the vector of QoS values which were produced when user *w* and user *u* invoke service s respectively. $\bar{r}(w)$ and $\bar{r}(u)$ represent the vector of average QoS values of user *w* and user *u* respectively. It can be seen from the Formula (1) that $sim(a,u)$ is in the interval of [-1, 1]. The larger a value is, the more similar two users are.

As similar as the user-based collaborative filtering, Item-based collaborative filtering adopts PCC to compute similarity between Web service i and j . The computational formula is as Formula (2):

$$sim(i,j)=\frac{\sum_{u\in V}\left(r(u,i)-\bar{r}(i)\right)\left(r(u,j)-\bar{r}(j)\right)}{\sqrt{\sum_{u\in V}\left(r(u,i)-\bar{r}(i)\right)^2}\sqrt{\left(r(u,j)-\bar{r}(j)\right)^2}} \quad (2)$$

where $sim(i,j)$ denotes degree of similarity between service i and j , V is a set of users that commonly invoke service i and j , $r(u,i)$ and $r(u,j)$ denote the vector of QoS values which were produced when user u invokes service i and j respectively. $\bar{r}(i)$ and $\bar{r}(j)$ represent the vector of average QoS values of service i and j respectively.

D. *Similar Neighbors Selection and Missing Values Prediction*

After computing the degree of similarity between the active user and all other users, we get a user similarity vector. Likewise, we also get a service similarity vector which is composed by the degree of similarity between the target service and all other services. Therefore, two sets of neighbors can be defined. Similar to [4], we adapted the traditional *Top-K* algorithm, i.e., after identifying *Top-K* similar neighbors for the active user (or the target service), the similar neighbors whose similarity is equal to or smaller than 0 will be removed.

Denote the set of similar neighbors of user *u* by $N(u)$ and the set of similar neighbors of service *s* by $N(s)$.The user-based method uses Formula (3) to predict missing QoS values for the active user.

$$\hat{r}_u(u,s)=\bar{r}(u)+\frac{\sum_{u'\in N(u)}Sim(u',u)\left(r(u',s)-\bar{r}(u')\right)}{\sum_{u'\in N(u)}Sim(u',u)} \quad (3)$$

where $\bar{r}(u)$ denotes average QoS values of the active user, and $\bar{r}(u')$ denotes the average QoS values of user u'.

Similarly, the item-based method uses Formula (4) to predict the missing QoS values for the active user.

$$\hat{r}_s(u,s)=\bar{r}(s)+\frac{\sum_{s'\in N(s)}Sim(s',s)\left(r(s',s)-\bar{r}(s')\right)}{\sum_{s'\in N(s)}Sim(s',s)} \quad (4)$$

where $\bar{r}(s)$ denotes average QoS values of service s , and $\bar{r}(s')$ denotes the average QoS values of service item s'.

Since these two predicted values may have different prediction performance, we use a tunable parameter λ to balance these two predicted values. We combine the two methods by using Formula (5):

$$\hat{r}(u,s)=\lambda\hat{r}_u(u,s)+(1-\lambda)\hat{r}_s(u,s) \quad (5)$$

By the above mechanism, the parameter λ determines how much the hybrid method relies on the location-aware UPCC prediction or the location-aware IPCC prediction. The parameter λ makes the method feasible to different environment.

E. *Addressing The Cold-Start Problem*

As stated before, the CF algorithm will fail in predicting the missing QoS values if no similar users and services can be found. This may occur when the user-service matrix is sufficiently sparse or the active user has invoked few services or the target service has few users, in particular, when the active user and the target are newly registered.

To address above problem, we use $\bar{r}(U_a,s)$, $\bar{r}(U_c,s)$, or $\bar{r}(U,s)$ ((i.e. $\bar{r}(s)$) as the predicting QoS values, where *a* and *c* are respectively the number of the AS and the ID of the country that *u* is located in. (See the beginning of this section for definitions of $\bar{r}(U_a,s)$, $\bar{r}(U_c,s)$ and $\bar{r}(U,s)$). Noted that $\bar{r}(U_a,s)$, $\bar{r}(U_c,s)$, and $\bar{r}(U,s)$ can be computed in advance, and only when the prior is empty, can the posterior be used for prediction.

$\bar{r}(U_a,s)$, $\bar{r}(U_c,s)$, and $\bar{r}(U,s)$ may all be empty if *s* has been never been invoked before. In this case, we use $\bar{r}(u,S_a)$, $\bar{r}(u,S_c)$,or $\bar{r}(u,S)$ (in a order of decreasing priority) as the predicting QoS values, where *a* and *c* are respectively the number of the AS and the ID of the country that *s* is located. (For definitions of $\bar{r}(u,S_a)$, $\bar{r}(u,S_c)$ and $\bar{r}(u,S)$, also see the beginning of this section) .

It's not difficult to see, only in the case that both the active user and the target service have never experienced invocations, will our QoS prediction method report a failure.

VII. Experiments

Since a key component of Web service recommendation is the QoS value prediction, we use the QoS prediction accuracy to measure recommendation quality. We conduct a set of experiments to evaluate and validate our new method. Particularly, we address the following questions through experiments:

1) How does user location or service location affect the QoS values of services? Does closeness in location indicates similarity in QoS values?

2) How does parameter λ influence prediction accuracy? The parameter λ determines how much the hybrid method relies on the location-aware UPCC prediction or the location-aware IPCC prediction which have direct influence on prediction accuracy.

3) How does our approach compare with other CF-based Web service recommendation methods? Our experiments are developed by using Matlab and performed on an HP desktop computer with configuration as: Intel Core i3 3.20GHz CPU, 2GB RAM, and Windows 7 operating system.

A. Dataset

In our experiments, we adopt a real-world Web service dataset: WSDream dataset 2[5], which was published in references [12,13]. The dataset contains QoS records of service invocations on 5825 Web services from 339 service users, which are transformed into a user-service matrix. Each item of the user-service matrix is a pair of values: 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 use either RTT matrix or TP matrix to compute user similarity and service similarity in the experiments.

TABLE II. TOP 5 COUNTRIES WITH MOST SERVICES

Rank	Country Name	Number of Services	Proportion of Services
1	United States	2389	0.4101
2	United Kingdom	510	0.0876
3	Canada	432	0.0742
4	Germany	298	0.0512
5	China	271	0.0465

TABLE III. TOP 5 AUTONOMOUS SYSTEMS WITH MOST SERVICES

Rank	AS Number	Number of Services	Proportion of Services
1	AS271	281	0.0482
2	AS786	257	0.0441
3	AS4134	160	0.0275
4	AS26496	155	0.0266
5	AS11426	125	0.0215

The dataset also contains IP addresses of all users and URLs of all Web services. According to IP addresses of users and URLs of Web services, we identify locations of all users and services. We find that all 339 users are distributed within 137 ASs and 31 countries, and 5102 services among all are distributed within 1021 ASs and 74 countries. The location distributions of the other 723 services are unknown because we fail to transform their URLs into IP addresses. Table II and Table III show top 5 countries and top 5 ASs with most services in the dataset.

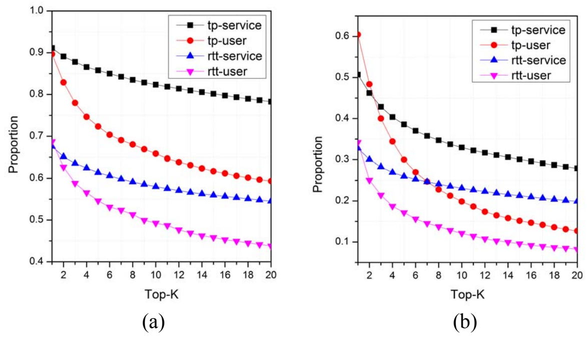

Figure 3. Proportion of local neighbors of a user (or service) among all its Top-K similar neighbors. (a) Country regarded; (b) AS regarded.

B. Relationship Between Location Closeness and QoS Similarity

To evaluate relationship between location closeness and QoS similarity, we conduct the following experiments:

1) For each user, first identify its *TOP-K* similar neighbors based on QoS similarity measurement, and then find out average probability that a user and its neighbors belong to same AS or country. Higher probability indicates higher relationship between location closeness and QoS similarity. Similarly, we also test the relationship between location closeness and QoS similarity using data of services.

2) For each AS or country, compute average QoS similarity between every pair of users within it, and compute an average value across all ASs or countries. We call this value as *user-based average internal QoS similarity* and denote it by *U-IQS*$_A$ or *U-IQS*$_C$. Furthermore, for each AS or country, we also first compute an average QoS similarity between users within it and the other users outside of the AS or country, and then compute an average value across all ASs or countries, which is called as *user-based average external QoS similarity* and denoted by *U-EQS*$_A$ or *U-EQS*$_C$. Similarly, we compute *service-based average internal QoS similarity* (denoted by *S-IQS*$_A$ or *S-IQS*$_C$) and *service-based average external QoS similarity* (denoted by *S-EQS*$_A$ or *S-EQS*$_C$). If each *average internal QoS similarity* is significantly greater than corresponding *average external QoS similarity*, it indicates that location closeness is highly correlated to QoS similarity.

Fig. 3 presents results of the first experiment above. Note that QoS similarity among users or services is computed using each QoS attribute separately (for simplicity), therefore we obtain 4 sequences of similar pairs, i.e., similar user pairs based on RTT measurement (i.e., rtt-user), similar user pairs based on TP measurement (i.e., tp-user), similar service pairs based on RTT measurement (i.e.,

[5] http://www.wsdream.net:8080/wsdream/

rtt-service), and similar service pairs based on TP measurement (i.e., tp-service).

We can see from Fig. 3 (a), the *Top-K* similar neighbors of a user or service have high probability to be located in the same country as the user or the service. For example, when *Top-K*=1 and considering tp-service, in average more than 90% of the *Top-K* similar neighbors of a service are located in the same country as the service. Fig.3 (b) shows the probabilities of *Top-K* similar neighbors of a user or a service being located in the same *AS* as the user or the service. The results in Fig.3 (b) seem to be not as good as those in Fig.3 (a), which may be due to the fact that most ASs has only a very small number of users and services in the dataset, hence a user or service within them must have similar neighbors from other ASs, especially when *Top-K* becomes large. Another reason affecting the experimental results may be that there are many services with poor availability or reliability and thus their QoS values are very low to all users (e.g., RTT is very large due to time out, or TP is almost 0), which may result in an overestimated high similarity between such services as well as between the users who have invoked such services. We find out that this phenomenon does exist and certainly reduces the correlation between location closeness and QoS similarity.

TABLE IV. COMPARISON OF AVERAGE INTERNAL QOS SIMILARITY AND AVERAGE EXTERNAL QOS SIMILARITY

	U-IQS$_C$*, *U-EQS$_C$	***U-IQS$_A$*, *U-EQS$_A$***	***S-IQS$_C$*, *S-EQS$_C$***	***S-IQS$_A$*, *S-EQS$_A$***
RTT	0.6357, 0.4153	0.8164, 0.4526	0.4774, 0.2494	0.4502, 0.2784
TP	0.7786, 0.4303	0.6336, 0.4488	0.5869, 0.0828	0.5710, 0.0985

Table IV shows comparison of *average internal QoS similarity* and *average external QoS similarity* regarding both country and AS. Again, the QoS similarity is computed using each QoS attribute separately. We can see that every average internal QoS similarity is significantly greater than its corresponding average external QoS similarity, which forms a solid basis for our location aware CF recommendation algorithm.

C. Prediction Accuracy Evaluation

Mean Absolute Error (MAE) is often used in collaborative filtering methods to measure the prediction quality. MAE is defined as Formula (6):

$$MAE = \frac{\sum_{i,j} \left| r(i,j) - \hat{r}(i,j) \right|}{N} \qquad (6)$$

where $r(i,j)$ denotes actual QoS values of Web service j observed by service user i, and $\hat{r}(i,j)$ represents the predicted QoS values of service *j* for user *i*, and N denotes the number of predicted values. Because different QoS properties of Web services have distinct value ranges, like [4], we also use the Normalized Mean Absolute Error (NMAE) metric to measure the prediction accuracy. NMAE is defined as:

$$NMAE = \frac{MAE}{\sum_{i,j} r_{i,j} / N} \qquad (7)$$

And smaller NMAE values represent higher prediction accuracy.

We compare our method with other well-known prediction methods: user-based algorithm using PCC (UPCC) [9], item-based algorithm using PCC (IPCC) [15], and hybrid algorithm WSRec [4] to evaluate its prediction performance. Formula (3) and Formula (4) are used for computing the UPCC and IPCC respectively. Each experiment is performed 100 times and their average values are taken as results.

The 339 service users are divided into two groups: 300 randomly selected training users and the rest as test (active) users. The RTT matrix is divided into the RTT-training-matrix and the RTT-test-matrix, and so is the TP matrix. In order to simulate the real world situation, we randomly remove entries of the training matrices and the test matrices to reduce data density to 20%. The removed entries of the test matrices are used to evaluate the prediction quality. We set λ =0.7 and λ =0.8 for RTT and TP predicting respectively, in which setting our method has best performance, as shown in Fig. 4.

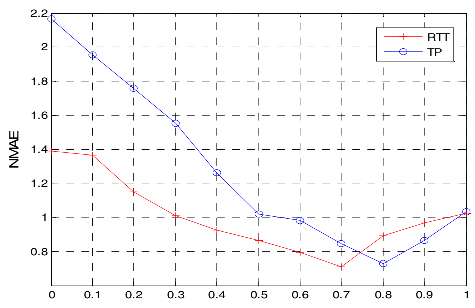

Figure 4. Impacts of λ on prediction accuracy (*Top-K*=10).

Table V shows the prediction accuracy comparison of different methods when *Top-K*=5 or *Top-K*=10. UPCC denotes the traditional user-based algorithm using PCC, LA-UPCC denotes the adapted user-based CF algorithm by considering locations of users and services. IPCC denotes the traditional item-based algorithm using PCC, LA-IPCC denotes the adapted item-based CF algorithm by considering locations of users and services. WSRec [4] is a hybrid algorithm combing UPCC and IPCC. The LACF denotes our hybrid location-aware collaborative filtering algorithm. It can be seen from the table that LACF has significantly smaller MAE and NMAE values, which indicates better prediction performance. It also can be observed from the table that the prediction accuracy under setting *Top-K*=10 is slightly higher than under *Top-K*=5 in most cases.

D. Prediction Efficiency Evaluation

Another advantage of our prediction method is its efficiency. This is because that we can limit similar user or service search to a much smaller domain in contrast to previous CF-based QoS prediction methods.

TABLE V. COMPARISON OF PREDICTION ACCURACY OF DIFFERENT METHODS

	***Top-K*=5**				***Top-K*=10**			
	RTT		**TP**		**RTT**		**TP**	
	MAE	**NMAE**	**MAE**	**NMAE**	**MAE**	**NMAE**	**MAE**	**NMAE**
UPCC	0.7944	1.1451	43.9897	1.2144	0.7281	1.0256	41.5676	1.0317
IPCC	1.0025	1.6558	99.3364	2.2285	0.9576	1.3861	90.7544	2.1671
WSRec	0.6921	0.9889	41.9882	1.0103	0.6384	0.9682	40.2146	0.9863
LA-UPCC	0.6612	0.9801	35.2262	0.8236	0.5847	0.7627	33.5492	0.7995
LA-IPCC	1.0201	1.5172	73.9172	1.6683	0.8750	1.2101	65.0855	1.5934
LACF	0.5988	0.7682	32.6430	0.7967	0.5431	0.7104	31.5120	0.7296

We carry out experiments to evaluate the predicting time of LA-UPCC, LA-IPCC and LACF, and compare it with other existing methods such as WSRec, IPCC, UPCC. Given settings: *Top-K*=10, density=20%, we totally predict 100 missing values, and compute their average predicting time. Table VI compares the average predicting time (unit second) of different methods. We can see our methods are all very efficient due to that their predicting time is much shorter than that of the other methods.

TABLE VI. COMPARISON OF PREDICTING TIME OF DIFFERENT METHODS (SECOND)

	UPCC	**LA-UPCC**	**IPCC**	**LA-IPCC**	**WSRec**	**LACF**
RTT	0.0516	0.0038	0.1012	0.0037	0.1532	0.0076
TP	0.0477	0.0026	0.0907	0.0034	0.1385	0.0061

VIII. CONCLUSIONS

This paper presents a location-aware collaborative filtering method for QoS value prediction and QoS-based Web service recommendation. Based on our observation that a strong relationship exists between users' location closeness of and users' QoS similarity, we propose a location-aware UPCC method to identify similar users for an active user. Likewise, we propose a location-aware IPCC method to identify similar services for a target service. We integrate location-aware UPCC and IPCC into a hybrid collaborative filtering method. Through experiments on a large-scale real-world Web services dataset, we show that the performance of our method outperforms existing collaborative filtering based recommendation methods by a significant improvement of both prediction effectiveness and efficiency. In future work, we will take relationships among QoS factors into consideration and study how to incorporate them into QoS prediction.

ACKNOWLEDGMENT

The work was supported by National Natural Science Foundation of China under grant No.90818004 and No.61100054, Program for New Century Excellent Talents in University under grant No.NCET-10-0140, Excellent Youth Foundation of Hunan Scientific Committee No.11JJ1011, and Scientific Research Fund of Hunan Educational Committee under grant No.09K085 and No.11B048.

REFERENCES

[1] L.-J. Zhang, J. Zhang, and H. Cai, Services computing, Springer and Tsinghua University Press, 2007.

[2] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," ACM Transactions on Web, 1(1):6, 2007.

[3] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS prediction for Web services via collaborative filtering," Proc. IEEE International Conference on Web Services (ICWS 07), 2007, pp. 439-446.

[4] Z. Zheng, H. Ma, M.R. Lyu, and I.King, "WSRec: a collaborative filtering based web service recommendation system," Proc. IEEE International Conference on Web Services (ICWS 09), 2009, pp. 437-444.

[5] X. Chen, X. Liu, Z. Huang, and H. Sun, "RegionKNN: a scalable hybrid collaborative filtering algorithm for personalized Web service recommendation," Proc. IEEE International Conference on Web Services (ICWS 10), 2010, pp. 9-16.

[6] G. Zhang, and G. Zhang, "Agent selection and P2P overlay construction using global locality knowledge," Proc. IEEE International Conference on Networking, Sensing and Control (ICNSC 07), 2007, pp. 519 - 524.

[7] E. Rich, "User modeling via stereotypes," Cognitive Science, Vol.3, No.4, 1979, pp. 329-354.

[8] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," IEEE Internet Computing, 2003, pp. 76 - 80.

[9] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," Proc. 14th Conference on Uncertainty in Artificial Intelligence(UAI 98), 1998, pp. 43-52.

[10] Y. Jiang, J. Liu, M. Tang, and X. F. Liu, "An effective Web service recommendation method based on personalized collaborative filtering," Proc. IEEE International Conference on Web Services (ICWS 11), 2011, pp. 211 - 218.

[11] L. Zhang, B. Zhang, Y. Liu, Y. Gao, Z. Zhu, "A Web service QoS prediction approach based on collaborative filtering," Proc. IEEE Asia-Pacific Services Computing Conference, 2010, pp.725-731.

[12] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world Web services," Proc. IEEE International Conference on Web Services (ICWS 10), 2010, pp.83-90.

[13] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based QoS prediction in cloud computing," Proc. IEEE Symposium on Reliable Distributed Systems (SRDS 11), 2011, pp.1-10.

[14] G. Huston, BGP Routing Table Analysis Reports, http://bgp.potaroo.net/.

[15] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," Proc. ACM Conference on Computer Supported Cooperative Work (CSCW 94), 1994, pp.175-186.