Computing manuscript No.
(will be inserted by the editor)

A Location-sensitive and Network-aware Broker for Recommending Web Services

Saad Saeed · Muhammad Asim · Thar Baker · Zakaria Maamar

Received: date / Accepted: date

Abstract Collaborative Filtering (CF) is one of the renowned recommendation techniques that can be used for predicting unavailable Quality-of-Service (QoS) values of Web services. Although several CF-based approaches have been proposed in recent years, the accuracy of the QoS values, that these approaches provide, raises some concerns and hence, could undermine the real "quality" of Web services. To address these concerns, context information such as communication-network configuration and user location could be integrated into the process of developing recommendations. Building upon such context information, this paper proposes a CF-based Web Services recommendation approach, which incorporates the effect of locations of users, communication-network configurations of users, and Web services run-time environments on the recommendations. To evaluate the accuracy of the recommended Web services based on the defined QoS values a set of comprehensive experiments are conducted using a real dataset of Web services. The experiments are in line with the importance of integrating context into recommendations.

Keywords Collaborative filtering · Context · Quality- of-Service · Recommendation · Web service

Saad Saeed
Department of Computer Science, FAST-NUCES, Islamabad, Pakistan.
E-mail: saadsaeed01@live.com

Muhammad Asim
Department of Computer Science, FAST-NUCES, Islamabad, Pakistan
E-mail: muhammad.asim@nu.edu.pk

Thar Baker
Department of Computer Science, Liverpool John Moores University, Liverpool, UK.
E-mail: T.baker@ljmu.ac.uk

Zakaria Maamar
College of Technological Innovation, Zayed University, Dubai, UAE.
E-mail: zakaria.maamar@zu.ac.ae

1 Introduction

Service-Oriented Architecture (SOA) promotes the use of self-descriptive software components, usually known as Web Services (WSs), to achieve the interoperability of disparate business applications, sometimes, deployed on the cloud. With the abundance of similarly-functional WSs [1], potential users need assistance with selecting those WSs that meet their needs and satisfy their requirements. Thus, a WS recommendation approach would be highly appreciated by users [2, 3].

The quality of any recommendation in terms of validity and reliability is dependent on many factors. In the field of WSs, Quality-of-Service (QoS) (*aka* non-functional properties) plays a crucial role in WSs discovery and selection prior to their inclusion in business applications. QoS describe a WS's non-functional characteristics such as *response time*, *throughput*, *availability*, and *reliability* [15]. Yet, acquiring QoS values is known to be "tedious" and prone-to-errors due to multiple reasons: (*i*) effective QoS values, usually, depend on both users' and WSs' operating environments (e.g., location, bandwidth, and platform) and (*ii*) obtaining QoS values from real WSs can be time and resource consuming; long-duration observations and numerous remotely deployed WS invocations are required [5, 11, 29, 31].

Several techniques have been proposed for WSs recommendation, such as Collaborative Filtering (CF), content-based, and link prediction-based [11]. CF has recieved the most attention due to its efficiency and simplicity. CF predicts and, then, recommends a particular WS for a particular user based on historic data collected from other users who experienced the same WS. Formally, a CF-based application contains a set of users ($u_1, \cdots, u_n$), a set of WSs ($s_1, \cdots, s_m$), and users' experiences with those WSs. These experiences are often represented in a user-service Matrix (M, Table 1). Each entry $m(u,s)$ in M represents the QoS value of s_i that u_j has invoked. CF effectiveness is dependent on how accurate the estimated QoS values are.

Let us consider a real-world scenario where 4 users in different locations invoked 4 WSs located in different geographic areas, each area has a particular network configuration, in terms of bandwidth and latency, among other factors. In Fig. 1, the dotted lines represent the invocation of users to WSs. The QoS values in term of response time (*aka* round trip time) are recorded from users after invoking WSs and depicted in Table 1. In this table, ? indicates a missing value, i.e., the user has not invoked the WS, yet. Traditional CF approaches compute the similarity between users without considering contextual information, such as network configuration, platform types and capabilities, and user location. This information would definetly have an impact on the quality of recommendation, when the available QoS information is largely sparse or there exists limited or no interaction between users and WSs, like in the current example. Existing CF approaches have failed to provide accurate similarity between users [26]. Indeed, a number of recent CF approaches have acknowledged the influence of contextual information. For instance, users in the same region are **likely** to experience the same response time when invoking the same WS [11]. Thus, it is reasonable to fill the missing QoS values for u_2 in Table 1 based on the QoS values of WSs experienced by u_1 and *vice versa*. This would allow to have a richer WS invocation matrix, which should enable CF to perform well.

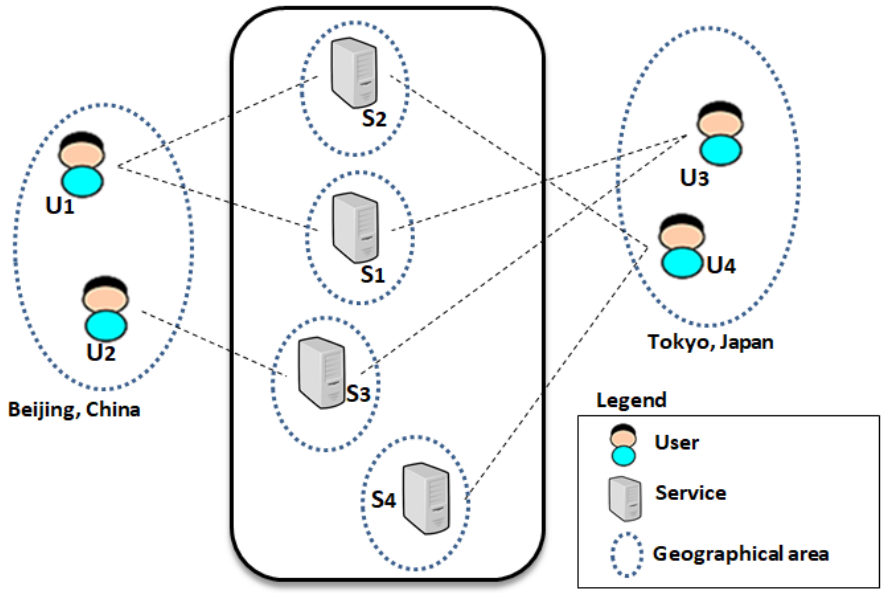

Fig. 1 Representation of user-Web service invocation

Table 1 Incomplete matrix of user-service invocation

	S_1	S_2	S_3	S_4
U_1	2.5sec	4.0sec	?	?
U_2	?	?	5.2sec	?
U_3	6.3sec	?	0.8sec	?
U_4	10.1sec	?	0.8sec	6.6sec

Both *user associated IT-infrastructure* (i.e., network configuration parameters like bandwidth and latency) and *WS run-time environment* (i.e., CPU and storage) have a role in measuring "similar" QoS values [26]. It has been demonstrated in [11] that WSs, which operate in a "stable" run-time environment, provide a small variation of QoS values for all users, although the WSs may have different functionalities. Contrarily, those WSs, which operate in an "unstable" run-time environment, provide different QoS values for different users. Therefore, to measure similar QoS values during recommendation, the WSs from an unstable run-time environment are deemed better than those that operate in a stable one.

In this paper, we take this observation into consideration and make the following contributions:

– We propose a CF-based QoS prediction approach for WSs recommendation that uses unsupervised machine learning (for instance, K-means) to clusters users based on their respective locations. The approach, then, narrows down the most similar neighbors in terms of similar historical QoS values with respect to the target user (i.e., the one who expects a recommendation for a certain WS) by

searching neighbors in the target user's own cluster. Finally, the WS with the most suitable QoS value is recommended to the target user.
- We add the effect of user's network configuration parameter to the recommendation analysis when identifying similar WSs between the target user and his neighbors.
- We conduct 5 extensive different experiments to verify whether location and network have an impact on the QoS values of WSs perceived by users, and that WSs with closer locations and similar networks will have more similar QoS values.
- We conduct another set of experiments (two subsets of 50 and 100 rounds) using this time a real dataset of WSs, which shows that our approach significantly outperforms some well-known WSs recommendation approaches, in terms of response time and throughput.

The rest of this paper is organized as follows. Section 2 discusses some related work. Section 3 presents the motivations of this work. Section 4 introduces the broker-based approach for recommending WSs. The broker's duties are discussed in the same section, as well. Section 5 presents how user similarity is computed and similar neighbors are selected. Section 6 discusses the proposed QoS prediction model. Section 7 reports the experiments that back our QoS prediction model. Finally, Section 8 concludes the paper and identifies some future work points.

2 Related work

CF techniques can be divided into 2 categories: model-based [7, 12, 13, 27] and memory-based [11, 31]. Model-based use statistical and machine learning models to formulate a model from a dataset upon which predictions are made. Although they are sometimes "difficult" to implement these CF approaches predict results in a short time [4, 18].

Memory-based can be further decomposed into: user-based [8, 16] and item-based [9, 20]. In user-based, a set of similar users declared as neighbors to the target user are narrowed down based on their historical information and, then, using their invocation record prediction is made for the target WSs. In item-based, a set of similar items that are neighbors to the target WSs are narrowed down based on their historical information and, then, the prediction is made for the target WS. Memory-based CF techniques are highly effective in general and can be easily implemented. However, they are often slow, less scalable [18], and do not perform well when users and services are in a large number.

Shao et al. [21] standardized the QoS values of WSs and proposed a user-based approach to predict missing values. Wu et al. [27] proposed a user-based CF approach, which applies K-Means clustering to narrow down trustworthy users for predicting the missing QoS values. The Pearson Correlation Coefficient (PCC) has been used in many CF-based recommendation approaches to measure user similarity. However, the traditional PCC may not work or fail to compute accurate similarity when the available information is little or largely sparse [10]. McLaughlin et al. [14] improved user similarity computation by adjusting the traditional PCC with weight related to common item ratings between the concerned users. Finally, Liu et al. [11]

used an enhanced PCC to incorporate the effect of location and network on predicting QoS values.

Many recent CF-based approaches have acknowledged the impact of context, such as network configuration and user geographical location, on predicting a service's QoS. It has been found that context improves prediction accuracy. For instance, Chen et al. [4] consider location in WSs' QoS prediction. They used IP addresses and QoS similarities to cluster users in their respective regions. However, their technique overlooks WSs' locations for prediction, which we do not do. Moreover, due to the dynamic nature of IP addresses, measuring the closeness of 2 users by comparing their addresses may not be accurate [11]. Chen et al. [5] improved their work in [4] by using latitude-longitude pair to cluster users into regions and then use similar regional centers for QoS predictions. Tang et al. [25] clustered users based on Autonomous System (AS)[1] and country, but do not incorporate the effect of network on QoS predictions, **which we do**. Liu et al. [11] incorporated the effect of location and network on predicting QoS values. However, their approach may not give provide results in a situation where 2 users in the same AS are located at a greater distance from each other, whereas a more suitable and geographically closer user might be available outside this AS. Yu et al. [31] used both time and location for making WSs' QoS prediction. Lo et al. [12] found similar neighbors for WSs and users by using geographical information, and then applied matrix factorization to make accurate predictions. He et al. [7] created a location-based Hierarchical Matrix Factorization (HMF) model for predicting missing QoS values. Both global and location-based local QoS matrices are utilized to train the model. Missing QoS values can be obtained by combining the results from both global matrix factorization and local matrix factorization.

Compared to the aforementioned approaches, we use unsupervised machine learning technique (Section 4) to cluster users on the basis of location to address data sparsity issue that existing CF approaches suffer from. We also consider user-network configuration parameter during similarity calculation. Finally, we propose a modified PCC to predict the missing QoS values.

3 Research motivation

This section discusses the motivations of our work in terms of impact of WS runtime environment on QoS values (Section 3.1), impact of user location parameter on QoS values (Section 3.2), and impact of user-network-configuration parameter on QoS values (Section 3.3).

3.1 Impact of WS run-time environment on quality-of-service

Existing CF-based recommendation approaches assume that commonly invoked WSs by users (target and candidate) contribute equally to the QoS similarity calculation

[1] An autonomous system is either a single network or a collection of networks within the Internet that is managed and supervised by a common network administrator (such as a business enterprise or university). Each autonomous system is assigned a globally unique ID, known as Autonomous System Number (ASN).

between 2 users. Our argument is that WSs should have different contributing weight depending on their runtime environments with focus on processing and storage capabilities. Fig. 2 illustrates how WSs with different QoS values can contribute differently when measuring the user similarity.

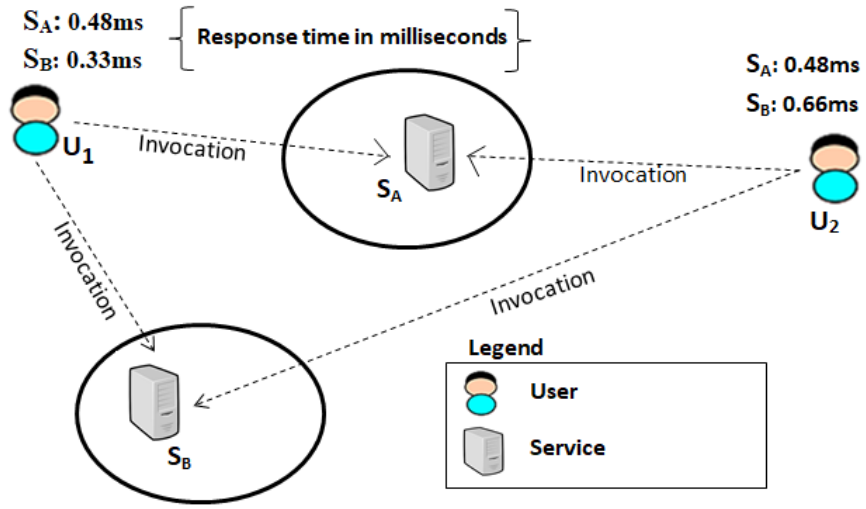

Fig. 2 Impact of network on WSs' QoS values

Suppose that a WS (S_A) runs on a stable environment. Both U_1 and U_2 may experience similar QoS with S_A due to the fact that S_A offers similar performance to them (i.e., response time of S_A is 0.43ms for both U_1 and U_2). Because of the environment stability, a small variation of QoS values is likely to be observed for U_1 and U_2. However, this does not imply that U_1 and U_2 can be considered similar when predicting QoS values of other WSs such as S_B that these 2 users could use. On the contrary, in this example, since s_B runs on an unstable environment and close to U_1, but, far from U_2, it is likely that both users will observe different QoS values for U_B.

3.2 Impact of users location on quality-of-service

The effect of users location on a WS's QoS values is important and backed up by Google Transparency Report that has given a similar observation about Google Services [5]. Although the location has been used in the past as a contextual information when recommending WSs, the main challenge is to efficiently use location to significantly improve the recommendation accuracy. Acquiring Web services' and users' locations information can be done through IP addresses[2] that are mostly known. Several services and databases are available (e.g., the Who is lookup service[3],and GeoIP

[2] http://www.iplocation.net
[3] http://www.whois.net

databases[4]) which can be used to find information on geo-location based on the provided IP address such as country, autonomous system, city and type of network, etc. The authors in [11] has also used the same resource to collect location information. In [4], Xi et al. use IP addresses to group users together. Although this seems reasonable due to the shortage of IPv4 address [11], 2 users living in separate countries may have "close looking" IP addresses (Table 2 with 2 users having *4.67.68.0* and *4.67.64.0* IP addresses, respectively). However, users could be located in different countries. This is deemed a fundamental concern for IP-based user clustering.

Table 2 User clustering based on IP address

Ip-address	AS.NO	Country
4.56.0.0	AS863	Canada
4.67.64.0	AS9996	Japan
4.67.68.0	AS863	Canada
4.68.248.0	AS1148	Netherlands
4.68.294.32	AS863	Canada
4.71.36.4	AS1148	Netherlands

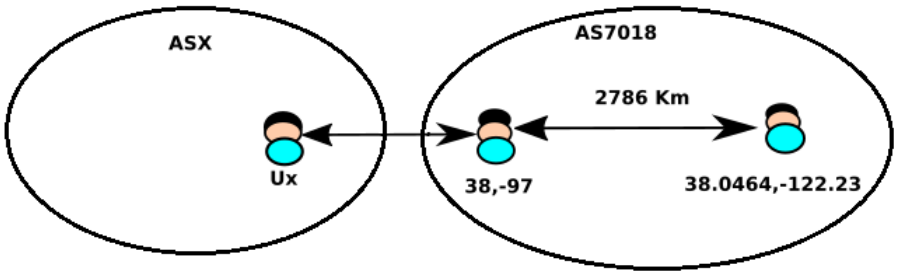

Fig. 3 User's location's in autonomous systems

To address these concerns, Liu et al. [11], Tang et al. [25], and Yu and Huang [31] use the notion of Hierarchies (H) to search for the most similar users. The innermost hierarchy contains users, who fall into the target users' same ASs. The middle hierarchy contains users, who belong to the user's country, and the outermost hierarchy contains the rest of users. The main limitation with the above approaches are:

– Most similar users will be first narrowed down based on users' own ASs. Now, there are users who are more similar to the target user in these users' respective countries, but members of the target user's own AS will be prioritized, first.
– Almost 90% of time a country wide search is initiated because ASs are not very dense and have few numbers of users [11].

[4] http://maxmind.com

– 2 users who reside in the same AS may not be physically located close to each other. For instance, in *AS7018* there are 2 users with latitude and longitude *38*,*-97* and *38.0464*,*-122.23*, respectively (Table 5). The distance between them is 2786Km. However, the basic theme discussed in [5, 11, 25, 31] is that the closest the users, the more similar they will be in term of historical QoS information. In the discussed case, this assumption is not counted true as per Fig. 3.

User location plays a key role in similarity measurement. Instead of measuring the similarity between all users, it is time saving (and computation saving too) to measure the similarity of those users, who appear to be "near" the target user (Fig. 4), where users are organized in different hierarchies based on their distance from other users. Let us assume that user (u_0) is linked to hierarchy (H_0). All users in H_0 are at a distance of $1x$ from u_0, and that this distance increases when we move further towards distant hierarchies. We would like to measure how similar other users are to u_0 based on s_A's QoS values since they all have used s_A. One option is to randomly pick users and measure their similarities regarding u_0 one by one. Eventually, we will have the subset of most similar users to u_0, but this is computationally expensive since the whole set of users is searched. To overcome this problem, we measure similarity for users residing in H_0 in which users (u_1 and u_2) are located at a distance of $1x$ from u_0. If we are not able to find the top most similar users in H_0, only then we check more distant hierarchies (H_1 and H_2 in this case).

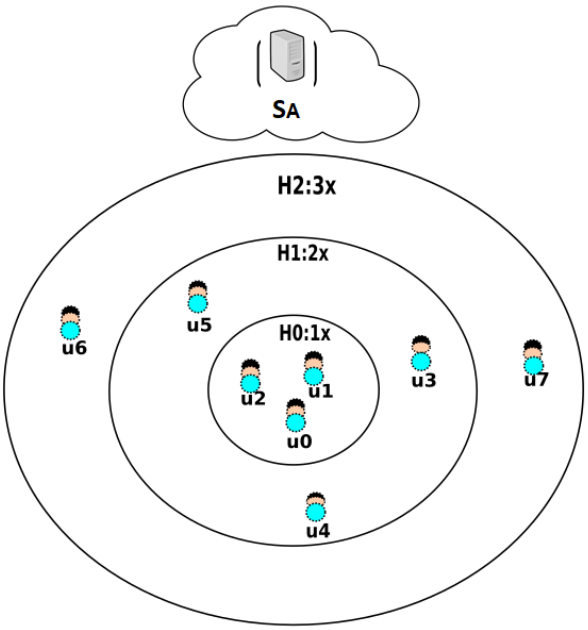

Fig. 4 User clustering according to hierarchies

3.3 Impact of user-network configuration on quality-of-service

Since users are located in different places, their experiences with using (even the same) WSs vary largely due to the technical configurations of the communication networks deployed in these places. For a better recommendation of WSs to users, we consider the impact of network configuration on this recommendation. In addition to user location parameter, we narrow down the analysis of this impact to connection type offered to users. We specialize connection type into bandwidth and latency when sending requests to services and sending responses to these requests back to users.

During experimentation, we considered 2 different types of network connections, DSL and Corporate. Each type is identified based on users' IP addresses that are included in the WSDream dataset. In addition to this dataset, we used a second one that a local telecommunication company has made available after anonymizing the content and dropping all sensitive details. This extra dataset was required because WSDream does not include details about latency and bandwidth. Thus, joining both datasets using "connection type"as a common property has led to an enriched WS-Dream dataset. Table 3 and Table 4 depict the respective properties of each dataset.

Table 3 Structure of the local telecommunication company dataset

Connectin Type*	Bandwidth(MB)	Latency(ms)
DSL	10	4
Corporate	100	1

Table 4 Structure of WSDream dataset

IP Address	Country	Connection Type*	AS	Latitude	Longitude
12.46.129.15	United States	DSL	AS7018	38.0464	-122.23
122.1.115.91	Japan	Corporate	AS4713	35.685	139.7514

Let us consider a real-world situation where 3 users (u_1, u_2, and u_3) in different locations with different connection types invoke s_1. The connection type of u_1 is Corporate with a bandwidth of 100MB and a latency 1 ms; the connection type for u_2 is Corporate with a bandwidth of 100MB and latency 1ms; whereas u_3 is a DSL user with a bandwidth of 10MB and latency 4ms. Thus, it is more likely that both u_1 and u_2 experience similar performance in terms of bandwidth and latency, and can be categorized as similar.

4 Clustering and brokerage in support of recommending Web Services

This section describes first, our approach for clustering users using physical locations and, then the components of this approach that will help predict missing QoS values.

4.1 Clustering

A main limitation of existing recommendation approaches [5, 11] that rely on location only, is that they are not able to precisely pin down users who are geographically close to the target user. To address this limitation, we use K-Means technique [6] to cluster users based on their locations in the form of Latitude-Longitude pair. All users are positioned in their respective clusters and users who belong to the same cluster are geographically considered close to each other; hence, there is a possibility that they have the same network configuration parameters (like bandwidth and latency) . K-Means groups data into different clusters in a way that the data in the same cluster is more alike than those in other clusters. The adoption of Latitude-Longitude pair along with K-Means allows us applying the Euclidean distance model to group the most closely located neighbors, which should help minimize our search space and find the most similar neighbors in term of histotrical QoS information in our own cluster.

Compared to the work of Wu et al. [27] who applied K-Means clustering to narrow down trustworthy users for QoS predictions. we use location-based K-Means to cluster users.

4.2 Brokerage

Suppose a user wants to choose the best possible candidate WS for weather forecast. She sends a request to a Service Broker (SB) that is aware of users' locations, user's network configuration parameters, WSs run-time evironments, and WSs' QoS values. Using all these details, the SB predicts the QoS values of all possible candidate WSs' with missing QoS values and recommends to the user the WS that optimizes response time and throughput. Fig. 5 presents the stakeholders and modules of our proposed recommendation approach.

There are 3 stake holders in ourapproach: WS providers, SB, and WS users [24]. The SB has agreements with providers for maintaining their properties (like service type, location, and cost) and satisfies users' needs in terms of WSs along with storing these users' locations[5]. WSs are indexed by the SB according to the similarity of their properties. Upon receiving a WS request from a user, the SB recommends the optimal WS from a set of functionally similar WSs. Details about the modules of the broker-based approach are presented below:

1. *User-Service Matrix* contains records, at a specific time, of users invoking WSs. In our approach, since we are interested in the effect of location and network on QoS of WSs, hence, the matrix will hold response time and throughput values.
2. *User clustering module* groups users based on their respective locations. When a user sends this module a QoS request, it identifies which cluster the user belongs to and then, sends this user's id and other users' ids in the cluster to the *similarity measurement module*.

[5] Privacy concern is out-of this paper's scope.

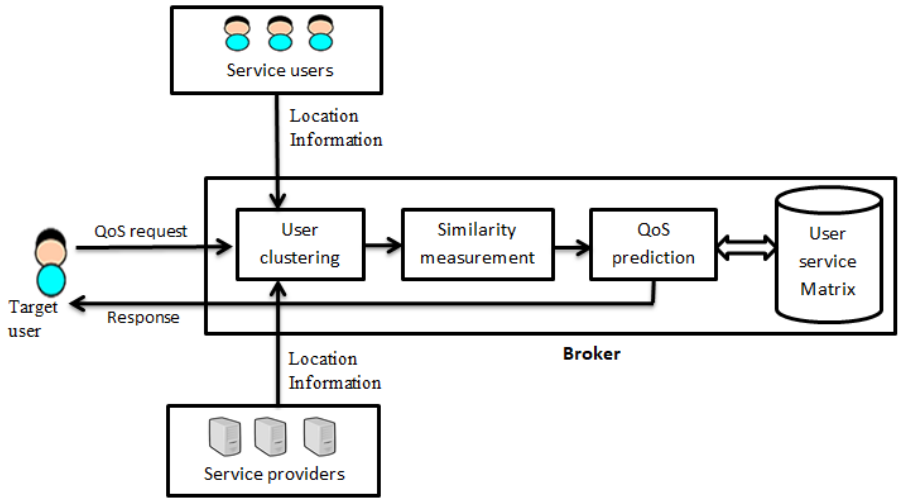

Fig. 5 A broker-based architecture for Web services recommendation

3. *Similarity measurement module* assesses how similar other users are to the target user. Similarity is measured using historical information WSs of the common WSs, which is available in the user-service matrix. Afterwards, the top most similar users are identified and their ids are forwarded to the *QoS prediction module*.
4. *QoS prediction module* estimates the QoS values for the WSs that the user has selected as possible candidates by using the historical information of most similar users. The results will be returned to the user.

5 Computing user similarity and selecting similar neighbors

In this section, we formally describe how user similarity is computed along with how similar neighbors are selected. To this end, we adjust PCC to measure similarity.

5.1 Notations and definitions

- $U=\{u_1,u_2,..,u_n\}$ is a set of users, where u_k $(1 \leq \mathrm{k} \leq \mathrm{n})$ is the k^{th} user and n is the total number of users.
- $S=\{s_1,s_2,...,s_m\}$ is a set of WSs, where s_k $(1 \leq \mathrm{k} \leq \mathrm{m})$ is the k^{th} WS and m is the total number of Web services.
- U_s is a group of users who have invoked a certain *WS* referred to as s.
- $M=\{m(u,s) \text{ — } u \in \mathrm{U},\ s \in S\}$ is a user-service matrix, where $m(u,s)$ is the invocation record (response time or/and throughput) value of service s for user u. $m(u,s)$=NULL means u has not invoked s, yet.
- $C=\{c_1,c_2,\cdots,c_p\}$ is a set of clusters, where c_k $(1 \leq \mathrm{k} \leq \mathrm{p})$ is a cluster and p is the total number of clusters. u_{c_p} represents the cluster p to which user u belongs

- B=$\{bu_1,bu_2,..,bu_n\}$ is a set of bandwidths that are available to users, where bu_k $(1 \leq k \leq n)$ is the k^{th} user's bandwith and n is the total number of users.
- L=$\{lu_1,lu_2,..,lu_n\}$ is a set of latencies which are expereinced by users, where lu_k $(1 \leq k \leq n)$ is the k^{th} user's latency and n is the total number of users.

5.2 Similarity computation

To measure user similarity, PCC is widely used in recommendation systems [4, 5, 11] (Equation 1).

$$PCC(u_t,u_c) = \frac{\Sigma_{s\varepsilon S_{u_t}\cap S_{u_c}}(m(u_t,s)-avg(u_t|S)).(m(u_c,s)-avg(u_c|S))}{\sqrt{\Sigma_{s\varepsilon S_{u_t}\cap S_{u_c}}(m(u_t,s)-avg(u_t|S))^2}.\sqrt{\Sigma_{s\varepsilon S_{u_t}\cap S_{u_c}}(m(u_c,s)-avg(u_c|S))^2}} \quad (1)$$

where

- $S_{u_t} \cap S_{u_c}$ represents the common WSs that invoked by both target user u_t and candidate user u_c;
- $m(u_t,s)$ represents the QoS value of s when invoked by u_t;
- And, $avg(u_t|S)$ is the average QoS values for all WSs invoked by u_t;

5.2.1 Impact of WSs run-time environment on similarity computation

Though Equation 1 gives the similarity between 2 users, it fails to incorporate the effect of WS runtime environment. As mentioned in Section 3, some WSs have better QoS values than others because of the computing environment upon which they run. To address this limitation, Liu et al. [11] introduce a deviation-based weight (dev) that is calculated as follows.

First, the QoS values $m(u_t,s)$ between 2 real numbers 0 and 1 are normalized. Here, 2 cases need to be considered: when a large QoS value indicates better QoS (e.g., when throughput is large it is considered better), Equation 2 is used; contrarily, Equation 3 is used when a small QoS value indicates a better QoS (e.g., when response time is small it is considered better).

$$norm(u,s) = \frac{m(u,s)-min_s}{max_s-min_s} \quad (2)$$

$$norm(u,s) = \frac{max_s-m(u,s)}{max_s-min_s} \quad (3)$$

Here max_s and min_s represent the maximum and minimum values of s, respectively. When max_s=min_s, $norm(u,s)$ is 1.

Equation 4 computes the standard deviation for s. Here $avg(s)$ is the s's average QoS value for all users who have invoked it and $|U_s|$ is the total number of users who invoked s.

$$dev_s = \frac{\sqrt{\Sigma_{u \varepsilon U_s}(norm(u,s) - avg(s))^2}}{|U_s|} \tag{4}$$

It should be noted that the higher the deviation, the more unstable the WS is.

5.2.2 Impact of user-network-configuration parameters on similarity computation

In addition to the effect of deviation caused by WSs runtime environment, we also add the effect of user network configuration parameters (i.e., bandwidth and latency). Those users who share the same connection type, will experience the "same" bandwidth and latency and thus, will be deemed more similar. Equation 5 and Equation 6 are used to incorporate the effect of difference of bandwidth and latency experienced by 2 users u_t and u_c.

$$DiffBW_{u_t u_c} = \frac{1}{|b_{u_t} - b_{u_c}|} \tag{5}$$

$$DiffL_{u_t u_c} = \frac{1}{|l_{u_t} - l_{u_c}|} \tag{6}$$

It should be noted here that if 2 users have different connection types the difference of bandwidth and latency between them will be greater than 0 and this will have an inverse effect on the similarity index. However, if they share the same connection type, $DiffBW_{u_t u_c}$ will be 1 and no effect will be added to the similarity measurement.

5.2.3 Impact of user location on similarity computation

Another limitation with the traditional PCC is that it does not incorporate the effect of location. Those users who are closely located to the target user should be considered first compared to those who are far away. To achieve this, we introduce a location-sensitive weight that calculates the distance of the target user to the candidate user for whom we want to measure the similarity relative to the common WS. Following are the steps to calculate this distance:

– First, using Equations 7:11 we calculate the relative distance between u_t and u_c with respect to the WS using Haversine distance [23].

$$DiffLat_{u_t s} = Lat_{u_t} - Lat_s \tag{7}$$

$$DiffLon_{u_t s} = Lon_{u_t} - Lon_s \tag{8}$$

$$a = (\sin(DiffLat_{u_t s}/2)^2 + \cos(Lat_{u_t}) * \cos(Lat_s) * (\sin(DiffLon_{u_t s})^2)) \tag{9}$$

$$c_{u_t s} = 2 * a\tan 2(\sqrt{a}, \sqrt{1-a}) \tag{10}$$

$$d_{u_t s} = R * c_{u_t s} \tag{11}$$

Here $d_{u_t s}$ is the distance between u_t and s. $c_{u_t s}$ is the intermediate great circle distance. R is the radius and its value is 6467km. Similarly the distance between u_c and s is represented by $d_{u_c s}$ and is given by Equation 12.

$$d_{u_c s} = R * c_{u_c s} \tag{12}$$

– The difference between $d_{u_t s}$ and $d_{u_c s}$ reveals how far the target user and candidate user are to each other, with respect to s (Equation 13). Here it should be noted that the smaller the distance, the closest the users are.

$$dist_{u_t u_c} = d_{u_t s} - d_{u_c s} \tag{13}$$

Equation 4, Equation 5, Equation 6 and Equation 13 return the deviation caused by Wss run-time environments, difference of bandwidth, difference of latency and, distance between the target user and candidate user, respectively. We incorporate these factors in the traditional PCC (Equation 1). Similarity will be measured using a modified Pearson Co-relation Coefficient (Equation 14).

$$mPCC(u_t, u_c) = DiffBW_{u_t u_c}.DiffL_{u_t u_c} \frac{\Sigma_{s \epsilon S_{u_t} \cap S_{u_c}} \frac{dev}{dist_{u_t u_c}} (m(u_t,s) - avg(u_t|S)).(m(u_c,s) - avg(u_c|S))}{\sqrt{\Sigma_{s \epsilon S_{u_t} \cap S_{u_c}} (\frac{dev}{dist_{u_t u_c}} m(u_t,s) - avg(u_t|S))^2} . \sqrt{\Sigma_{s \epsilon S_{u_t} \cap S_{u_c}} \frac{dev}{dist_{u_t u_c}} (m(u_c,s) - avg(u_c|S))^2}} \tag{14}$$

5.3 Selection of similar neighbors

As mentioned in Section 5.2, we faced 3 challenges with similarity calculation: WSs do not contribute equally towards this calculation, users who are located closely to each other and have same network configuration parameters have a higher chance of being more similar. We address these challenges with Equation 14. Another, important challenge that we tackle is how to reduce the search space, which means instead of searching the whole user set for similar neighbors, we could reduce this set by checking similarity of specific users. To tackle this challenge, we divide users into clusters. Each cluster contains users who are closely located to each other. The following are the steps for narrowing down the search space referred to as Top-X neighbors.

– Calculate the similarity between users present in the target users's own cluster using Equation 14.
– Create a set of top most similar users from the target user's own cluster and denote this set by Top-X neighbors.
– Information related to most similar users is forwarded to the *QoS prediction module* Fig.5 so, that, it estimates the QoS values for the candidate WSs.

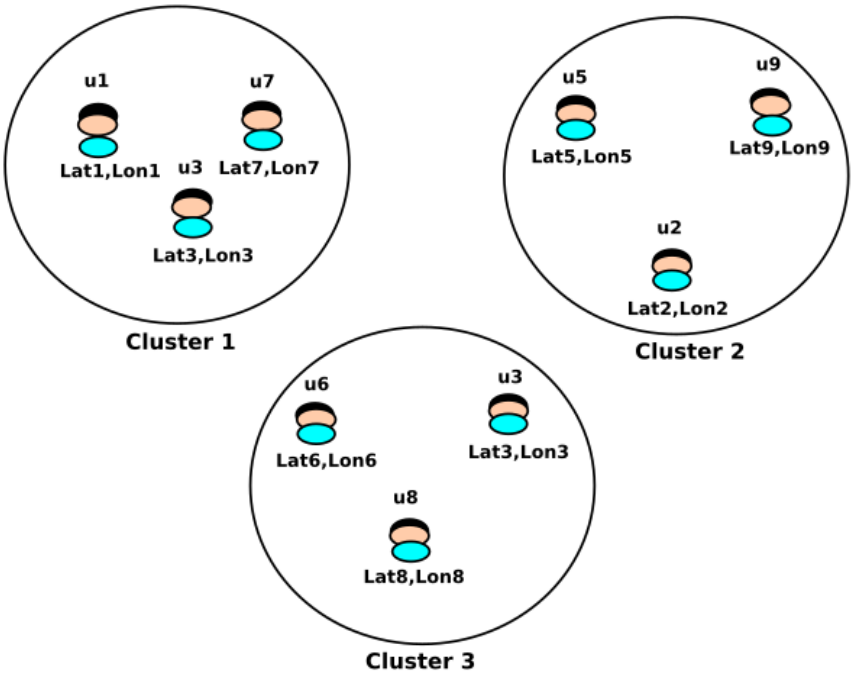

Fig. 6 Clustering using latitude and longitude parameters

6 Quality-of-service prediction for recommending Web Services

This section describes the QoS prediction approach so, that, the most suitable WS is recommended to the user.

6.1 Quality-of-service prediction

CF-based approaches usually adopt Equation 15 to predict QoS values.

$$n(u_t,s) = avg(u_t) + \frac{\Sigma_{u_x \varepsilon Top-X} Sim(u_t,u_x).((n(u_x,s) - avg(u_x))}{\Sigma_{u_x \varepsilon Top-X} Sim(u_t,u_x)} \quad (15)$$

Where $n(u_t,s)$ represents the predicted value of s for u_t, Top-X neighbors represents the top most similar users to u_t. u_x is a user who belongs to Top-X neighbors, $Sim(u_t,u_x)$ is the similarity factor between u_t and u_x, and, $n(u_x,s)$ represents the QoS value of the target service s observed by u_x.

However, Equation 15 rarely provides accurate results because WSs' QoS factors such as response time are objective and this equation is for data that are subjective in nature [11, 13]. Due to the objective data, there can be a large variations in WSs' QoS ranges. This variation will result in different averages. Ma et al. [13] label this variation as GAP and if the GAP between 2 users is large than $avg(u_t)$ and $avg(u_x)$

would be different and this will affect the overall prediction accuracy. To counter this issue, we checked out the work of [30] and made use of Equation 16 to predict missing QoS values.

$$n(u_t, s) = \frac{\Sigma_{u_x \varepsilon Top-X} Sim(u_t, u_x).(n(u_x, s)}{\Sigma_{u_x \varepsilon Top-X} Sim(u_t, u_x)} \tag{16}$$

6.2 Web services recommendation

Suppose a user u_t who wants to find a WS with the most optimal response time and throughput values from a set of functionally similar WSs. Our recommender system will first, find which cluster u_t belongs to, then the most similar neighbors to this user will be narrowed down using Equation 14. Then, using Top-X neighbors, similar neighbors are generated for each candidate WS using Equation 16. Finally, the WS with the most optimal QoS value is proposed to the user.

7 Evaluation and experimentats

This section describes the evaluation that took place along with discussing the experiment results.

7.1 Evaluation

We performed various experiments to evaluate our WSs recommendation approach. Specifically, we raised the following questions: (*i*) does location impact similarity between users, (*ii*) how accurate are our recommendations compared to other location based CF approaches, and (*iii*) what is the effect of factor x (most similar neighbors) on predicting accuracy. We developed several in-house Python programs for the experiments on an HP notebook with the following specifications: Intel(R) Core (TM) i5-4210U @1.70 Ghz, 6GB RAM with Ubuntu 16.04 as an operating system. For clustering, we set *K (no of clusters)* to 3 and this is justified by the elbow plot in Fig. 7. For both K-means clustering and elbow plot, we have used latitude, longitude coordinates as features (Table 5).

The experiments were carried out using a real dataset of WSs, WSDream dataset 2[6]. The dataset contains response-time and throughput values of 5825 WSs invoked ~~used~~ at a certain time by 339 users located in 31 countries. Each matrix contains 1974675 invocation records. We used response-time matrix and throughput matrix for the experiments. Besides these 2 parameters, information related to users like IP-addresses, countries, IP numbers, ASs, Latitude-Longitude coordinates were included in the dataset (Table 5).

[6] available at wsdream.github.io

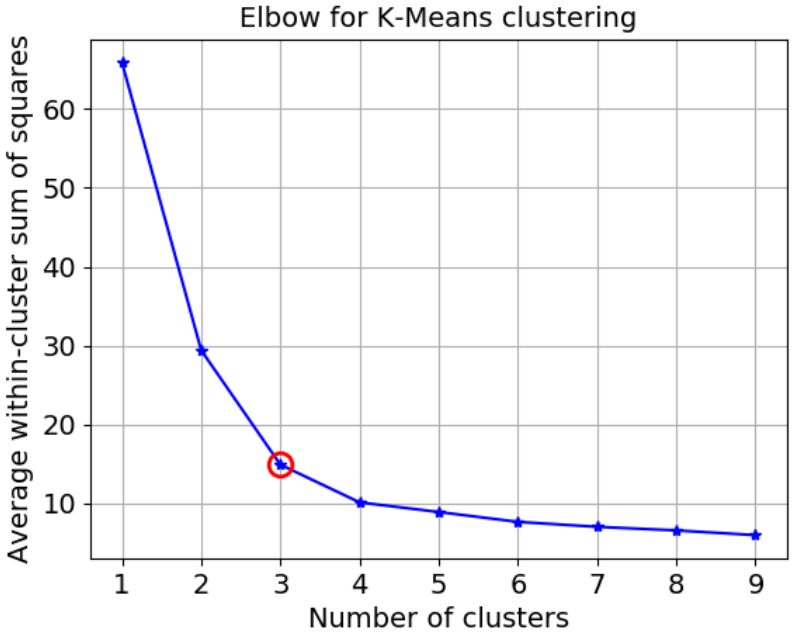

Fig. 7 An elbow plot showing the precise value of K to be used

Table 5 Details collected about users from WSDream dataset 2

User-ID	IP Address	Country	Ip No	AS	Latitude	Longitude
0	12.108.127.138	United States	208437130	AS7018	38	-97
1	12.46.129.15	United States	204374287	AS7018	38.0464	-122.23
2	122.1.115.91	Japan	2148143924	AS4713	35.685	139.7514
3	128.10.19.52	United States	2148143925	AS17	40.4249	-86.9162
4	128.10.19.53	United States	2154771517	AS17	40.4249	-86.9162

7.2 Relation between QoS similarity and location proximity

To calculate the similarity between users, we used mPCC discussed in Section 5.2. To prove that a strong relation exists between location proximity and QoS similarity of users, we developed the following experiment:

- We selected 5 different configurations of parameters x and performed experiments on 20 random users for each configuration on both response time and throughput matrices. We then calculated the proportion of most similar neighbors which were found in the users' own clusters for each configuration. A higher proportion depicts a strong correlation between location proximity and QoS similarity.

Table 6 describes the results of the relation between location closeness and QoS similarity. From Table 6 we can say that there is a strong relation between location proximity and QoS similarity between users. For example, for x=*10* we were able

Table 6 Relation between location closeness and QoS similarity

x(ResponseTime)	Success Rate	x(Throughput)	Success Rate
2	100%	2	100%
4	100%	4	100%
6	100%	6	100%
8	100%	8	100%
10	100%	10	100%

to narrow down top 10 similar users with a success rate of 100% for response time matrix and 100% for throughput matrix.

7.3 Prediction accuracy evaluation

Mean Absolute Error (MAE, Equation 17) and Normalized Mean Absolute Error (NMAE, Equation 18) are widely used to determine how accurate a CF approach is. A smaller value of MAE and NMAE represents an excellent accuracy.

$$MAE = \frac{\Sigma_{s \varepsilon S} |N(u_t, s) - n(u_t, s)|}{N} \tag{17}$$

Here $N(u_t, s)$ represents the actual QoS value of s, $n(u_t, s)$ represents the predicted value of s, and N is the total number of services. $\Sigma_{s \varepsilon s}$ represents all services for which we have made predictions.

$$NMAE = \frac{N.MAE}{\Sigma_{s \varepsilon S} N(u_t, s)} \tag{18}$$

The techniques used for benchmarking are **UMEAN** (it takes the mean of all WSs that a user has invoked) [32], **IMEAN** (it takes the mean of all invoctions made by users on that WS) [32], **IPCC** (it narrows down similar WSs using PCC and than predicts QoS value using information of those services) [32], **UPCC** (it narrows down similar users using PCC, the matrix density used was 30% and than predicts QoS value using information of those users) [32], **RegionKNN** [4] (it divided users into regions based on their IP addresses and than make predictions using information of those users who are in the same region), and, finally, **ULACF** [11] (it divided users into regions based on their ASs and countries and than make predictions using information of those users who are in the same ASs and countries). All benchmarks were implemented in Python and matrix density was kept 30%.

To evaluate our approach, we divided the 339 users into 2 sets: test users and active users. During Round 1, 50 test users were selected and during Round 2, 100 test users were selected randomly and a random WS was selected for prediction. To demonstrate that our approach works with a limited number of QoS values we reduced the matrix density to 30%. The following parameters were used for the experiment x=10 and K=3. Authentic estimation was generated by using MAE (Equation 17) and NMAE (Equation 18) on the results of every test user. Prediction accuracies of various approaches are reported in Table 7 and Table 8. L.S.R.S represents the proposed **L**ocation **S**ensitive **R**ecommendation **S**ystem. MAE and NMAE performance

related to response time matrix is represented by RTMAE and RTNMAE, whereas performance related to throughput matrix is represented by TPMAE and TPNMAE, respectively. From Tables 7 and 8, we clearly see that L.S.R.S performs much better than other approaches [4, 11].

Table 7 Prediction comparison on the basis of response time

	Rounds=50		Rounds=100		Average	
	RTMAE	**RTNMAE**	**RTMAE**	**RTNMAE**	**RTMAE**	**RTNMAE**
RegionKNN	0.5074	0.5584	0.5054	0.5562	0.5064	0.5573
ULACF	0.3178	0.3102	0.6224	0.4568	0.4701	0.3885
UMEAN	0.8741	0.9619	0.8761	0.9642	0.8751	0.9630
IMEAN	0.6792	0.7474	0.6784	0.7466	0.6788	0.7470
UPCC	0.4521	0.4976	0.4565	0.5024	0.4543	0.4995
IPCC	0.4185	0.4605	0.4160	0.4578	0.4175	0.4591
L.S.R.S	**0.2899**	**0.3738**	**0.3665**	**0.3890**	**0.3282**	**0.3801**

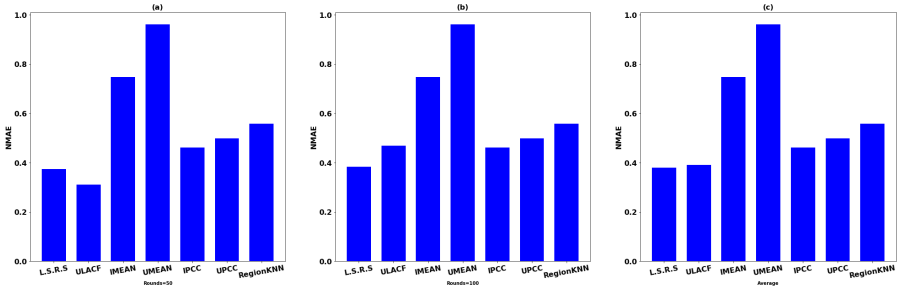

Fig. 8 Performance evaluation based on NMAE (response time)

Table 8 Prediction Comparison on the basis of throughput

	Rounds=50		Rounds=100		Average	
	TPMAE	**TPNMAE**	**TPMAE**	**TPNMAE**	**TPMAE**	**TPNMAE**
RegionKNN	23.8585	0.5020	23.7609	0.5007	23.8097	0.5013
ULACF	10.6188	0.2305	14.4024	0.4760	12.5106	0.3532
UMEAN	53.8052	1.1321	53.8725	1.1351	53.8388	1.1336
IMEAN	26.5440	0.5585	26.5374	0.5592	26.5407	0.5588
UPCC	17.6020	0.3704	17.4354	0.3674	17.5187	0.3689
IPCC	21.5995	0.4545	21.4315	0.4516	21.5115	0.4530
L.S.R.S	**5.041**	**0.2298**	**10.7697**	**0.3901**	**7.9005**	**0.3095**

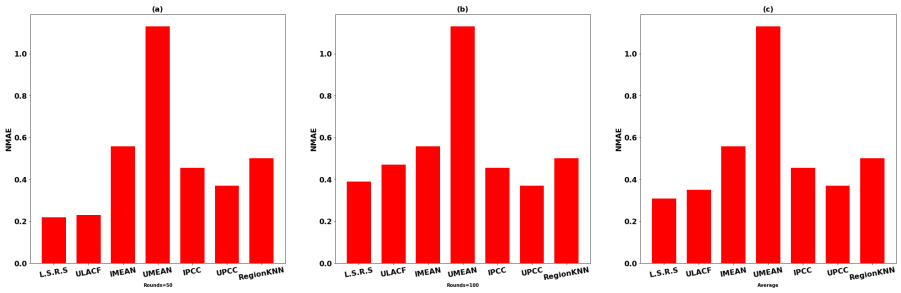

Fig. 9 Performance evaluation based on NMAE (throughput)

7.4 Effect of x on prediction accuracy

To understand the effect of x on prediction accuracy we performed experiments on both response time and throughput matrices. The results of the experiments are shown in Fig 10. For the response-time matrix, we observe that the prediction accuracy keeps decreasing as we increase x. For the throughput matrix, the prediction accuracy first decreases, then becomes stable. From these results we conclude that there is no need to increase the number of most similar users in order to obtain accurate predictions. Indeed, the reason for this is that when x is increased more distant neighbors are selected that effect the prediction accuracy.

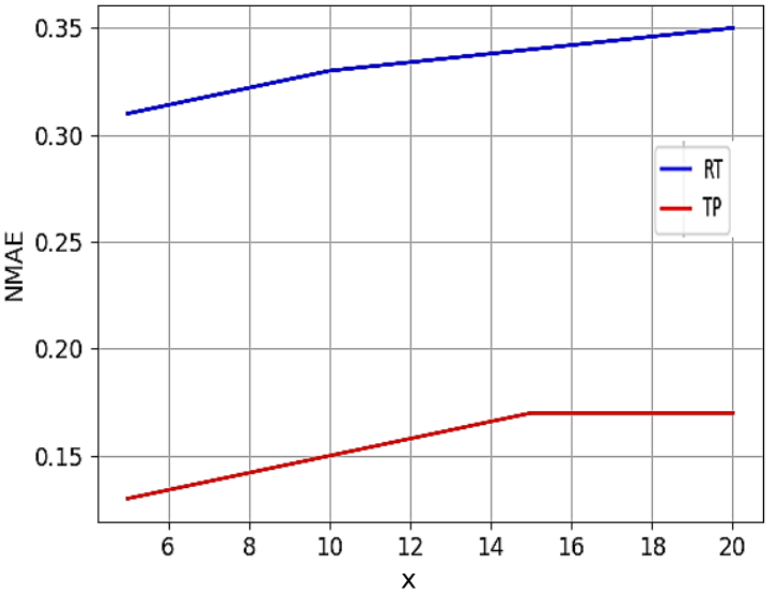

Fig. 10 Effect of x on prediction accuracy

8 Conclusion and Future Work

We presented a novel broker-based approach for Web services recommendation that is sensitive to users' location, aware of WSs runtime environment and aware of users' network configuration parameter. Different from existing approaches, we cluster users based on location and made use of the network connecting WSs are deployed. We ran several experiments on the response-time and throughput using in-house Python-based programs and real dataset of 5825 WSs (an enriched WSDream dataset 2). Those experiments demonstrated that our approach is more accurate compared to other state-of-the-art recommendation approaches. As future work, we plan to create an "item-based approach" using the same clustering technique. Also, we plan to add time as contextual element to further improve our prediction accuracy.

References

1. Al-Masri E, Mahmoud QH, Investigating web services on the world wide web, Proceedings of the 17th international conference on World Wide Web, ACM, pp 795–804 (2008)
2. Asim M, Llewellyn-Jones D, Lempereur B, Zhou B, Shi Q, Merabti M, Event driven monitoring of composite services, International Conference on Social Computing, IEEE, pp 550–557 (2013)
3. Baker T, Asim M, Tawfik H, Aldawsari B, Buyya R, An energy-aware service composition algorithm for multiple cloud-based iot applications, Journal of Network and Computer Applications, Elseiver, vol 89, pp 96–108 (2017)
4. Chen X, Liu X, Huang Z, Sun H, Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation, IEEE International Conference on Web Services (ICWS), IEEE, pp 9–16 (2010)
5. Chen X, Zheng Z, Yu Q, Lyu MR, Web service recommendation via exploiting location and qos information, IEEE Transactions on Parallel and Distributed Systems, IEEE, vol 25, pp-1913–1924 (2014)
6. Hartigan JA, Wong MA, Algorithm as 136: A k-means clustering algorithm, Journal of the Royal Statistical Society Series C (Applied Statistics), JSTOR, vol 28, pp-100–108 (1979)
7. He P, Zhu J, Zheng Z, Xu J, Lyu MR, Location-based hierarchical matrix factorization for web service recommendation, IEEE International Conference on Web Services (ICWS), IEEE, pp-297–304 (2014)
8. Jin R, Chai JY, Si L, An automatic weighting scheme for collaborative filtering, Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp 337–344 (2004)
9. Linden G, Smith B, York J, Amazon. com recommendations: Item-to-item collaborative filtering, IEEE Internet computing, IEEE, vol 7 ,pp 76–80 (2003)
10. Liu H, Hu Z, Mian A, Tian H, Zhu X,A new user similarity model to improve the accuracy of collaborative filtering, Knowledge-Based Systems, Elseiver, vol 56 , pp 156–166 (2014)

11. Liu J, Tang M, Zheng Z, Liu XF, Lyu S, Location-aware and personalized collaborative filtering for web service recommendation, IEEE Transactions on Services Computing, IEEE, vol 9 , pp 686–699 (2016)
12. LoW, Yin J, Deng S, Li Y,Wu Z,Collaborative web service qos prediction with location-based regularization, IEEE International Conference on Web Services (ICWS), IEEE, pp 464–471 (2012)
13. Ma Y, Wang S, Hung PC, Hsu CH, Sun Q, Yang F, A highly accurate prediction algorithm for unknown web service qos values, IEEE Transactions on Services Computing, IEEE, vol 9 ,pp 511–523 (2016)
14. McLaughlin MR, Herlocker JL, A collaborative filtering algorithm and evaluation metric that accurately model the user experience,Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp 329–336 (2004)
15. Menasce DA, Qos issues in web services, IEEE internet computing, IEEE, vol 6, pp 72-75 (2002)
16. Miller BN, Albert I, Lam SK, Konstan JA, Riedl J,Movielens unplugged: experiences with an occasionally connected recommender system, Proceedings of the 8th international conference on Intelligent user interfaces, ACM, pp 263-266 (2003)
17. Patil SR, Mapari VP,Web service recommendation using location-aware and personalized collaborative filtering,International Journal of Engineering Science , vol 3675 (2016)
18. Patil SR, Mapari VP, DYPCOE A, A Survey On: Web Service Recommendation Using Location-Aware and Personalized Collaborative Filtering, International Journal of Engineering Science , vol 3625 (2016)
19. Rich E, User modeling via stereotypes, Cognitive science, vol 3, pp 329–354 (1979)
20. Sarwar B, Karypis G, Konstan J, Riedl J, Item-based collaborative filtering recommendation algorithms, Proceedings of the 10th international conference on World Wide Web, ACM, pp 285–295 (2001)
21. Shao L, Zhang J,Wei Y, Zhao J, Xie B, Mei H, Personalized qos prediction forweb services via collaborative filtering, IEEE International Conference on Web Services (ICWS), IEEE, pp 439–446 (2007)
22. Shimodaira H, Similarity and recommender systems, School of Informatics, The University of Eidenburgh, vol 21 (2014)
23. Sinnott R, Computing under the open sky, Sky and Telescope, vol 68 (1984)
24. Sundareswaran S, Squicciarini A, Lin D, A brokerage-based approach for cloud service selection, IEEE 5th International Conference on Cloud Computing, IEEE, pp 558–565 (2012)
25. Tang M, Jiang Y, Liu J, Liu X,Location-aware collaborative filtering for qos-based service recommendation, IEEE International Conference on Web Services (ICWS) , IEEE, pp 202–209 (2012)
26. Tang M, Zhang T, Liu J, Chen J,Cloud service qos prediction via exploiting collaborative filtering and location-based data smoothing, Concurrency and Computation: Practice and Experience, Wiley Online Library, vol 27, pp 5826–5839 (2015)
27. Wu C, Qiu W, Zheng Z, Wang X, Yang X, Qos prediction of web services based on two-phase k-means clustering, IEEE International Conference on Web Services

(ICWS), pp 161–168 (2015)
28. Wu J, Chen L, Feng Y, Zheng Z, Zhou MC, Wu Z, Predicting quality of service for selection by neighborhood-based collaborative filtering, IEEE transactions on Systems, Man, and Cybernetics: Systems, IEEE ,vol 43, pp 428–439 (2013)
29. Xu Y, Yin J, Lo W, Wu Z, Personalized location-aware qos prediction for web services using probabilistic matrix factorization,WISE,pp 229–242 (2013)
30. Yin Y, Yu F, Xu Y, Yu L, Mu J, Network location-aware service recommendation with random walk in cyber-physical systems,SENSORS, Multidisciplinary Digital Publishing Institute, vol 9,pp 2059 (2017)
31. Yu C, Huang L,A web service qos prediction approach based on time-and location-aware collaborative filtering, Service Oriented Computing and Applications, Springer, vol 10,pp 135-149 (2016)
32. Zheng Z, Ma H, Lyu MR, King I, Qos-aware web service recommendation by collaborative filtering,IEEE Transactions on services computing,IEEE,vol 4,pp 142-150 (2011)