TRQP: Trust-Aware Real-Time QoS Prediction Framework Using Graph-Based Learning

Suraj Kumar and Soumi Chattopadhyay(✉)

Indian Institute of Information Technology Guwahati, Guwahati, India
{suraj.kumar,soumi}@iiitg.ac.in

Abstract. QoS prediction algorithm requires to be real-time to be integrated with most real-time service recommendation or composition algorithms. However, real-time algorithms are prone to compromise on the solution quality to improve their responsiveness, which we aim to address in this paper. The collaborative filtering (CF) technique, the most widely used QoS prediction method, consider the influences of all users/services while predicting the QoS value for a given target user-service pair. However, the presence of untrustworthy users/services, whose QoS invocation patterns are different from the rest, may lead to degradation in prediction accuracy. Moreover, in many cases, the quality of the prediction algorithms often deteriorates to ensure faster responsiveness due to their inability to capture non-linear, higher-order, and complex relationships among user-service QoS data. This paper proposes a trust-aware QoS prediction framework leveraging a novel graph-based learning approach. Our framework (TRQP) is competent enough to identify trustworthy users and services while learning effective feature representation for finding a rich collaborative signal in an end-to-end fashion. Our experiments on the publicly available WS-DREAM-1 dataset show that TRQP is not only eligible as a real-time algorithm but also is well capable of handling various challenges associated with QoS prediction problems (e.g., extracting complex non-linear relationships among QoS data) and outperformed major state-of-the-art methods.

1 Introduction

Recommending suitable service for a target user comes under commercial and personal interest. However, it is a challenge in a decentralized environment, where the functionally equivalent web services are increasing rapidly. Due to the frequent addition of new functionally redundant services, obtaining the QoS profile for each service for every user is practically infeasible and time/resource-consuming. Therefore, QoS prediction [6] of services across different users appears as a fundamental problem to solve.

This work is supported by the Science and Engineering Research Board, Department of Science and Technology, Government of India, under Grant SRG/2020/001454.

J. Troya et al. (Eds.): ICSOC 2022, LNCS 13740, pp. 143–152, 2022.
https://doi.org/10.1007/978-3-031-20984-0_10

Recent studies reveal that collaborative filtering (CF)-based methods are most effective for QoS prediction [24]. CF-based methods exploit the QoS log for predicting the QoS value. The memory-based CFs [18,22] are the most simplistic methods by design for QoS prediction. However, they fail to achieve a desirable prediction accuracy due to many challenges, including the data sparsity, scalability, cold-start, etc. Model-based CFs (e.g., matrix factorization [11], factorization machines [14], deep-learning-based [21] approaches) and a few hybrid methods [2,4] combining memory-based and model-based CF have been introduced to address the above challenges. Although these methods are able to achieve satisfactory performance, however, most of them are not suitable for the real-time system due to their slower responsiveness [2,4]. On the other hand, the methods with faster responsiveness have significantly low prediction accuracy [18]. A clear trade-off between the prediction time and prediction accuracy has been observed in the literature [6]. Some recent papers attempted to address this issue [3]. However, due to the absence of quantitative measurement of prediction time to designate an algorithm to be a real-time one, these works are yet to be faster enough to be chosen as a real-time algorithm.

Another important observation is that most of the conventional CF-based methods consider the influence of every user/service to predict the QoS of a target user-service pair. However, all the users/services present in the QoS log are not trustworthy because they may have very different QoS invocation patterns compared to the rest. These users/services are generally referred to as grey-sheep [8]. The influence of grey-sheep users/services in the computation of the QoS of non-grey-sheep users/services lead to highly inaccurate results. Avoiding untrustworthy users/services could improve the QoS prediction accuracy. Li et al. [10] proposed a reputation algorithm for detecting trustworthy users based on geographical location, which could result in the inappropriate set of untrustworthy users since geographical distance may not be equivalent to network-wise distance. The authors in [17] proposed a two-phase K-means clustering-based credibility-aware QoS prediction method, where a cluster with a minimum number of users is considered untrustworthy. However, the clusters with minimum number of users can still be large set. The authors in [12,15] provided a similar approach for detecting grey-sheep using 3σ rule. Although these methods were proposed to detect the untrustworthy users/services, the notion of trustworthiness, however, is yet to be standardized.

In this paper, we propose a real-time, trust-aware QoS prediction algorithm using graph-based learning that can achieve reasonably high prediction accuracy. The graph has been established as a functional data structure that can explore higher-order connectivity (i.e., depth of relationship) in the non-euclidean data space, which helps the graph exploit every possible relationship from nodes and edges. In recent years, graph neural network (GNN) [9] has attracted vast research attention. However, to the best of our knowledge, QoS prediction using graph-based learning is mostly unexplored in the literature. We now summarize the major **contributions** of our paper:

(i) We propose a novel framework for real-time QoS prediction (TRQP) utilizing a graph convolution network that captures the multi-hop collaborative signal

by the node message passing and aggregating them over the users/services. The comparative experimental analysis with ablation study shows the efficiency of TRQP in terms of prediction accuracy and prediction time.

(ii) We propose an effective method for identifying trustworthy users/services. Our analysis shows that TRQP improved the accuracy for the non-gray-sheep users/services.

(iii) We performed extensive experiments on publicly available WS-DREAM-1 dataset [23] to validate the performance of TRQP.

In subsequent sections, we formulate the problem and discuss our solution.

2 Formulation of QoS Prediction Problem

Given a set of n users $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ with their contextual data (consisting of latitude, longitude, country, and autonomous systems), a set of m services $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ with the set of contextual data (includes latitude, longitude, country, service provider), and their partial interactions in terms of a QoS parameter q given in the form of a sparse QoS log matrix $\mathcal{Q}$, the objective of the QoS prediction problem is to predict the value of the QoS parameter for a target user-service pair, where each valid entry of $\mathcal{Q}$ (say, q_{ij}) represents the value of q of $s_j \in \mathcal{S}$ when invoked by $u_i \in \mathcal{U}$.

The conventional collaborative filtering (CF) based approaches fail to achieve high accuracy due to the presence of grey-sheep users/services [8]. Since grey users/services have their unique QoS invocation patterns, predicting the QoS for the non-grey-sheep users/services with the help of the QoS patterns of grey-sheep users/services results in a high prediction error.

The main objective of this paper is to identify the grey-sheep users/services from the given set of users and services and come up with a prediction framework that not only provides a high prediction accuracy but also has a lower prediction time to make the framework compatible with a real-time system.

3 Proposed QoS Prediction Framework

In this section, we discuss different modules of our solution framework (namely, TRQP) for trust-aware QoS prediction problem.

3.1 Identification of Trustworthy Users and Services

The first component of TRQP focuses on identifying the grey-sheep users/services. We first compute an abnormality score of each user u_i and a service s_j (say, $\mathscr{A}(u_i)/\mathscr{A}(s_j)$) as described in [8]. A user u_i (or service s_i) is considered to be more trustworthy as compared to another user u_j (or service s_j) if $\mathscr{A}(u_i)$ is less than $\mathscr{A}(u_j)$ (or, $\mathscr{A}(s_i) < \mathscr{A}(s_j)$). A user $u_i \in \mathcal{U}$ (service $s_j \in \mathcal{S}$) is called grey-sheep, if $\mathscr{A}(u_i)$ ($\mathscr{A}(s_j)$) is more than a given threshold $\mathcal{T}^u_{\mathscr{A}}$ ($\mathcal{T}^s_{\mathscr{A}}$). $\mathcal{T}^u_{\mathscr{A}}$ and $\mathcal{T}^s_{\mathscr{A}}$ are hyper-parameters, required to be set externally. In this

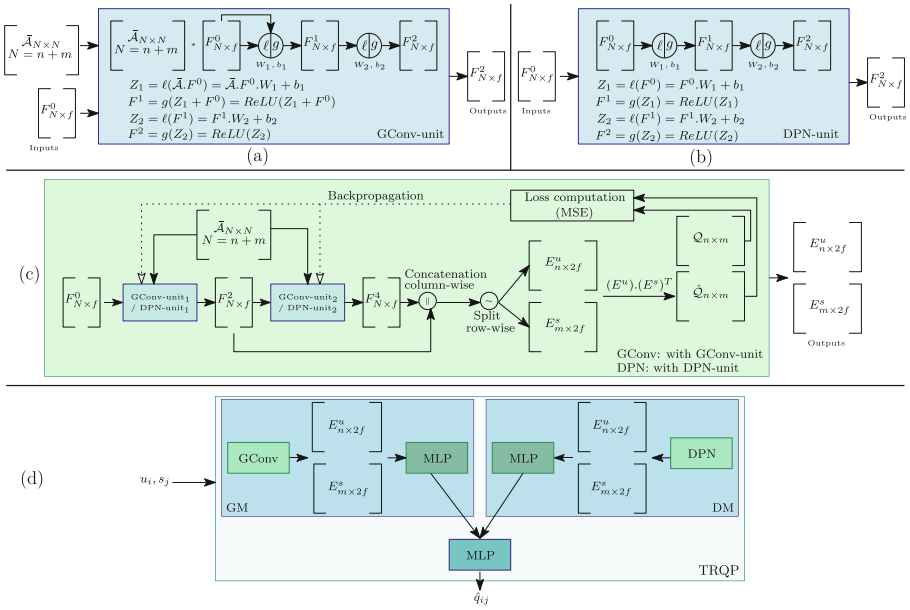

Fig. 1. Model architecture for TRQP

paper, we consider $\mathcal{T}^u_{\mathscr{A}} = \mu^u_{\mathscr{A}} + c*\sigma^u_{\mathscr{A}}$ ($\mathcal{T}^s_{\mathscr{A}} = \mu^s_{\mathscr{A}} + c*\sigma^s_{\mathscr{A}}$), where $\mu^u_{\mathscr{A}}$ and $\sigma^u_{\mathscr{A}}$ ($\mu^s_{\mathscr{A}}$ and $\sigma^s_{\mathscr{A}}$) are the mean and standard deviation of the abnormality scores of users (services), respectively. c is a hyper-parameter, which can be tuned externally. Once the grey-sheep users and services are identified, we remove them from the given list of users and services to train our model for prediction. In the next subsection, we present the details of our model.

3.2 Design of the Learning Framework for TRQP

In this subsection, we discuss our proposed architecture (designated as, TRQP) for QoS prediction. TRQP is an ensemble learning model combining two networks (i.e., GM and DM), as shown in Fig. 1. Each of these networks includes two separate modules. The first module is accountable for computing user/service feature embedding, the second module is responsible for QoS prediction.

GM consists of a graph convolution network [9] (say, GConv) for obtaining user/service feature embedding followed by a multi-layer perceptron (MLP) for QoS prediction, where DM comprises a deep prediction framework (say, DPN) for generating user/service feature embedding followed by an MLP for prediction. A final MLP is used in TRQP to aggregate the outputs of GM and DM for the final prediction. We now elaborate each of these modules below.

3.2.1 Architecture of GM. Here, we introduce the graph modeling for the prediction problem. We begin with defining the QoS prediction graph.

Definition 1 (QoS Prediction Graph (QPG)). A QoS prediction graph $G = (V_1 \cup V_2, E)$ is a bipartite graph, where the vertices V_1 and V_2 represent the set of users $\mathcal{U}$ and the set of services $\mathcal{S}$, respectively. An edge $e_{ij} = (v_i^1, v_j^2) \in E$ exists between two vertices $v_i^1 \in V_1$ and $v_j^2 \in V_2$ if the QoS log $\mathcal{Q}$ includes a valid QoS entry for the service s_j corresponding to v_j^2 invoked by the user u_i corresponding to v_i^1. ■

We represent the QPG in terms of an adjacency matrix $\mathcal{A}^{(n+m)\times(n+m)}$, which is used in graph convolutional network to obtain user/service feature embedding. However, instead of using $\mathcal{A}$ by itself, we normalize $\mathcal{A}$, so that more influence of the higher degree nodes can be avoided during learning [9]. Moreover, normalization helps in scaling. The normalized matrix is denoted by $\bar{\mathcal{A}} = \mathcal{D}^{-\frac{1}{2}}\mathcal{A}\mathcal{D}^{-\frac{1}{2}}$, where $\mathcal{D}$ is a diagonal matrix representing the degree of each node of the QPG G by its diagonal elements. It may be noted, each non-zero element of $\bar{\mathcal{A}}$, i.e., $\bar{\mathcal{A}}(i,j)$, is normalized by the square root of the number of invocations of the corresponding user u_i and service s_j as recorded in $\mathcal{Q}$, i.e., $\bar{\mathcal{A}}(i,j) = \frac{\mathcal{A}(i,j)}{\sqrt{d_{ii}}.\sqrt{d_{jj}}}$. Each node of QPG is associated with an embedding representing the features of that node (i.e., initial user/service feature embedding consisting of the latent representation of QoS profile and contextual data of user/service of length f).

3.2.2 Description of GConv: We now discuss the architecture of GConv, as shown in Fig. 1(c). We begin with illustrating the primary component of GConv, i.e., GConv-unit, as presented in Fig. 1(a). GConv-unit takes the normalized adjacency matrix $\bar{\mathcal{A}}_{N\times N}$ and an input feature matrix F^i with dimension $N \times f$, where $N = (n + m)$. The objective of a GConv-unit is to accumulate the input feature embedding of each node $v_i^k \in (V_1 \cup V_2), k \in \{1, 2\}$ of G with the feature embedding of its subsequent hop (i.e., the node directly connected to v_i^k through an edge in E of G), as modeled by the four equations of Fig. 1(a). Therefore, the output of the GConv-unit is another feature matrix of the same dimension.

The user/service embedding matrix $\mathcal{E}$ serves as the input feature embedding matrix for GConv, i.e., $F^0 = \mathcal{E}$. The initial node embedding for each node $v_i^k \in (V_1 \cup V_2), k \in \{1, 2\}$ of G is refined while propagated through multiple GConv-units by accumulating the features of other nodes directly/indirectly connected to v_i^k via a path in G. Therefore, in a GConv network with L number of GConv-units, the final embedding for each node in QPG is able to aggregate the feature embedding of all neighbors reachable through L-hops.

3.2.3 Description of MLP of GM: An MLP is used for QoS prediction in GM. The network is trained with a sample for each $u_i \in \mathcal{U}$ and $s_j \in \mathcal{S}$ such that $q_{ij} \neq 0$ in $\mathcal{Q}$. The concatenation of the features of u_i and s_j obtained from GConv is used as the feature to train the MLP, while q_{ij} is served as the target value. It may be noted, the MLP is trained before the deployment of TRQP.

3.2.4 Architecture of DM: The architecture of DM is similar to GM. Here, instead of GConv, a deep prediction network (DPN) comprising DPN-unit (refers

to Fig. 1(b)) is used to generate the user/service embedding, while a following MLP is used for QoS prediction. The architecture of DPN is similar to GConv. The only difference in DPN is that the adjacency matrix is not used in DPN.

3.2.5 Architecture of TRQP: As discussed earlier, TRQP comprises GM, DM, and an additional MLP to combine the outputs of GM and DM (Refer to Fig. 1(d)). The MLP is trained before the deployment of TRQP as well. Each training sample of MLP consists of a feature vector of size 2, the outputs of GM and DM, for each $u_i \in \mathcal{U}$ and $s_j \in \mathcal{S}$ such that $q_{ij} \neq 0$ in $\mathcal{Q}$. q_{ij} is again served as the target value. The output of the MLP is considered the final output of TRQP.

Finally, we employ an outlier detection algorithm [3] for detecting outliers from the dataset, which are removed to measure the performance of TRQP. The next section presents the performance of TRQP through experiments.

4 Experimental Analysis

We have implemented our proposed method in TensorFlow with Python. The training of TRQP was done on NVIDIA's Quadro RTX 3000/PCIe/SSE2 GPU with 1920 cores, and 6 GB memory. For testing, we used i9-10885H @ 2.40 GHz×16 processor with x86_64 CPU with 128 GB RAM.

To validate the performance of TRQP, we performed extensive experiments on WS-DREAM-1 dataset [23]. Table 1 shows the description of the dataset used for our experiment.

Table 1. WS-Dream-1 dataset description

QoS	(# user, # service)	Min	Max	Mean	Median	Std. Dev
Response time (RT)	(339×4998)	0.001	19.999	0.915	0.319	2.000
Throughput (TP)	(339×5004)	0.017	1000.0	46.786	14.018	108.918

The configuration of TRQP, used for our experiments, is as follows. For identifying grey-sheep users/services, we used $c = 2$ throughout our experiments. We reported our results by eliminating 3% outliers. The size of initial user/service feature embedding is 255. Our GConv and DPN includes 2 GConv-units and 2 DPN-units, respectively. In our experiment, we have used ADAM optimizer and mean squared error as the loss function [7].

4.1 Experimental Analysis

We compared the performance of TRQP with 14 major state-of-the-art (SoA) methods with and without trustworthiness taken into consideration. Tables 2(a)

and (b) show the comparative analysis of TRQP in terms of the prediction accuracy, measured by mean absolute error (MAE) [3]. Figure 2(a) shows the comparative study of TRQP in terms of the prediction time. Below we summarize our observations from Tables 2(a), (b) and Fig. 2(a).

(i) In all cases other than OffDQ, TRQP outperformed the SoA for both training percentages for both datasets. The improvement of TRQP over the second-best value for each of the 4 cases is shown in the final row of Table 2(b).

(ii) Although OffDQ performed better than TRQP in terms of the prediction accuracy, the results of the OffDQ was presented by removing 5% to 15% outliers. However, in our case, we removed 3% outliers and about 10% entries due to grey-sheep analysis. Moreover, TRQP performed better than OffDQ in terms of the prediction time (refer to Fig. 2(a)). While the prediction time for OffDQ is in the order of 10^{-1} s, the same for the TRQP is in the order of 10^{-5} s.

Table 2. Comparison of TRQP with SoA on prediction accuracy (MAE)

Without Trust-aware Prediction				
Methods	Response time		Throughput	
	10%	20%	10%	20%
WSRec [22]	0.6394	0.5024	19.9754	16.0762
NRCF [16]	0.5312	0.4607	-	-
RACF [18]	0.4937	0.4208	-	-
GMF [1]	0.4737	0.4233	-	-
DAFR [21]	0.3461	0.3404	16.9020	15.5670
LBFM [20]	0.3750	0.3421	-	11.9291
CNCF [5]	0.3380	0.3140	18.189	16.826
OffDQ [3]	0.2000	0.1800	9.1600	8.6700
TRQP	**0.2540**	**0.2520**	**10.5760**	**9.5660**

(a)

With Trust-aware Prediction				
Methods	Response time		Throughput	
	10%	20%	10%	20%
TAP [15]	0.5502	-	-	-
RAP [13]	0.5250	0.4400	19.4333	16.4104
CAP [17]	0.5030	0.4394	15.1148	13.8192
RMF [19]	0.4877	0.4414	-	-
LRMF [10]	0.4719	0.4384	-	-
S-RAP [12]	0.4833	-	-	-
TRQP	**0.2540**	**0.2520**	**10.5760**	**9.5660**
Improvement	24.85%	19.75%	30.03%	19.81%

(b)

(iii) One of the crucial characteristics of a real-time QoS prediction algorithm is that it is supposed to have negligible prediction time compared to the service's response time. This makes the prediction framework compatible with a real-time recommendation system, where a service is first recommended based on its predicted QoS, before its execution. Therefore, one preliminary criterion of a real-time prediction algorithm is to have a much lesser prediction time compared to the response time of services. As observed in Table 1, the minimum response time of service is in order of 10^{-3} s. In comparison to the response time of services, our framework has an insignificant prediction time (i.e., in the order of 10^{-5} s), which makes TRQP a real-time algorithm.

(iv) Furthermore, TRQP outperformed the SoA methods that are known to have less prediction time.

Ablation Study: From this analysis onwards we have used the RT dataset with 10% training data. Figures 2(b) and (c) present the results for our ablation study. Our observations from Figs. 2(b) and (c) are listed below:

(i) Figure 2(b) shows the model ablation study, where we reported the performance of the individual components of TRQP. As evident from Fig. 2(b), TRQP performed the best in the presence of all its components.

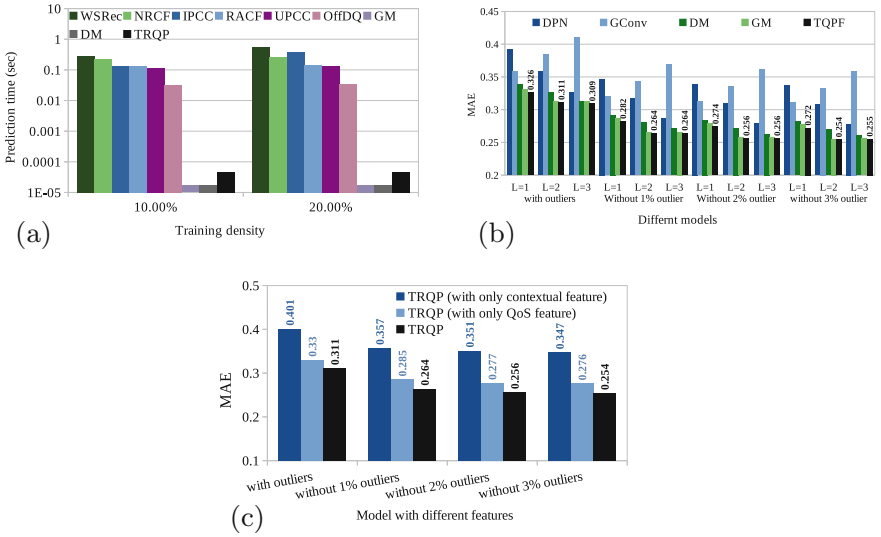

Fig. 2. (a) Comparison of TRQP with respect to SoA on prediction time; Ablation study (b) Model ablation; (c) Feature ablation

(ii) As observed in Fig. 2(a), GM and DM have better prediction times as compared to TRQP. However, TRQP performed better than GM and DM in terms of prediction accuracy.

(iii) We reported the performance of all the networks for $L = 1, 2, 3$ (i.e., GConv/DPN with 1/2/3 GConv-units/DPN-units). For $L = 2$ and $L = 3$, the performance of TRQP is almost the same and better than the performance for $L = 1$. This may be due to the over-smoothing problem in graph convolution network [9]. GConv with more number of GConv-units cannot improve its performance, and it often leads to severe degradation in the feature extraction since it may end up obtaining similar embedding for all the nodes in QPG. In our experiment, we used 2 GConv-units.

(iv) Figure 2(c) shows the feature ablation study. We reported the performance of TRQP with only contextual features, only QoS features, and their combinations. As it turned out, TRQP, with the combination of the contextual and QoS features, performed the best as compared to the others. The performance of TRQP with only the contextual feature was not good. TRQP achieved a 25.6% improvement on average over TRQP with only contextual features. However, in the absence of contextual features, we can still use TRQP with only QoS features for the prediction, as TRQP achieved only a 7.1% improvement on average over TRQP with only QoS features.

(v) Furthermore, we observed the overall MAE for the non-grey-sheep users or services obtained from TRQP is 0.254, which is less than the one obtained from TRQP with grey-sheep users or services (which is 0.261 as reported). This, in turn, shows the effectiveness of our trustworthiness analysis.

In summary, TRQP without grey-sheep users and services achieved reasonably high prediction accuracy while being suitable for a real-time system.

5 Conclusion

This paper proposes a trust-aware, real-time QoS prediction framework. To the best of our knowledge, TRQP is one of the first methods in the QoS prediction literature to leverage the graph-based feature embedding exploiting the graph convolution for QoS prediction. The graph convolution over bipartite representation of QoS data helps exploit the non-linear, deep/higher-order, and complex relationship among user/service QoS data that enhances the collaborative signal for better QoS prediction. We also propose a means to determine trustworthy users/services. Focusing on the trustworthiness problem, identifying the grey-sheep users/services, and removing them to achieve better prediction accuracy proves the usefulness of our framework for trust-aware QoS prediction. The experimental analysis in the paper shows that TRQP outperformed major SoA methods in terms of prediction accuracy and/or prediction time.

As a future endeavor, we wish to develop more sophisticated algorithms for predicting the QoS for untrustworthy users/services. We also aim to explore a Spatio-temporal graph convolution for time-aware QoS prediction.

References

1. Chang, Z., Ding, D., Xia, Y.: A graph-based QoS prediction approach for web service recommendation. Appl. Intell. **51**(10), 6728–6742 (2021)
2. Chattopadhyay, S., Banerjee, A.: QoS value prediction using a combination of filtering method and neural network regression. In: ICSOC, pp. 135–150 (2019)
3. Chattopadhyay, S., et al.: OffDQ: an offline deep learning framework for QoS prediction. In: ACM the Web Conference, pp. 1987–1996. WWW (2022)
4. Chowdhury, R.R., et al.: CAHPHF: context-aware hierarchical Qos prediction with hybrid filtering. IEEE Trans. Serv. Comput. **15**(4), 2232–2247 (2022)
5. Gao, H.: Xothers: context-aware QoS prediction with neural collaborative filtering for internet-of-things services. IEEE IoT J. **7**(5), 4532–4542 (2019)
6. Ghafouri, S.H., et al.: A survey on web service QoS prediction methods. IEEE TSC (2020). https://doi.org/10.1109/TSC.2020.2980793
7. Goodfellow, I.J., Bengio, Y., Courville, A.C.: Deep Learning. Adaptive Computation and Machine Learning. MIT Press, London (2016)
8. Gras, B., et al.: Identifying grey sheep users in collaborative filtering: a distribution-based technique. In: ACM UMAP, pp. 17–26 (2016)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR 2017 (2017)

10. Li, S., Wen, J., Wang, X.: From reputation perspective: A hybrid matrix factorization for QoS prediction in location-aware mobile service recommendation system. Mob. Inf. Syst. **2019**(8950508), 1–12 (2019)
11. Lo, W., et al.: An extended matrix factorization approach for QoS prediction in service selection. In: IEEE SCC, pp. 162–169 (2012)
12. Muslim, H.S.M., et al.: S-RAP: relevance-aware QoS prediction in web-services and user contexts. Knowl. Inf. Syst. **64**, 1997–2022 (2022)
13. Qiu, W., et al.: Reputation-aware QoS Value prediction of web services. In: IEEE SCC, pp. 41–48 (2013)
14. Shen, L., et al.: Contexts Enhance accuracy: on modeling context aware deep factorization machine for web API QoS prediction. IEEE Access **8**, 165551–165569 (2020)
15. Su, K., et al.: TAP: a personalized trust-aware QoS prediction approach for web service recommendation. Knowl.-Based Syst. **115**, 55–65 (2017)
16. Sun, H., et al.: Personalized web service recommendation via normal recovery collaborative filtering. IEEE TSC **6**(4), 573–579 (2013)
17. Wu, C., et al.: QoS prediction of web services based on two-phase k-means clustering. In: IEEE ICWS. pp. 161–168 (2015)
18. Wu, X., et al.: Collaborative Filtering Service Recommendation based on a Novel Similarity Computation Method. IEEE Trans. Serv. Comput. **10**(3), 352–365 (2017)
19. Xu, J., et al.: Web service personalized quality of service prediction via reputation-based matrix factorization. IEEE Trans. Reliab. **65**(1), 28–37 (2016)
20. Yang, Y., et al.: A location-based factorization machine model for web service QoS prediction. IEEE Trans. Serv. Comput. **14**(5), 1264–1277 (2021)
21. Yin, Y., et al.: QoS prediction for service recommendation with features learning in mobile edge computing environment. IEEE Trans. Cogn. Commun. Netw. **6**(4), 1136–1145 (2020)
22. Zheng, Z., et al.: QoS-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**(2), 140–152 (2011)
23. Zheng, Z., et al.: Investigating QoS of Real-World Web Services. IEEE Trans. Serv. Comput. **7**(1), 32–39 (2014)
24. Zheng, Z., et al.: Web service QoS prediction via collaborative filtering: a survey. IEEE Trans. Serv. Comput. (2020). https://doi.org/10.1109/TSC.2020.2995571