




S-RAP: relevance-aware QoS prediction in web-services and user contexts

Hafiz Syed Muhammad Muslim^{1,2} · Saddaf Rubab^{1,3}  · Malik M. Khan¹ · Naima Iltaf¹ · Ali Kashif Bashir^{4,5} · Kashif Javed¹

Received: 16 December 2021 / Revised: 30 May 2022 / Accepted: 3 June 2022 /

Published online: 27 June 2022

© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract

With quick advancement in web technology, web-services offered on internet are growing quickly, making it challenging for users to choose a web-service fit to their needs. Recommender systems save users the hassle of going through a range of products by product recommendations through analytical techniques on historical data of user experiences of the available items/products. Research efforts provide several methods for web-service recommendation in which QoS-related attributes play primary role such as response-time, throughput, security, privacy and web-service-delivery. Derivable attributes including, user-trustworthiness and web-services reputation in contexts of users and web-services can also affect the QoS prediction. The proposed research focuses on a web-service recommendation model, S-RAP, for QoS prediction based on derivable attributes to predict QoS of a web-service that a user who has not invoked it before would experience. Services-Relevance attribute is proposed in this publication, which emphasizes on employing the historical data and extracting the degree of relevance in the users and web-services context to predict the QoS values for a user. The proposed system produces satisfactorily accurate rating predictions in the experiments evaluated by the Mean Absolute Error and Normalized Mean Absolute Error metrics. The results compared with state-of-the-art models show a relative improvement by 4.0%.

Keywords Recommender systems · Web-services · QoS prediction · Collaborative filtering · Machine learning

✉ Saddaf Rubab
saddaf@mcs.edu.pk

¹ National University of Science and Technology (NUST), Islamabad 44000, Pakistan

² Department of Software Engineering and Computer Science, Faculty of Computing, Riphah International University, Islamabad, Pakistan

³ Department of Computer Engineering, College of Computing and Informatics, University of Sharjah, Sharjah 27272, UAE

⁴ Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, UK

⁵ School of Electrical Engineering and Computer Science, National University of Science and Technology (NUST), Islamabad 44000, Pakistan

1 Introduction

The recommender systems make it easier for users to choose any new items and offered services by anticipating the rating value a user would assign to an item. Such systems are widely used in practical applications today, and the most common uses are recommendations of movies, research articles, television series, books, hotel reservations, and more. These systems bridge the gap between users and online applications and make their decision-making process easier by providing a catalog of recommended items/objects that can be of liking to a user [18]. Recommender systems that use a collaborative filtering approach consider user data to predict ratings and make appropriate recommendations. These historical data are utilized to calculate the relation between entities, such as similarity between users and items, and using this information to predict the missing values in the sample space. This has applications in a diversity of fields, such as consumer product categories, movies, shows, web-services, software applications, books, and many others. We focus on the use of recommender systems in the recommendation of the web-services to the end users.

A web-service is a software component that is platform independent and supports interoperable communication between machines/computers/devices over a network [8, 31]. This technology has developed rapidly over the past few decades, enabling vendors and web-service providers to customize various profitable infrastructures and business opportunities in the e-industry [2]. The aim is to provide a web-service with a higher added value than individual web-services which leads to the growth of more web-service-oriented applications [9].

Quality of Service (QoS) attributes of web-service are cost and execution time, availability, reputation, success rate, security, privacy, and frequency of use [17]. For a user, the execution cost might refer to the expenses that they have to bear to avail a web-service, and in the context of the web-service provider, it refers to the cost incurred to the provider in hosting the web-service over a network [21]. The availability is considered in a user's perspective that how often and where is a web-service available. Since web-services are readily available through a network, the inverse availability, or unavailability, is more noteworthy for a user [15]. The availability could be bound by some factors such as availability in some regions and unavailability in others. The successful execution rate talks about the data of the web-service successful and failed executions, how often the web-service provided invalid, or useless facilitation to the user. The reputation refers to a web-service being highly liked by a group of users [17]. The usage frequency is about how often a web-service is invoked. A highly reputed web-service would have a high usage frequency and a frequently invoked web-service would likely have a high reputation. These attributes are what steer the overall experience of the user and eventually become the reasons of a service's success or failure [3]. The aim of a recommender system is to recommend a service to a user which fulfills the user-requirements of the above-mentioned quality attributes.

The web-services have become crowded, and multiple functionally equivalent web-services are present in the market [28]. In such a scenario, it is only natural of the users to avail the best web-service for their needs. Such needs differ against each user individually such as requirements of quality, availability, affordability, frequency of usage. A web-service that is liked by a user in the long term could be disliked by another user who does not need a long-term use of this web-service, rather just a one-time use [11]. Similarly, in perspective of web-services, feedback of a regular user would be more accurate as compared to the feedback of a user who has only used a web-service once or twice [14]. This is the problem of data-scarcity that this domain faces in the historical QoS data. It addresses that it is impractical for the users to provide QoS information by rating all the available web-services because

web-service invocation in a real-world scenario requires a lot of time and resources. Assessing some QoS characteristics such as reliability and reputation is very difficult because they require long-term observation and a series of calls. The increasing presence and acceptance of web-services on the internet not only demands security and privacy, but also requires effective selection and recommendation with which users of a web-service can be recommended the optimal web-services out of a large number of the available web-services. There is a need for a way to measure the quality of everyday products and a system to recommend the right products to a user based on user needs [1]. Since web-services can be used multiple times simultaneously, it is difficult to evaluate the quality of web-services provided remotely over the network due to the uncertainty of distance, network quality and other non-functional characteristics [32]. Therefore, this matter needs resolution in the form of recommender systems that would enable the users to effectively choose a web-service that meets their requirements and affordability [7]. These systems aim to eliminate the need for users to go through multiple products by statistically recommending to them the most suitable products through historical user data [1].

Recommendation technology can offer relevant things to a user based on their interests and preferences, and also optimizes the cost and expenditure of time in using web-services in a definitive situation [30]. Such technology often discusses the use of latent correlation factors in the prediction such as similarity, trust, reputation, and reliability [20, 23, 26]. This research work offers a methodology that utilizes the historical data in figuring the latent correlations of the web-services and users, which are then used in the process of predicting unexperienced QoS values. Comprehensive experiments have been performed to check the reliability and accuracy of the proposed technique using two metrics, Normalized Mean Absolute Error and Normalized Root Mean Squared Error. The experimentation results have been compared with the state-of-the-art approaches, and S-RAP has shown noticeable improvement in prediction accuracy. The main contributions of the work are (A) focusing on use of information in the products perspective, web-services in this scenario, (B) defining a latent correlation metric, namely web-services relevance which can be considered a derivative of item-item similarity, and (C) suggestion of using the same method coupled with higher dimensional data attributes while performing the extraction of latent correlation.

The organization of the rest of the paper is as follows: Sect. 2 discusses the research directions and efforts available in the latest research conducted. Section 3 discusses the proposed methodology with detailed working and factors involved. Section 4 explains the experimentation and results obtained from the proposed model. Section 5 concludes the research and suggests future directions.

2 Literature review

A recommender system based on collaborative filtering produces a list comprising item recommendations for users by calculating users rating for an item. This approach focuses on the relationship between user and item. In this mechanism, typically a user-item matrix is an input to the prediction system along with data of the active user. The system works out to calculate which other users are similar to the current user and then recommends items that are also liked by those similar users [19]. In other words, you can say that it recommends articles by searching for like-minded people. If a user likes a particular post, a similar user will like that post too. Facebook is an ideal example of collaborative filtering, which uses the similarity between two user profiles to suggest friends to a user.

Researchers have been using various collaborative filtering approaches in the recommendation of most suitable web-services to the users, for maximizing the productivity and satisfied user experience. In an early research, Chen et al. presented a study [4] on the recommendation and visualization of personalized web-services with recognition of web-service quality. They propose a collaborative filtering model designed to recommend large-web-services. Firstly, they combine model-based and memory-based collaborative filtering algorithms for web-services recommendation that improve time complexity and recommendation accuracy compared to previous web-services recommendation methods. Chen et al. propose a system [6] to calculate exact similarity based on historical QoS data available for a prediction algorithm. They focus on the scarcity of historical data, which affects the calculation of similarity which is a key factor in collaborative filtering. They argue about the problem of QoS scores prediction by taking into account the effect of the QoS data on the Collaborative Filtering method [6]. Their model can be understood in three steps: calculation of similarity, selection of neighborhood entities, and prediction based on this data. They pre-process the historical data in order to generate the required users-services rating matrix referred to by R . After which the neighborhood selection is performed in which the calculated similarity matrix is employed to select the similar neighbors set for which strategies based on the threshold and the top K similar entities are used for neighborhood selection. In the third step, the top chosen neighbors are used by the neighborhood-based Collaborative Filtering method for the final QoS prediction, or integrated into the Matrix Factorization model for learning a prediction model.

2.1 Contextual information and latent correlations

Xu et al. introduce the notion of context in the recommendation of web-services by defining the context as hidden relational information about the entities involved, users and web-services [29]. They discuss that Matrix Factorization approach can decompose a high-level matrix into multiple lower-dimensional matrices. For the prediction of the quality of web-service, the Human web-service-Invocation Matrix can be taken into account as the User-Characteristics Matrix and the web-service-Characteristics Matrix. The perception is that the lack of Quality of web-service that a user receives when invoking a web-service depends on how the user's hidden factors affect the latent factors of web-service. They emphasize on the location where the web-service is hosted as the web-service context because the web-services offered by the same vendor are likely to share the same execution resource. In their method, the web-services managed by the same vendor as the selected web-service form the neighboring set. They also introduce second-type neighbors to cater to the services that have very few or no neighbors. The two neighbors of the first type and the second type are used in a coupled environment for the operation of their proposed model. This study presents methods which map geographical distance to the similarity and select the best one. Secondly, they confirm that contextual data are really useful in predicting quality of web-service.

Another study by Chen et al. [5] also discusses the use of contextual information for predicting the web-service quality [5]. They argue that the QoS score of a web-service relates to the scores of its geographic neighbors. They present a neighborhood matrix factorization model based on Unified web-service geographical locations (GNMF) that improves prediction accuracy by taking advantage of neighborhood approaches and latent features. In this approach, for a web-service, a set of geographically similar neighbors is summarized at the region level which also is according to the latent geographical information and the QoS matrix of the user*service. The neighbor web-services are systematically integrated into a

Matrix Factorization Machine, and the prediction model is resolved as an optimization problem. They used the location context in relation to web-services and but did not utilize the geographical latent information at users' side.

Li et al. propose a system of recommendation of the web-services based on the recognition of context characteristics on the server side. They argue that current research only focuses on QoS information in client side to predict missing QoS scores [10]. Hence, they focus on using context features from server-side web-services and predicting missing values. They argue that in real-world applications, the context characteristics of users and web-services (e.g., the functional categories provided by a web-service) greatly affect the quality of the web-service. In their system, the algorithm takes into account the similarities of the client-side QoS values. At the server side, details can be obtained about functional models of web-services by analyzing the context characteristics of WSDL files. This methodology bases upon on the technique of matrix factorization and takes into account both the historical call records of the web-services of the users and the context characteristics of the web-services.

Chen Wu et. al. in his study [26] talk about calculating credibility of data as the latent correlation factor and using it as basis to perform the prediction of missing QoS values. They use a two-phase clustering mechanism. In the first phase, they perform clustering on the historical data to screen-out the untrustworthy users. In the second phase, the users are clustered based on their untrustworthy index after which the model predicts the missing QoS scores. This method has a basis on the users based similarity, focusing on the users' perspective of the historical data. Su et. al. [23] introduced trust-based prediction methodology in which latent correlation factor of trust between users is evaluated using historical data. This trust factor determines how much the entities correlate with each other. They use a collaborative filtering to perform the prediction of the missing QoS values. They use reputation and user-user similarity to calculate a trust factor between users. Based on a list of the most trusted users, the QoS prediction for a user is performed. In this work, the services perspective has been ignored which holds equally valuable information in terms of statistics. This leads us to the thought process of incorporating the use of services perspective for calculating latest correlation factors, which became the basis of the proposed research work. Although an exhaustive research on the focused topic would be impossible to perform, the research work performed provided a thorough insight in different perspectives of the web-services systems, and the how they influence the recommender systems. Multiple techniques from the literature record were studied experimented with and architecture for the proposed model of a web-services recommender system was formulated.

3 Proposed methodology

The S-RAP model considers the data in the two perspectives of users and web-services and processes these data in separate components to generate a predicted QoS values. It derives latent co-relational concepts between the entities from the available data and uses it in the prediction model. The study introduces a metric "relevance" in the perspective of web-services, as the primary contribution, which represents the degree to which two web-services are relevant to each other, and hence it has been named Services-Relevance-Aware Prediction (S-RAP). In the users' perspective, the system uses a trust metric in the prediction of the missing QoS figures. Both components work independently, while their results are compiled together in the final phase of the algorithm to produce the final predicted QoS values. The working of the entire system is explained in the subsections.

3.1 Web-services perspective

On the web-service side, data are initially clustered on basis of users to ensure consistency in similarity of web-service usage. In proposed system, k-means clustering was used. The clusters are made with Eq. 1 where J defines the extent to which a rating value belongs to a cluster. Here, the C_j^k is the k th cluster for web-service j and μ is the center of k th cluster. R_{ij} specifies the observed rating of the service j by the user i .

$$J = \sum_{k=1}^{k_u} \sum_{R_{ij} \in C_j^k} \|R_{ij} - \mu_j^k\| \quad (1)$$

3.1.1 Services similarity

To understand relevance, it is considered that similar web-services observed by the users in the same clusters will have a higher degree of relevance. To compute the degree of the relevance among the web-services, the web-services are clustered for each user based on user-rating according to Eq. 2.

$$J = \sum_{i=1}^U \sum_{q=1}^{q_u} \sum_{R_{ij} \in C_j^q} \|R_{ij} - \mu_j^q\| \quad (2)$$

One thing to note here is important that this clustering is on the one-dimensional web-services rating data against each user, unlike the clustering done in the previous step. This process also gives a trend in the rating values assigned by a user, since web-services of similar quality have similar rating values and vice versa. After getting the web-service clusters for each user, the number of times two web-services are clustered in the same cluster is recorded. It can be understood by Eq. 3:

$$f(s_j, s_r) = \sum_{u \in U_w} I_u(s_j, s_r) \quad (3)$$

Here, function I indicates if the web-services s_s and s_r are in the same cluster for the user u , it is a Boolean function. The function f returns the number of times that these two web-services are grouped together against all the users.

It should be noted here that the minimum frequency with which two web-services can be grouped together can be zero and the maximum number can be equal to the number of users, which means that these two web-services are each grouped together in the same group of users. These numbers for each web-service are calculated and managed separately to compute the factor of similarity between web-services using Eq. 4 [23].

$$\text{Sim}(s_j, s_r) = (f(s_j, s_r) - f_{\min}(s_j)) / (f_{\max}(s_j) - f_{\min}(s_j)) \quad (4)$$

This equation takes the degree of occurrence of the two web-services in the same group and normalizes them against web-service s_j . This obtained similarity is in the interval $[-1, 1]$, where the greater the similarity, the greater the value.

3.1.2 3-Sigma rule

It is a general idea that the web-service QoS scores observed by a wide range of users follow a Gaussian distribution, which helps us to use the 3-sigma (standard deviation) rule of the Gaussian distribution to our advantage, which indicates that the probability that a QoS value

observed by a user is within 3 sigmas on both sides of the distribution mean is 99.7%, as in Eq. 5 [22].

$$P\left(\mu_j^{\max} - 3\sigma_j^{\max} < R_{ij} \leq \mu_j^{\max} + 3\sigma_j^{\max}\right) = 0.997 \quad (5)$$

These feedback values are classified as positive or negative based on this information. A feedback is positive if its difference with the mean value is less than or equal to the 3-sigma value, and negative if greater. Equation 6 explains this process mathematically.

$$R_{ij} = \begin{cases} \text{Positive, if } \left| R_{ij} - \mu_j^{\max} \right| \leq 3\sigma_j^{\max} \\ \text{Negative, if } \left| R_{ij} - \mu_j^{\max} \right| > 3\sigma_j^{\max} \end{cases} \quad (6)$$

In this process, web-service feedback vectors are created for each web-service that identify a review experience as positive or negative, and positive and negative review counts are maintained for each web-service. This information is then used to evaluate the web-services and assess their reputation. The rank/reputation of a web-service is directly proportional to the number of positive reviews. Reputation mechanisms can encourage honest feedback and help users decide whom to trust. The beta reputation system is a widely known trust rating methodology based on probability in which the reputation is computed by combining an a priori reputation score with the new feedback information [25]. This function in Eq. 8 is used here to signify the probability distribution of the binary event for the occurrence of negative or positive feedback. Equation 7 defines the probabilistic variables alpha and beta.

$$\alpha = p_i + 1 \quad \text{and} \quad \beta = n_i + 1 \quad \text{where } p_i, n_i \geq 0 \quad (7)$$

As mentioned earlier, it is assumed that observed feedback vector of negative and positive values of a user u_i contains the positive feedback referred to by p_i and the negative feedback referred to by n_i . Further, to obtain the user probability density function of u_i , which will provide positive feedback in future, the following information will be used.

$$p_e = \alpha / (\alpha + \beta) \quad (8)$$

Here, P_i and N_i are the positive and negative feedbacks for a web-service. This equation presents that once feedback vector of a web-service is known, value of the rank can be dynamically computed. Keeping in view Eqs. 6 and 8, we can calculate the said web-service rank using the equation [24]. The range of the Rank attribute is within [1], where the higher the value, the higher rank can be calculated by Eq. 9:

$$\text{Rank}(S_i) = (p_i + 1) / (p_i + n_i + 2) \quad (9)$$

3.1.3 Services relevance

Now, to calculate the Relevance values, the reputation/rank values are used together with the similarity values calculated earlier. This can be understood through Eq. 10. The rank of a web-service directly affects its relevance with another web-service. If rank of a web-service is high, and it has a high similarity with another web-service, both these web-services will be highly relevant.

$$\text{Relevance}(S_i, S_j) = (2 * \text{Rank}(S_j) * \text{Sim}(S_i, S_j)) / (\text{Rank}(S_j) + \text{Sim}(S_i, S_j)) \quad (10)$$

As discussed earlier, the relevance attribute is calculated for all the web-services in the context of each cluster separately, after which the relevance values for these web-services

are conjoined by employing a lambda parameter which is derived based on the initial cluster sizes proportionally, as in Eq. 11, where the assumption is a larger cluster provides more accurate data as compared to a smaller cluster.

$$\begin{aligned} \text{Relevance}_U (S_i, S_j) = & \text{Relevance}_1 (S_i, S_j) * \lambda_1 + \text{Relevance}_2 (S_i, S_j) \\ & * \lambda_2 + \text{Relevance}_3 (S_i, S_j) * (1 - \lambda_1 - \lambda_2) \end{aligned} \quad (11)$$

where Relevance_u is the universal relevance between the S_i and S_j . The relevance calculated here is a value within the range $[-1, 1]$.

3.1.4 QoS values prediction at web-services side

After the relevance values for all the web-services have been calculated, the system moves on to the part where the prediction values are calculated. In this phase, a neighbor list of top K most relevant web-services of each web-service is defined, which is then used in the calculation of the prediction value in this web-services context. The prediction value is computed based on Eq. 12.

$$\text{Prediction}_{ij}^{(s)} = \left(\sum_{s_r \in S(s_j)} R_{ir} * \text{Relevance}(s_j, s_r) \right) / \left(\sum_{s_r \in S(s_j)} \text{Relevance}(s_j, s_r) \right) \quad (12)$$

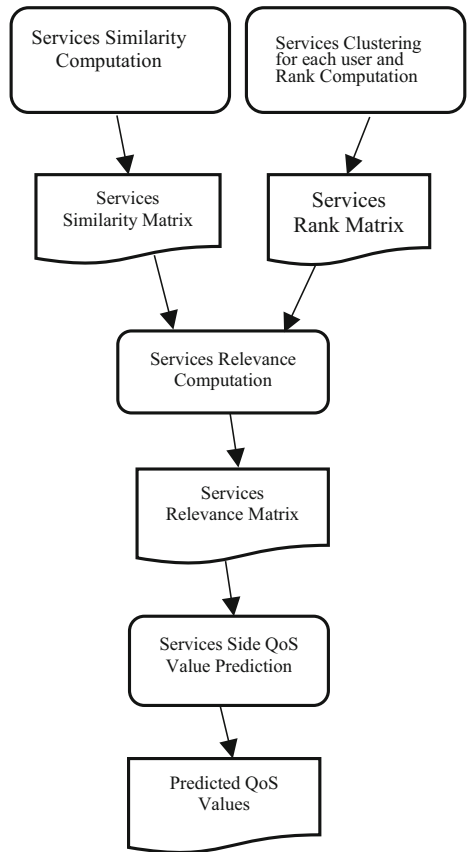
where $\text{Prediction}_{ij}^{(s)}$ is predicted figure of the web-service s_j observed by the user u_i in web-services context. The web-service referred to as s_r is a neighbor web-service of s_j . $S(s_j)$ is the neighbor set of the web-service s_j . R_{ir} is the QoS value experienced by the user u_i of web-service s_r . This prediction value is later used with the output value of the algorithm designed in the user context.

Figure 1 displays the overall flow of the functions and processes performed on the Web-Services side of the framework. The Services Similarity Computation block works based on aforementioned Eqs. 2, 3, and 4. Equations 5–9 are used in the Services Clustering for each user and Rank Computation block which results in the Services Rank Matrix as the output. Equation 10 makes use of the output of the previous two blocks in the computation of Services Relevance Computation as can be seen in Fig. 4, the output of which is Services Relevance Matrix. Equation 11 normalizes the Relevance values from different clusters into one relevance value for a service. Finally, Eq. 12 uses the output of the Services Relevance Computation block to make the prediction of missing QoS values.

3.2 Users' perspective

On the users side of the algorithm, a similar mechanism is used for the QoS prediction, as in Fig. 2, referred to as trust-aware prediction [23]. The data are clustered first with the interpretation that majority of the historical QoS data fall into a same range, as discussed by Zibin et al. [33]. This also supports that most of the data which deviate from the normal range would be dishonest observation data, QoS values. In simpler words, an observation that highly deviates from the normal value is not likely to occur. Thus, if a user always submits QoS feedbacks which highly deviate from majority, they are perhaps not an honest user. On the basis of this supposition, probability of a user being honest can be evaluated according to his past submission in UCluster. For clustering the users, K-means clustering is employed

Fig. 1 S-RAP Web-Services Side QoS Prediction



as in Eq. 13.

$$J = \sum_{k=1}^{k_u} \sum_{R_{ij} \in C_j^k} ||R_{ij} - \mu_j^k|| \tag{13}$$

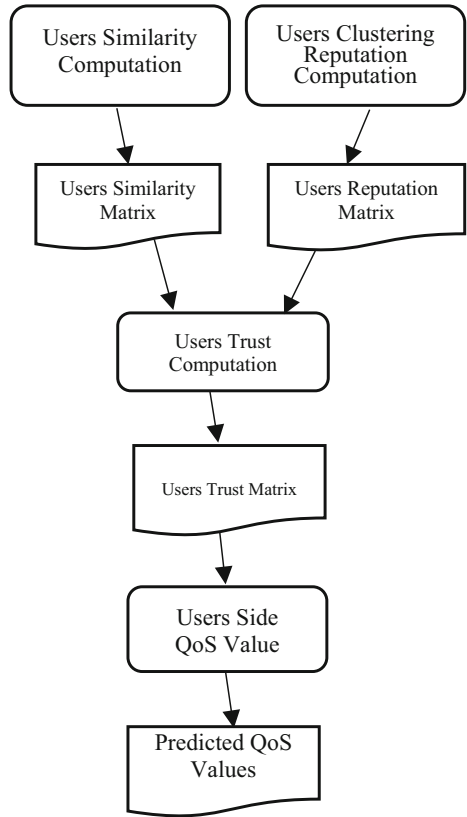
the C_j^k is the k th cluster for web-service j , and μ is the center of k th cluster. The system assumes that the majority of the users are honest in nature, and hence the largest cluster of the users is considered as the honest cluster [23]. Similar to web-services side, the positive and negative figures observed are calculated against each user, vectors of which are maintained, in the users context this time as in Eq. 14.

$$R_{ij} = \begin{cases} \text{Positive,} & \text{if } |R_{ij} - \mu_j^{\max}| \leq 3\sigma_j^{\max} \\ \text{Negative,} & \text{if } |R_{ij} - \mu_j^{\max}| > 3\sigma_j^{\max} \end{cases} \tag{14}$$

3.2.1 Users reputation

In this process, the users-feedback vectors for each user are created which identify a rating experience being positive or negative, and a count of positive and negative ratings is maintained for each web-service. This insight is then employed for evaluating the reputation of

Fig. 2 Users Side QoS Prediction



users. Reputation of a web-service and the number of positive ratings of the feedbacks are directly proportional. Reputation mechanisms provide a motivation for honest rating and help users in making the decision of whom to trust. This function in Eq. 15 is used here to calculate the probability of binary event of occurrence of either a positive or a negative feedback.

$$\text{Reputation}(u_i) = P_i + 1 / P_i + N_i + 2 \tag{15}$$

3.2.2 Users similarity

After reputation, the similarity among the users is calculated using the Pearson Correlation Coefficient metric in Eq. 16:

$$\text{Sim}(u_i, u_a) = \left(\sum_{j \in S_{ia}} (R_{ij} - \bar{R}_i)(R_{aj} - \bar{R}_a) \right) / \left(\sqrt{\sum_{j \in S_{ia}} (R_{ij} - \bar{R}_i)^2} \sqrt{\sum_{j \in S_{ia}} (R_{aj} - \bar{R}_a)^2} \right) \tag{16}$$

where $\text{Sim}(u_i, u_a)$ is similarity between the users u_i and u_a . The calculated similarity is between the range of $[-1, 1]$. Any value closer to the positive 1 would mean that the two users are more similar in behavior. R_{ij} is the QoS observed by the user i for the web-service s_j , and \bar{R}_i represents mean of the QoS values observed by user u_w .

3.2.3 Users trust

To calculate the trust factor between two users, the reputation and similarity, both, are employed in Eq. 17 [23]:

$$\text{Trust}(u_i, u_a) = (2 * \text{Rep}(u_j) * \text{Sim}(u_i, u_a)) / (\text{Rep}(u_j) + \text{Sim}(u_i, u_a)) \quad (17)$$

$\text{Rep}(u_j)$ is reputation of the user j , and $\text{sim}(u_i, u_j)$ is similarity as calculated in Eq. 15. Since the figure of reputation is within the range [1], and similarity is within the range [-1,1], the trust calculated is in the range [-1,1]. The higher the similarity between the users and the higher the reputation of the second user would be, the more would they be trust worthy for the first user.

3.2.4 QoS values prediction at users side

The QoS value is predicted on the basis of trust value as per Eq. 18:

$$\text{Prediction}_{ij}^{(u)} = \bar{R}_i + \left(\left(\sum_{u_r \in S(u_i)} \text{Trust}(u_i, u_r) * (R_{rj} - \bar{R}_a) \right) / \left(\sum_{u_r \in S(u_i)} \text{Trust}(u_i, u_r) \right) \right) \quad (18)$$

The mean value of the neighbor user is subtracted from the calculation and alternatively, and the mean value of the subject user is added in the evaluation to remove personal biasness at the user level, with the assumption that the overall biasness of a user would remain same for a web-service. Note that in this phase the system has used the evaluation mechanisms in the users' context and a normalized prediction QoS value is generated.

3.3 Predictions accumulation

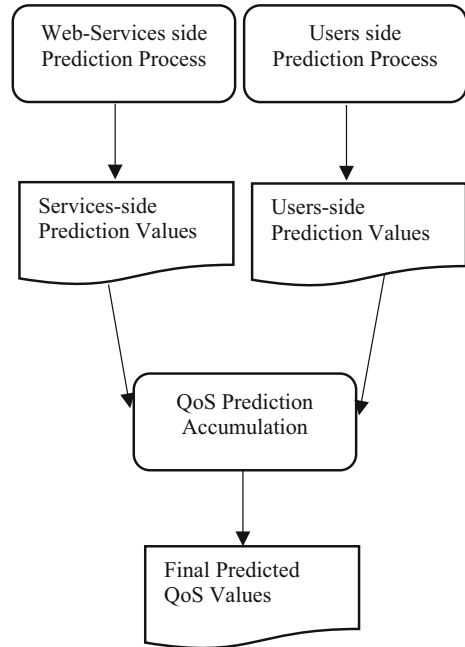
From the research of similar models working on the web-services in the literature, it has been observed that most models focus only one side of the data. If the users perspective is addressed, the web-services perspective is often overlooked and vice versa. If only one of the discussed methods is followed, the information in the other context would be wasted. It clearly implies that results obtained from one of these methods have room for improvement. However, if both the prediction scores are accumulated with a mechanism, the resultant values prove to be more accurate. A lambda parameter is used to combine both predicted values to produce the final active prediction value for a given user and web-service.

$$\hat{R}_{ij} = \lambda * R_{ij}^S + (1 - \lambda) * R_{ij}^U \quad (19)$$

In Eq. 19, \hat{R}_{ij} is the active prediction, final output of the whole model. R_{ij}^S is the prediction value in the web-services context, and R_{ij}^U is the prediction value in the users context.

An overall schema overview of the S-RAP can be seen in Fig. 3. Both the modules, working on the services perspective and users perspective of the data execute simultaneously on the same dataset, the outputs of which are Services-side Prediction Values and Users-side Prediction Values. These outputs are used in Eq. 19 as inputs to accumulate the predictions using a lambda mechanism and Final Predicted QoS Value as output. In the accumulation process, the lambda is the deciding factor of the extent to which the users and services perspective responsible for the final output of the model. The degree of the scarcity of the dataset plays an important role in deciding the lambda value in a real-world scenario. It

Fig. 3 S-RAP Complete Schema



would be reasonable to state that the lambda value stands for the contribution of the data in an entity's perspective (user's or web-services') in the prediction of the QoS scores. Setting an extreme value to the lambda would result in the biasness of the final prediction towards of the involved entities, rendering the effect of the lambda useless. The experimentations have supported this speculation as well, giving the optimal values when a moderate lambda value is set retaining some extent of insight from both web-services and users' perspectives.

4 Experimentation and results

4.1 Dataset details

WS-DREAM, Web-Services – Distributed RELiability Assessment Mechanism, is a project in the Chinese University of Hong Kong [12]. It focuses on the assessment and evaluation of web-services, being used over distributed networks. At the time, they host two types of dataset, a) QoS Datasets, b) Usage-log Datasets. Log datasets have system logs of web-services invocation, execution, and delivery with other details, while the QoS datasets, which have been employed in this research work, contain the QoS scores of web-services used by a number of users. The dataset used in this project is referred to as “WS-DREAM web-service QoS dataset#1 [13]. It provides real QoS values of response-time and throughput delivered by a web-service, obtained from 339 users on 5,825 web-services. This dataset has approximately 2 million user feedbacks, as per their experiences of the respective web-services.

4.2 Experimentation setting

4.2.1 Generating untrustworthy data

Different types of users can have different criteria of rating a web-service, some of which ratings could be biased due to unfavorable conditions of network, traffic, and many other things either on the users' side or the server side. To simulate that factor in the original data, a certain amount of users' data is removed and is replaced with randomly generated figures. These figures are incorrect ratings affected by factors mentioned earlier. These users and values would represent the untrustworthy users. The density of the untrustworthy users is defined, according to the requirements of the projected experiment. The decision of a user being untrustworthy or not is based on a probabilistic process through random number generation and is decided at runtime. The advantage of making this process dynamic is that it simulates the real-world experience more accurately, unlike any other setting used in the experiments in the literature review.

4.2.2 Removing data

Once the data with an untrustworthy percentage of users have been prepared, a set density of the data from the entire data matrix is removed from the data matrix. This has been done to represent the unexperienced services by the users so that later on the accuracy of the proposed model can be tested. These removed data values are saved separately to serve as ground truth for evaluation. This density of removed entries, data cells in the matrix, would serve the purpose of the unexperienced web-services by the respective users, giving the experiment a more realistic simulation. In this step as well, the users*services are selected dynamically to give the experiment a realistic feel. The values removed are preserved separately in a matrix for the future reference, along with their index positions. These preserved values are used as reference values for the evaluation of the designed approach.

4.2.3 Evaluation metrics

Each of the experiments is performed number of times and mean values are considered the final values. The data is then consolidated and evaluated. Generally, the Mean Absolute Error (MAE) metric is used for evaluating the accuracy of the prediction in the recommender systems [16].

$$\text{MAE} = \left(\sum_{ij} (R_{ij} - \hat{R}_{ij}) \right) / N. \quad (20)$$

where R_{ij} is the QoS values experienced by the user i for web-service j , \hat{R}_{ij} is predicted value, and s represents total number of the values that were removed in the earlier step.

$$\text{NMAE} = \text{MAE} / \left(\sum_{ij} (R_{ij}) / N \right) = \left(\sum_{ij} (R_{ij} - \hat{R}_{ij}) \right) / \sum_{ij} (R_{ij}) \quad (21)$$

The MAE specifies the exact deviation of the predicted figures from the original, historical values. It provides insight about the overall error, but does not work well when a comparison is needed where the data being compared are on different scales. This is handled by normalizing the deviation values to a standard scale, through which the results can be compared and evaluated. The evaluation metric that has been used here is the Normalized Mean Absolute Error (NMAE).

MAE is normalized to obtain a scaled difference (percentage) of the predicted values from the actual ones. This difference helps in the accuracy comparison of differently scaled models.

Another evaluation metric that has been used in this research study is the Normalized Roots Mean Squared Error (NRMSE). The math behind these metrics tells that the NMAE gives a linear figure of prediction deviation from the original value; however, the NRMSE provides a quadratic, higher dimensional, figure of the prediction deviation. In the NMAE, the individual errors are all given equal importance/weight while calculation an average deviation, but in NRMSE, the higher deviations get a higher weight in the calculation of the average deviation [27]. This is important in case of QoS scores prediction because a higher deviation from the original value would be critical for a model. Since the variances of QoS scores, such as throughput or response, matters even by tenth of a second, a higher prediction different would affect the accuracy of the recommender greatly. The working of RMSE and NRMSE can be viewed in Eqs. 22 and 23.

$$\text{RMSE} = \sqrt{\left(\sum_{ij} (R_{ij} - \hat{R}_{ij})^2\right) / N} \quad (22)$$

$$\text{NRMSE} = \text{RMSE} / \left(\sqrt{\sum_{ij} (R_{ij})^2 / N}\right) = \sqrt{\left(\sum_{ij} (R_{ij} - \hat{R}_{ij})^2\right) / \left(\sum_{ij} (R_{ij})^2\right)} \quad (23)$$

From these equations, it can be seen that RMSE gives a deviation figure within the scope of the data where the values are within a specified range. Different models have different range of values, and hence RMSE cannot be used to make a comparison between such models. NRMSE normalizes the value to a standard range, mostly [1] unless specified otherwise, which makes the comparison easier. The NRMSE gives a holistic view of the deviation of the prediction and can be used relatively. The experiment is performed multiple times with the one setting, and the mean value is selected as the final evaluation with that setting.

5 Results

The experiment was set with 339 users and 5825 web-services, and the other variables involved are as follows. Percent of untrustworthy users refers to the number of users deemed untrustworthy in the first step. Density of the data removed is the percentage of the data removed to represent the unexperienced entries. Trust threshold represents the least amount of trust that must exist from primary to secondary user for the secondary user to be involved in the calculation of predicted QoS value for the primary user. Web-services relevance lambda represents the weightage of clusters while consolidating the relevance value between two web-services. Top K web-services is the number of web-services deemed most relevant for the web-services for which the QoS value is being predicted. And finally, active-lambda represents the weightage of the value obtained via web-services and users context in generating the final, active QoS predicted value. The outputs with different settings of these variables can be seen in Tables 1, 2, 3, and 4.

Tables 1, 2, 3, and 4 present the NMAE and NRMSE results obtained through varying settings. It can be seen that the model developed in the web-services context proves to be more effective more than the approach in the users' context. Higher NMAE and NRMSE values would mean that the prediction deviates from the original value highly. One other thing that is evident here from the data is that the active prediction obtained by the integration of the predictions from the models in the web-services and users context returns an even more

Table 1 S-RAP Prediction NMAE with changing density of the removed data

Experiment settings							
Users	339	339	339	339	339	339	339
Services	5825	5825	5825	5825	5825	5825	5825
% of untrustworthy users	10	10	10	10	10	10	10
% of the data removed	5	8	10	13	15	18	20
Trust threshold	0.5	0.5	0.5	0.5	0.5	0.5	0.5
S-relevance lambda	1/3	1/3	1/3	1/3	1/3	1/3	1/3
Top K web-services	5	10	10	15	5	10	10
Active-lambda	0.5	0.5	0.5	0.5	0.5	0.5	0.5
QoS prediction	Predicted values (NMAE)						
Services Context NMAE	0.6056	0.6021	0.5612	0.5623	0.5714	0.5698	0.5835
Users Context NMAE	0.6435	0.6183	0.6014	0.5989	0.6182	0.6204	0.6353
S-RAP NMAE	0.5755	0.5324	0.5186	0.5118	0.5278	0.5335	0.5291

Table 2 S-RAP Prediction NRMSE with changing density of the removed data

Experiment settings							
Users	339	339	339	339	339	339	339
Services	5825	5825	5825	5825	5825	5825	5825
% of untrustworthy users	10	10	10	10	10	10	10
% of the data removed	5	8	10	13	15	18	20
Trust threshold	0.5	0.5	0.5	0.5	0.5	0.5	0.5
S-relevance lambda	1/3	1/3	1/3	1/3	1/3	1/3	1/3
Top K web-services	5	10	10	15	5	10	10
Active-lambda	0.5	0.5	0.5	0.5	0.5	0.5	0.5
QoS prediction	Predicted values (NRMSE)						
Services context NRMSE	0.5836	0.6079	0.5993	0.5811	0.6281	0.6612	0.6421
Users context NRMSE	0.9768	0.9355	0.9694	0.9721	0.9692	0.9752	0.9609
S-RAP NRMSE	0.6591	0.6826	0.6693	0.6082	0.6458	0.7293	0.7245

optimized NMAE. The increased density of the data removed helps with better and more accurate evaluation, and hence such results are more reliable. However, it is to be noted that the density of the data to be removed needs to be controlled; otherwise, too much data removal would result in data losing any real correlation, and metrics, such as relevance, rank, trust, and reputation, would return corrupted figures. The graphs in Figs. 4, 5, 6 and 7 depict the aforementioned results at a glance.

The above-given figures present the trend of the prediction accuracy with the change in settings of the experiment. Figures 4 and 5 depict that the model produces optimum results

Table 3 S-RAP Prediction NMAE with changing density of untrustworthy users

Experiment settings							
Users	339	339	339	339	339	339	339
Services	5825	5825	5825	5825	5825	5825	5825
% of untrustworthy users	5	8	10	13	15	18	20
% of the data removed	10	10	10	10	10	10	10
Trust threshold	0.5	0.5	0.5	0.5	0.5	0.5	0.5
S-relevance lambda	1/3	1/3	1/3	1/3	1/3	1/3	1/3
Top K web-services	5	10	10	15	5	10	10
Active-lambda	0.5	0.5	0.5	0.5	0.5	0.5	0.5
QoS prediction	Predicted values NMAE						
Services context NMAE value	0.5261	0.5345	0.5612	0.5933	0.6224	0.6545	0.6752
Users context NMAE value	0.541	0.5543	0.5973	0.6015	0.6432	0.6674	0.6843
S-RAP NMAE Value	0.5015	0.5198	0.5404	0.5682	0.5978	0.6235	0.6973

Table 4 S-RAP Prediction NRMSE with changing density of untrustworthy users

Experiment settings							
Users	339	339	339	339	339	339	339
Services	5825	5825	5825	5825	5825	5825	5825
% of untrustworthy users	5	8	10	13	15	18	20
% of the data removed	10	10	10	10	10	10	10
Trust threshold	0.5	0.5	0.5	0.5	0.5	0.5	0.5
S-relevance lambda	1/3	1/3	1/3	1/3	1/3	1/3	1/3
Top K web-services	5	10	10	15	5	10	10
Active-lambda	0.5	0.5	0.5	0.5	0.5	0.5	0.5
QoS prediction	Predicted values NRMSE						
Services context NRMSE value	0.6295	0.6403	0.6117	0.6358	0.6645	0.6781	0.6825
Users context NRMSE value	0.915	0.9214	0.9338	0.9505	0.9103	0.9321	0.9341
S-RAP NRMSE Value	0.6772	0.6962	0.6525	0.6982	0.7157	0.7236	0.7395

when a matrix density of 10–15 percent is removed in the calculation of the NMAE and NRMSE. The reason is that the model should have an adequate amount of data to compare the predictions with. Similarly, from Figs. 6 and 7, with an increasing number of untrustworthy user data, the prediction accuracy decreases; hence, there is an inversely proportional relation between the prediction accuracy and the percentage of users deemed untrustworthy. The more the randomized data for users are added, the more it would corrupt the accuracy

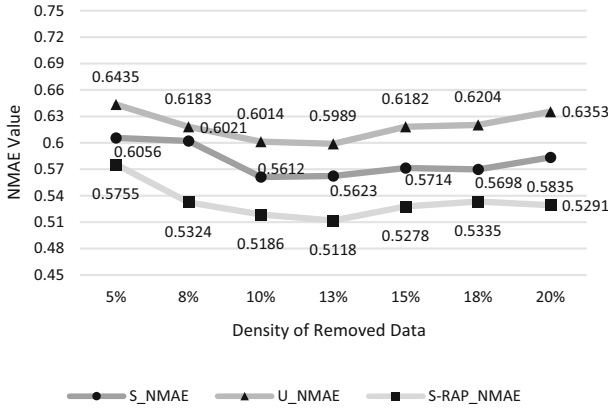


Fig. 4 S-RAP NMAE Values with Varying Densities of the Removed Data

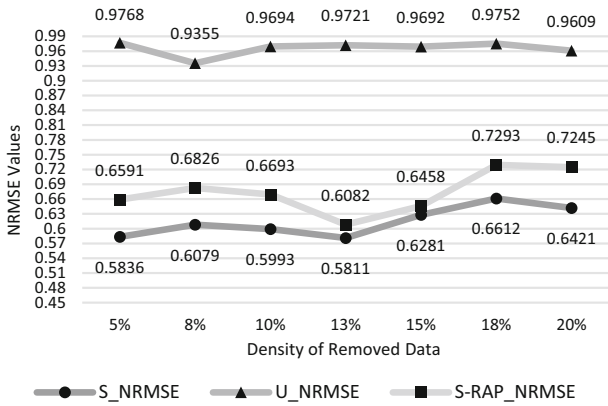


Fig. 5 S-RAP NRMSE Values with Varying Densities of the Removed Data

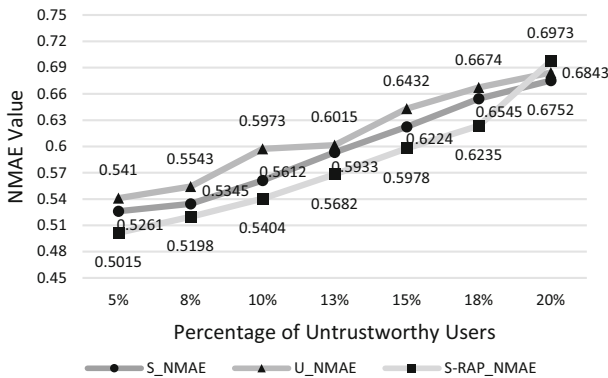


Fig. 6 S-RAP NMAE Values with Varying Percentage of Untrustworthy Users

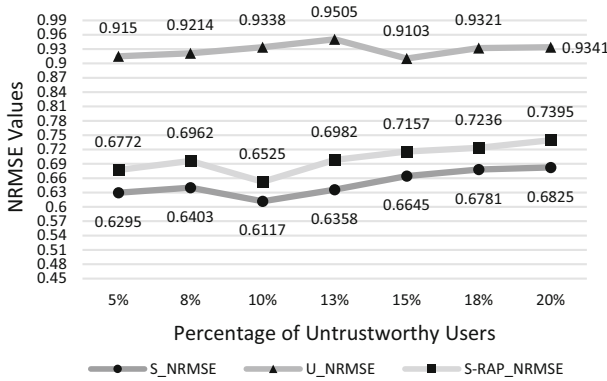


Fig. 7 S-RAP NRMSE Values with Varying Percentage of Untrustworthy Users

and effectiveness of the model. The NRSME values obtained in the experimentation suggest that the part of the algorithm in the users’ context produces higher errors compared to the errors as produced by the technique on web-services’ side. Due to that, the final prediction is affected, incorporating a slightly higher erroneous factor than the web-services’ side algorithm alone. The proposed S-RAP model is compared (Table 5) with the other recommender system/models, and it can be observed that the presented approach has provided satisfactory results, with the mean deviation of around the figures of 50%. From historical data, and literature review, it can be argued that the model is in the right direction with using the data in predicting the QoS values. Table 5 presents the values of Normalized Mean Absolute Error; the lower the NMAE, the better.

The model UIPCC [32] is a hybrid collaborative filtering approach that combines the user-based and service-based filtering to utilize the information of similar users and similar services. RAP [29] is a reputation-aware prediction technique. It firstly evaluates the reputation of users based on the historical user data available. The low reputed users’ data are excluded, and finally a hybrid CF approach is used to make the QoS prediction. CAP [25] is a credibility-aware prediction model employing two-phase K-means clustering for identifying the untrustworthy users. It also falls in the collaborative filtering domain, working and significance of which has been explained earlier in the introduction. It focuses on utilizing

Table 5 S-RAP Comparison with state-of-the-art approaches

Used approach	Prediction NMAE with untrustworthy users/removed density					
	10%/5%	10%/7%	10%/9%	10%/11%	10%/13%	10%/14%
UIPCC	1.434	1.364	1.271	1.183	1.065	1.044
RAP	0.995	0.970	0.911	0.925	0.872	0.815
CAP	0.658	0.625	0.590	0.586	0.574	0.579
TAP	0.598	0.581	0.547	0.542	0.529	0.531
S-RAP	0.575	0.556	0.530	0.528	0.511	0.513

the information of credible similar users to make the QoS prediction and TAP [23] is a trust-aware prediction model that focuses on calculation trust factor between users and using that to perform the prediction of the missing values.

From Table 6, it can be seen that the S-RAP approach produces accuracy much better than other state-of-the-art methods. The average improvement in the accuracy of the proposed model as compared to UIPCC is 57%, 43% improvement compared to RAP, 12% compared to CAP, and 4% improvement compared to the TAP, as displayed in Table 6.

Figure 8 depicts the trend of change in the prediction accuracy affected by the increasing percentage of untrustworthy users. The more the randomized data for users are added, the more it would corrupt the accuracy and effectiveness of the model. In Fig. 6, it can be seen here that at start with a small percentage of data removed the deviation is slightly higher, which comes to a normal value once the density is increased to an acceptable extent. This analysis is beneficial in deciding the allowed extent of scarcity of data in a real-world scenario.

Table 7 shows that with the increase in the data removed, there is an increase in the erroneous figures being incorporated in the NMAE and NRMSE. The observed increase is uniform and exhibits virtually equivalence growth. The NRMSE figure emphasizes on higher erroneous figures by amplifying them when the deviation is squared. This gives insight that the users' perspective of the algorithm incorporates limitations and can be improved.

Table 6 Compared accuracy improvement of the S-RAP

State of the art approach	S-RAP prediction NMAE improvement
UIPCC	57%
RAP	43%
CAP	12%
TAP	4%

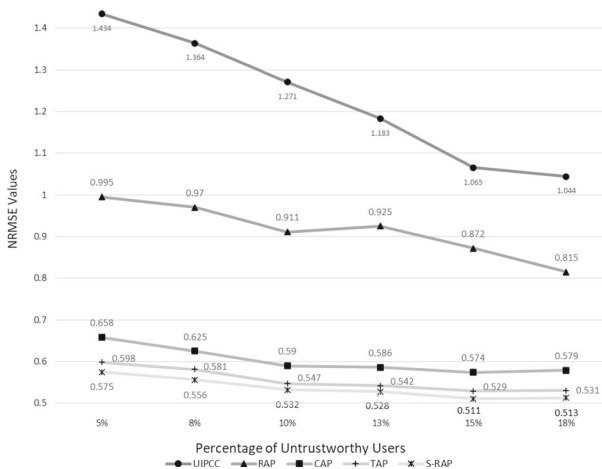


Fig. 8 Comparison with state-of-the-art approaches

Table 7 Prediction NMAE and NRMSE Comparison

Prediction NMAE and NRMSE comparison (untrustworthy users/removed data density)														
10% / 5%		10% / 8%		10% / 10%		10% / 13%		10% / 15%		10% / 18%		10% / 20%		
NMAE	NRMSE	NMAE	NRMSE	NMAE	NRMSE	NMAE	NRMSE	NMAE	NRMSE	NMAE	NRMSE	NMAE	NRMSE	
Services Context Prediction	0.6056	0.5836	0.6021	0.6079	0.5612	0.5993	0.5623	0.5811	0.5724	0.6281	0.5698	0.6612	0.5835	0.6421
Users Context Prediction	0.6435	0.9768	0.6183	0.9355	0.6014	0.9694	0.5989	0.9721	0.6182	0.9692	0.6204	0.9752	0.6353	0.9609
S-RAP Prediction	0.5755	0.6591	0.5324	0.6826	0.5186	0.6693	0.5118	0.6082	0.5278	0.6458	0.5335	0.7293	0.5291	0.7245

Table 8 Results obtained through varying setting of Web-Services side Lambda figure

	Prediction NMAE and NRMSE with different Lambda settings			
	Equivalent Lambda		Cluster size based Lambda	
	NMAE	NRMSE	NMAE	NRMSE
Web-Services Context Prediction	0.5593	0.5935	0.5271	0.5597
S-RAP Prediction	0.5439	0.6888	0.5397	0.6771

5.1 Effect of lambda parameter

The lambda mechanism has been used in the algorithms at two points. Once is when the services' relevance factor is generalized according to the different services clusters, and the other time is when the users-side and web-services-side prediction is combined. The effect of different settings of this lambda parameter will be discussed in this subsection.

5.1.1 Lambda at web-services side

To value lambda at this point, two mechanisms have been studied. One is where the lambda for all the clusters is equivalent, which means that all the clusters are given equal weight in the generalization of web-services relevance. However, it is a simple technique, and it makes sure that the information from all the clusters is retained and a practical web-services relevance is calculated. One disadvantage of this is that larger clusters, which may contain more accurate information as compared to the smaller clusters, do not get as much weight as the amount of information they pack. For that purpose, another is used in which the lambda weight given to a cluster technique is based on the size of that cluster. Because of that, most of the information in the larger clusters is retained, while some importance is given to the smaller clusters as well. The results obtained through both the stated settings are in Table 8. The results displayed in Table 8 are the average values obtained through multiple runs of the algorithm based on the different settings of Web-Services Lambda. It can be seen that the predictions based on equivalent lambda exhibit the extent erroneousness slightly more than the predictions based on the varying lambda decided on runtime according to the cluster sizes in the web-services module of the algorithm. This gives insight that further advanced statistical or machine learning techniques can be employed in the lambda definition on runtime to yield even better results.

5.1.2 Lambda at the prediction accumulation

At the stage of prediction accumulation, a similar mechanism of lambda evaluation has been used. One approach is that the lambda is given equal weight for both the users and web-services context. The average of predictions of both the users and web-services is taken against a user-web-service and that is considered as the final prediction. Statistically, this means that the information obtained through the historical data in both the users and web-services context is equally important and weighed. In the experimentation and through evaluation metrics, it has been observed that the algorithm displays some limitation in the users' context, where

Table 9 Results obtained through varying setting of active prediction lambda value

	Prediction NMAE and NRMSE with different lambda settings			
	Equivalent lambda		Lambda inclined towards web-services-based prediction	
	NMAE	NRMSE	NMAE	NRMSE
Web-services Context Prediction	0.5920	0.5909	0.5767	0.5765
User context Prediction	0.6861	0.9820	0.6241	0.9742
S-RAP Prediction	0.6114	0.6507	0.5532	0.6201

the NRMSE values evaluate to be higher, which in turns elevates the NRMSE of the active prediction. For that reason, we have employed statistical analysis on the obtained results and have derived that for the current algorithm, a lambda value inclining more towards the web-services context would yield the best results. The optimum value for lambda at the web-services context falls in the range [0.6, 0.7]. It is important that lambda is not set at either extreme since that would result in the consequent loss of important information in one of the contexts. The context for which an extreme high value of lambda is set would dominate the entire active prediction, silencing out the other context. Therefore, to prevent the ultimate loss of latent information at either side, while still optimizing the results, the range of lambda is set [0.6, 0.7] inclining towards web-services. The experiment is run multiple times with these settings of final prediction lambda, and average values are obtained which are displayed in Table 9. The NMAE and NRMSE obtained with the active prediction lambda setting based on statistical analysis of the obtained results produce better results. As discussed earlier, advanced machine learning techniques can be employed to further improve the lambda value at runtime based on the running instance of the algorithm.

6 Conclusion

This research study focuses on the web-services recommendation for normal users based on their preferences, usage, and needs. This work falls in the category of recommender systems which is a wide line of discipline being employed and utilized for an extensive range of purposes. Web-services have become way too overcrowded over the past few decades, with the exponential growth of bandwidth and internet availability. A few factors have caused this surge in the availability of such a huge variety some of which are availability of high speed internet, its affordability, and vast coverage. Same is the matter in the case of smartphones and other smart devices, which have played a substantial role in the growth of web-services. This publication argues, in consensus with the literature, that with the availability of millions of web-services, offered by thousands of vendors and providers, there must be efficient and practical ways to recommend the perfect web-services to a user according to their requirements, needs, and demands.

The S-RAP approach follows the collaborative filtering methodology from the machine learning discipline. As in the literature [10, 15, 19], the historical data of user experiences are used to calculate various correlation figures between web-services and users, such as rank,

relevance, reputation, and trust. These evaluations are then used in a setting of collaborative filtering model to make the prediction of unknown-experienced ratings. In the proposed approach, the predictions are first made separately in the context of users and web-services, which are then combined to formulate the active prediction. The dataset used for experiments is WS-Dreams QoS Dataset#1, and a series of comprehensive experiments have been performed. From these experiments, results, and evaluation, it has been observed that S-RAP approach has the capacity to generate satisfactory prediction values for users-unexperienced web-services. Furthermore, the S-RAP approach has opened a perspective of seeing, manipulating, and utilizing the historical data in terms of contextual data so that more efficient prediction mechanisms can be studied and researched upon.

The proposed S-RAP approach opens the perspective of the web-services to be focused while designing prediction mechanisms and models. Along with focusing on finding the most suitable web-services for a user, models can be created that also focus on finding the most suitable users for the available web-services, keeping web-services in the primary emphasis. This can prove useful in creation of even more accurate models after detailed research and studies. Focusing on the data from perspective of users, the model exhibits room for improvement with further research upon analysis against the NRMSE values.

Different techniques can be used to create a framework that focuses on the data of users which can further be ensembled with the S-RAP framework to improve the results at the users' side. Furthermore, historical data with secondary contextual information can be integrated with S-RAP to derive more comprehensive models. Such type research work can be performed in extension to the proposed research which can potentially produce results with higher accuracies.

Funding Not applicable.

Availability of data and materials Dataset is referenced.

Code availability Available on demand.

Declarations

Conflict of interest Not applicable.

References

1. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Trans Knowl Data Eng* 17:734–749
2. Buyya R (2010) Cloud computing: the next revolution in information technology. In: 1st international conference on parallel distributed and grid
3. Cao B, Liu J, Wen Y, Li H, Xiao Q, Chen J (2019) QoS-aware service recommendation based on relational topic model and factorization machines for IoT Mashup applications. *J Parallel Distrib Comput* 132:177–189
4. Chen Xi, Zheng Z, Liu X, Huang Z, Sun H (2011) Personalized QoS-aware web service recommendation and visualization. *IEEE Trans Serv Comput* 6(1):35–47
5. Chen Z, Shen L, Li F (2017) Exploiting Web service geographical neighborhood for collaborative. *Futur Gener Comput Syst* 68:249–259
6. Chen Z, Shen L, Li F (2019) Your neighbors are misunderstood: on modeling accurate similarity driven 2 by data range to collaborative web service QoS prediction. *Futur Gener Comput Syst* 95:404–419
7. Ding S, Xia C, Cai Q, Zhou K, Yang S (2014) QoS-aware resource matching and recommendation for cloud computing systems. *Appl Math Comput* 247:941–950

8. Han S-M, Hassan MM, Yoon C, Huh E (2009) Efficient service recommendation system for cloud computing market. In: 2nd international conference on interaction sciences: information technology, culture and human
9. Ko JM, Kim CO, Kwon I-H (2008) Quality-of-service oriented web service composition algorithm and. *J Syst Softw* 81(11):2079–2090
10. Li S, Wen J, Luo F, Gao M, Zeng J, Dong ZY (2017) A new QoS-aware web service recommendation system based on contextual feature recognition at server-side. *IEEE Trans Netw Serv Manag* 14:332–342
11. Luis MV, Luis R-M, Juan C, Maik L (2008) A break in the clouds: towards a cloud definition. *ACM SIGCOMM Comput Commun Rev* 39:50–55
12. Lyu MR, Zheng Z, Zhu J, He P (2008) WS-DREAM. The Chinese University of Hong Kong. <https://wsdream.github.io/>. Accessed 20 Jan 2021
13. Lyu MR, Zheng Z, Zhu J, He P (2008) WS-DREAM Dataset#1. <https://github.com/wsdream/wsdream-dataset/tree/master/dataset1>. Accessed 20 Jan 2021
14. Maamar Z, Hacid H, Huhns MN (2011) Why web services need social networks. *IEEE Internet Comput* 15(2):90–94
15. Martinello M, Kaâniche M, Kanoun K (2005) Web service availability—impact of error recovery and traffic model. *Reliab Eng Syst Saf* 89:6–16
16. Mobasher B, Burke R, Jannach D, Adomavicius G (2011) Enhancing collaborative filtering systems with personality information. In: *RecSys '11: proceedings of the fifth ACM conference on recommender systems*, Chicago, Illinois, USA
17. O'Sullivan J, Edmond D, ter Hofstede A (2002) What's in a service? *Distrib Parallel Databases* 12:117–133
18. Portugal I, Alencar P, Cowan DD (2015) The use of machine learning algorithms in recommender systems: a systematic review. *Expert Syst Appl* 97:205–227
19. Prabhu R, Shetty P, Shilpa SDR, Hegde R (2018) A review: recommender system using collaborative filtering and gray sheep problem. *Int J Eng Dev Res* 6(2):440–443
20. Qiu W, Zheng Z, Wang X, Yang X, Lyu MR (2013) Reputation-aware QoS value prediction of web services. In: *IEEE international conference on services computing*, Santa Clara, CA, USA
21. Rosatti G, Zorzi N, Zugliani D, Piffer S, Rizzi A (2018) A Web Service ecosystem for high-quality, cost-effective debris-flow hazard assessment. *Environ Model Softw* 100:33–47
22. Shao L, Zhang J, Wei Y, Zhao J, Xie B, Mei H (2007) Personalized QoS prediction for web services via collaborative filtering. In: *IEEE international conference on web services (ICWS 2007)*. Salt Lake City, UT, USA
23. Su K, Xiao B, Liu B, Zhang H, Zhang Z (2017) TAP: a personalized trust-aware QoS prediction approach for web service recommendation. *Knowl-Based Syst* 115:55–65
24. Teacy WTL, Luck M, Rogers A, Jennings NR (2012) An efficient and versatile approach to trust and reputation using hierarchical Bayesian modelling. *Artif Intell* 193:149–185
25. Whitby A, Josang A, Indulska J (2004) Filtering out unfair ratings in bayesian reputation systems. *Icfain J Manag Res* 4:48–64
26. Wu C, Qiu W, Zheng Z, Wang X, Yang X (2015) QoS prediction of web services based on two-phase k-means clustering. In: *IEEE international conference on web services*. New York, NY, USA
27. Wu H, Yue K, Li B, Zhang B, Hsu C-H (2018) Collaborative QoS prediction with context-sensitive matrix factorization. *Future Gener Comput Syst* 82:669–678
28. Xiong R, Wang J, Zhang N, Ma Y (2018) Deep hybrid collaborative filtering for Web service recommendation. *Expert Syst Appl* 110:191–205
29. Xu Y, Yin J, Deng S, Xiong NN, Huang J (2016) Context-aware QoS prediction for web service recommendation and selection. *Expert Syst Appl* 53:75–86
30. Yao L, Sheng QZ, Ngu AHH, Ashman H, Li X (2014) Exploring recommendations in internet of things. In: *37th international ACM SIGIR conference on research & development in information retrieval (SIGIR '14)*. Association for Computing Machinery, New York
31. Zheng Z, Ma H, Lyu MR, King I (2013) Collaborative web service QoS prediction via neighborhood integrated matrix factorization. *IEEE Trans Serv Comput* 6(3):289–299
32. Zheng Z, Ma H, Lyu MR, King I (2011) QoS-aware web service recommendation by collaborative filtering. *IEEE Trans Serv Comput* 4:140–152
33. Zibin Z, Zhang Y, Lyu MR (2014) Investigating QoS of real-world web services. *IEEE Trans Serv Comput* 7(1):32–39



Hafiz Syed Muhammad Muslim is a master software engineer (2021) from National University of Sciences and Technology, currently associated with the Riphah International University, Islamabad, Pakistan, as a Jr. Lecturer. His research areas include, from the projects that he has worked on, recommender systems, NLP, Digital Image Processing, and Artificial Neural Networks in the AI domain. In the domain of Software Engineering, his research interests are Requirements Engineering, Design Engineering, and Quality Engineering. He's currently (2022) exploring research areas to pursue doctoral studies in. He believes in extensive collaboration and exploring new ventures with like-minded individuals in science and technology.



Saddaf Rubab is affiliated with the National University of Sciences and Technology since 2018. She completed her doctoral studies at Universiti Teknologi PETRONAS, Malaysia in 2018. She received her MSc in Computer Software Computer Engineering from NUST College of Electrical and Mechanical Engineering (CEME), in 2012 and worked on different academia positions from 2009 to 2013. She has worked on forecasts and implementing AI techniques in various interdisciplinary areas. Dr. Rubab is also working closely with local/international industries to bridge the gap between academia and industry which helped her secure multiple industrial projects. Her research interests include distributed computing, security, and prediction systems. Dr. Rubab has received research grants as PI and Co-I for ICT related projects from various local and international research funding authorities and has been working on scientific projects of undergraduate and graduate level.



Malik M. Khan completed his PhD (Computer Science) and Masters in Science (Computer Science) from University of Southern California, Los Angeles. He completed his Postdoc (High-Performance Computing) from Norges Teknisk-Naturvitenskapelige Universitet—NTNU—Trondheim, Norway in 2014, as a Marie-Curie fellow under the ERCIM fellowship program. Later he worked as a Research Assistant on EU-H2020 Cloudlightning Project being associated with the HPC lab. Previously, his research has focused on autotuning, code optimization and code generation technology and he developed his code optimization framework for CUDA Nvidia GPUs. Currently, his research interests are in Software Engineering for Security-Sensitive Enterprise Systems, DevSecOps, Scalable Cloud-Native Applications and AI-based solutions.



Naima Iltaf has received the Ph.D. degree in Software Engineering from National University of Sciences and Technology, Pakistan, in 2013. She is currently an Associate Professor with the Department of Computer Software Engineering, National University of Sciences and Technology. She is author or coauthor of more than 40 articles published in international journals and conferences. She is engaged with a few academic and industrial research projects as PI and Co-I. Her research interests include Data mining, text mining, Natural language Processing and Recommender Systems.



Ali Kashif Bashir is Reader at the Department of Computing and Mathematics, Manchester Metropolitan University, UK. He is also affiliated with the University of Electronic Science and Technology of China (UESTC), China, National University of Science and Technology, Islamabad (NUST), Pakistan, and the University of Guelph, Canada in honorary roles. He is associated with research and industrial projects as PI and Co-I, accumulatively of around 7 million pounds. He is also serving as reviewer for EPSRC, UK, Commonwealth, UK, National Science and Engineering Research Council (NSERC), Canada, MITACS, Canada, Irish Research Council (IRC), Ireland and Qatar National Research Fund (QNRF), Qatar. Since 2016, he is serving IEEE Technology, Policy and Ethics newsletter as EICHe has published over 200 research articles, delivered more than 35 invited talks across the globe, organized 40 + guest editorials in including IEEE Transactions, and chaired around 40 conferences and workshops.



Kashif Javed is working as an Assistant Professor at the Department of Robotics, NUST, Islamabad. He has completed his Doctoral degree in Computer Science from Åbo Akademi University (ÅAU), Finland in 2017. He was the recipient of full scholarship from Ministry of Science and Technology in 2010–2014 and 2015–2017 from Åkademi University, Finland. He has participated in numerous academia and industry projects. He has contributed effectively in scientific community through publication of high quality research articles in renowned peer reviewed journals alongwith several conferences. His research interests include machine learning for computer vision applications, aerial robotics, implementation of evolutionary algorithms on real-world applications and Artificial intelligence. He is a reviewer of many journals such as the International Journal of Control, Automation and Systems, Neural Computing and Applications, Neuro Computing, Computers Materials, and Continua, and name a few more.