

RAHN: A Reputation Based Hourglass Network for Web Service QoS Prediction

Xia Chen^{1,2}, Yugen Du^{1,2*}, Guoxing Tang^{1,2}, Yingwei Luo^{1,2} and Benchi Ma^{1,2}

¹*Software Engineering Institute, East China Normal University, Shanghai 200062, China*

²*Shanghai Key Laboratory of Trustworthy Computing, Shanghai 200062, China*

Email: {chenxia200062}@gmail.com, {ygdu}@sei.ecnu.edu.cn

Abstract—As the homogenization of Web services becomes more and more common, the difficulty of service recommendation is gradually increasing. How to predict Quality of Service (QoS) more efficiently and accurately becomes an important challenge for service recommendation. Considering the excellent role of reputation and deep learning (DL) techniques in the field of QoS prediction, we propose a reputation and DL based QoS prediction network, RAHN, which contains the Reputation Calculation Module (RCM), the Latent Feature Extraction Module (LFEM), and the QoS Prediction Hourglass Network (QPHN). RCM obtains the user reputation and the service reputation by using a clustering algorithm and a Logit model. LFEM extracts latent features from known information to form an initial latent feature vector. QPHN aggregates latent feature vectors with different scales by using Attention Mechanism, and can be stacked multiple times to obtain the final latent feature vector for prediction. We evaluate RAHN on a real QoS dataset. The experimental results show that the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) of RAHN are smaller than the six baseline methods.

Index Terms—Web service, QoS prediction, reputation, deep learning

I. INTRODUCTION

The number of Web services is increasing day by day. The emergence of homogenization causes the process of finding the most suitable Web service to become more difficult for users. In general, QoS are susceptible to network fluctuations, hardware failures, service errors, etc., and do not represent the service invocation experience of users under normal conditions [1]. Therefore, if we want to recommend the most suitable Web service for each user, we need to obtain the corresponding QoS information in advance. Since obtaining the QoS information corresponding to all the Web services requires a lot of time and resource costs. Thus, researchers consider QoS prediction as a key method to obtain QoS information and have conducted extensive and in-depth research on it.

Nowadays, MF and DL have been widely recognized as the most general model-based QoS prediction methods. Since the reliability of historical QoS information can have a significant impact on MF-based QoS prediction, the effectiveness of QoS information can be affected by unreliable users and services. Therefore many researchers have tried to use reputation to quantify the reliability of users and services. With the development and popularization of DL technology in CV, and NLP,

some researchers also try to apply it to QoS prediction and confirm the importance of DL technology in QoS prediction.

After combining reputation and DL technology, this paper proposes an hourglass QoS prediction method based on reputation and DL, RAHN. Due to the arbitrariness of some users in providing QoS observations and the inherent instability of some Web services, it is necessary to take the user reputation and service reputation into consideration.

In summary, the main contributions of this paper can be summarized as follows: **1)** We propose a QoS prediction network RAHN, which contains RCM, LFEM, and QPHN. **2)** RCM can calculate user reputation and service reputation, which makes RAHN better robust to unreliable users and unreliable Web services. **3)** LFEM can extract latent features from the information and reputation of users and Web services. **4)** QPHN can mine high-level features and aggregate latent features at different scales to build better latent feature vector. **5)** Multiple experiments were conducted on the dataset containing a large amount of real QoS data to verify the effectiveness and superiority of RAHN.

II. RELATED WORK

A. MF-based QoS prediction

MF is a method that can fully utilize sparse QoS information for QoS prediction. Since its development, many researchers have proposed their own thinking for it. Zhong et al. [2] proposed Network Bias MF (NBMF) considering that different users may have different degrees of delay when invoking Web services, and achieved better prediction performance. Chen et al. [3] used Dirichlet distribution to compute the user reputation and achieved ideal results.

These works have demonstrated the important support of reliability for QoS prediction. Therefore, this paper incorporates both user reliability and service reliability into the thinking and calculates user and service reputation to assist QoS prediction.

B. DL-based QoS prediction

Generally speaking, DL models can utilize complex user information and Web service information more effectively to improve QoS prediction accuracy. Peng et al. [4] proposed a web service link prediction method based on topic-aware heterogeneous graph neural networks. It captures fine-grained topic-aware semantics while mining contextual topic distribution to achieve better prediction results. Jia et al. [5] overcomes

QoS data sparsity by fusing local and global location information of users and Web services in the interaction layer, using MLP to obtain high-dimensional nonlinear relationships and combining them with low-dimensional linear relationships.

All of the above works have shown that DL techniques have great potential in QoS prediction tasks and can lead to even better QoS prediction performance. Inspired by them, this paper designs RCM, DL based LFEM and DL based QPHN to improve the QoS prediction accuracy.

III. PRELIMINARIES

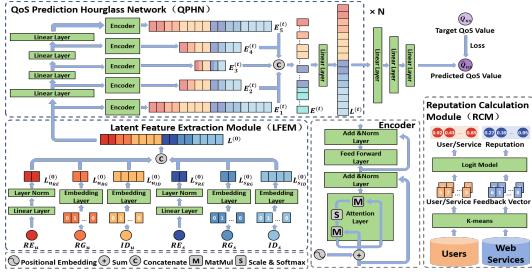


Fig. 1. The Overall Architecture of RAHN. 1) **RCM**: We use the K-means algorithm to cluster users and Web services into clusters, and then count the user feedback vectors and service feedback vectors according to the 3σ rule in the normal distribution. Then we apply Logit model to calculate user reputation and service reputation. 2) **LFEM**: We use Linear Layers and Embedding Layers to extract user latent features from user information and user reputation. And in the same way we extract Web service latent features. Then, user latent features and service latent features are concatenated together to form the final latent feature vector. 3) **QPHN**: We process the latent feature vectors using Linear Layers and Encoders at different scales to obtain multi-scale feature vectors with high-level feature information. These latent feature vectors are then concatenated and multiple QPHNs can be stacked for use.

A. Problem Formulation

In this section some necessary definitions will be given:

- **User Reputation (in Eq.(6))**: Certain users are casual and heavily subjective in submitting QoS observations. User reputation measures how trustworthy a user is.
- **Service Reputation (in Eq.(6))**: Some Web services may have QoS fluctuations due to their own bugs. Service reputation measures how stable a Web service is in providing QoS.
- **NPEd**: Parameter combinations in the experiment, e.g. NPEd=0108 denotes N=0, PE=1, d=08. It will be mentioned in the Fig.2 and 3 in section V.

B. Method Overview

Fig.1 illustrates the overview architecture of RAHN and individual modules. The details of these modules will be shown in Section IV.

IV. PROPOSED APPROACH

A. RCM

RCM can calculate the user reputation and service reputation. We use the K-means clustering algorithm to cluster users and Web services separately to obtain N_u user clusters and N_s service clusters. According to [6], we consider the

cluster containing the most elements as the reliable cluster. The specific definition formulas are as follows: $U_r = \{u|u \in C_u^k, x = argmax |C_u^k|, 1 \leq k \leq N_u\}$, $S_r = \{s|s \in C_s^k, x = argmax |C_s^k|, 1 \leq k \leq N_s\}$, where U_r denotes the set of reliable users, C_u^k denotes the k -th user cluster, N_u denotes the total number of user clusters, S_r denotes the set of reliable services, C_s^k denotes the k -th service cluster, N_s denotes the total number of service clusters, and $|C^k|$ denotes the number of elements in the k -th cluster.

Based on the work of [7], we believe that reliable clusters reflect what the public perceives as normal QoS observations, and that the QoS values should follow a normal distribution $N(\mu, \sigma^2)$ (μ is the mean and σ is the standard deviation). Based on the reliability clusters, we classify all QoS observations into two types: positive and negative feedback. QoS observations that are close to the normal value are positive feedback and vice versa. According to the 3σ rule, we consider QoS observations that are within the interval $(\mu_r - 3\sigma_r, \mu_r + 3\sigma_r)$ to be positive feedback, and vice versa to be negative feedback. Both μ_r and σ_r come from reliable clusters. The feedback vector can then be obtained from $F = [po, ne]$, where po indicates the amount of positive feedback and ne indicates the amount of negative feedback.

Suppose, the probability and utility of the user providing positive and negative feedback are p_1, p_2, U_1, U_2 , as shown in Eq.(1). According to the principle of utility maximization in economics, the user will choose the option with greater utility. Thus p_1, p_2 are defined as follows:

$$U_1 = V_1 + \epsilon_1, U_2 = V_2 + \epsilon_2, \quad (1)$$

$$p_1 = P(U_1 > U_2), p_2 = P(U_1 < U_2), \quad (2)$$

where the V_1 and V_2 denote the observable deterministic parts, and ϵ_1 and ϵ_2 denote the random terms.

Assume that ϵ_1 and ϵ_2 follow the standard Gumbel distribution and are independent of each other. Then $\epsilon_1 - \epsilon_2$ will satisfy the *Logistic*(0, 1). Its distribution function is as follows:

$$F(x) = P(X \leq x) = \frac{1}{1 + e^{-(x-\mu)/\gamma}}, (\mu = 0, \gamma = 1). \quad (3)$$

Assuming $V = \beta X$, then p_1 and p_2 can be transformed into:

$$p_1 = F(V_1 - V_2) = \frac{e^{\beta X_1}}{e^{\beta X_1} + e^{\beta X_2}} = \frac{e^{\beta po}}{e^{\beta po} + e^{\beta ne}}, \quad (4)$$

$$p_2 = F(V_2 - V_1) = \frac{e^{\beta X_2}}{e^{\beta X_1} + e^{\beta X_2}} = \frac{e^{\beta ne}}{e^{\beta po} + e^{\beta ne}}, \quad (5)$$

where β is the positive coefficient of X . Finally, reputation can be calculated from both positive and negative feedback:

$$Re = p_1 / (p_1 + p_2), \quad (6)$$

where Re denotes reputation and the value is in the range [0,1].

B. LFEM

LFEM can obtain an initial latent feature vector $L^{(0)}$. Each user or Web service has its own information (ID, region (RG) and previously obtained reputation). We use Linear Layer to

extract the hidden information in the reputation and obtain the feature vectors $L_{u_{RE}}^{(0)}$, $L_{s_{RE}}^{(0)}$. Then we transform ID and RG into one-hot vectors [8] respectively, and obtain the links in them by Embedding Layer to generate feature vectors $L_{u_{RG}}^{(0)}$, $L_{u_{ID}}^{(0)}$, $L_{s_{RG}}^{(0)}$, $L_{s_{ID}}^{(0)}$. These feature vectors will be concatenated to obtain $L^{(0)}$, which is represented as follows:

$$L^{(0)} = L_{u_{RE}}^{(0)} \oplus L_{u_{ID}}^{(0)} \oplus L_{u_{RG}}^{(0)} \oplus L_{s_{RE}}^{(0)} \oplus L_{s_{ID}}^{(0)} \oplus L_{s_{RG}}^{(0)}, \quad (7)$$

where \oplus denotes the concatenate operation, $L_{RE}^{(0)} \in \mathbb{R}^{1 \times (d/4)}$, $L_{RG}^{(0)} \in \mathbb{R}^{1 \times (d/2)}$, $L_{ID}^{(0)} \in \mathbb{R}^{1 \times (d/4)}$ and d is a positive integer divisible by 4. $L^{(0)} \in \mathbb{R}^{1 \times 2d}$, which blends the user with the latent features in the Web service.

C. QPHN

QPHNs can obtain latent feature vectors at multiple scales, and can be repeatedly stacked. We take $L^{(0)}$ as the initial input to the QPHNs and consider the processing of a QPHN as a function $\Phi(x)$ for obtaining the final latent feature vector $L^{(n)}$. The formula is shown below:

$$L^{(n)} = \Phi_n(\Phi_{n-1} \dots (\Phi_1(L^{(0)}))), \quad (8)$$

where $n > 0$ denotes the number of stacked layers of the QPHN, and by n iterations, we transform $L^{(0)}$ to $L^{(n)}$.

For the t -th layer QPHN, its input is $L^{(t-1)}$. We use Linear Layers of different sizes to obtain multi-scale latent feature vectors:

$$L_1^{(t)} = L^{(t-1)}, L_k^{(t)} = \text{ReLU}(g_{k-1}(L_{k-1}^{(t)})), \quad (9)$$

where $L_1^{(t)} \in \mathbb{R}^{1 \times 2d}$, $L_2^{(t)} \in \mathbb{R}^{1 \times d}$, $L_3^{(t)} \in \mathbb{R}^{1 \times (d/2)}$, $L_4^{(t)} \in \mathbb{R}^{1 \times d}$, $L_5^{(t)} \in \mathbb{R}^{1 \times 2d}$, denote the latent feature vectors of different scales, respectively. $k=2,3,4,5$. ReLU [9] is the activation function that we use. $f^{[a,b]}(x)$ denotes the linear layer, with a being the dimension of the input x and b being the output dimension. $g_1=f^{[2d,d]}, g_2=f^{[d,d/2]}, g_3=f^{[d/2,d]}, g_4=f^{[d,2d]}$.

Then, Encoders with different scales are used to extract high-level features in $L_i^{(t)}$ and form the new latent feature vector $E_i^{(t)} = \text{Encoder}(L_i^{(t)})$, where the structure of Encoder is shown in Fig.1 and $i \in [1, 5]$. We utilize the attention mechanism to focus on the key information in $L_i^{(t)}$ at different scales to obtain a more effective latent feature vector $E_i^{(t)}$. Next, we concatenate $E_i^{(t)}$ to get $E^{(t)}$ and adjust its dimension to obtain the same size as the input $L^{(t-1)}$ to get $L^{(t)}$:

$$E^{(t)} = E_1^{(t)} \oplus E_2^{(t)} \oplus E_3^{(t)} \oplus E_4^{(t)} \oplus E_5^{(t)}, \quad (10)$$

$$L^{(t)} = f^{[13d/2, 2d]}(E^{(t)}), \quad (11)$$

where $E^{(t)} \in \mathbb{R}^{1 \times (13d/2)}$ aggregates multi-scale latent feature vectors, containing both low-level feature information and high-level feature information.

D. QoS Prediction and Model Training

With the previous modules we obtained the final latent feature vector $L^{(n)}$. We get the predicted QoS value \hat{Q}_{us} by three Linear Layers which can be represented as: $\hat{Q}_{us} = f^{[d/2, 1]}(\text{ReLU}(f^{[d,d/2]}(\text{ReLU}(f^{[2d,d]}(L^{(n)}))))$, where \hat{Q}_{us}

denotes the QoS prediction result when user u invokes Web service s .

RAHN is trained by loss function, and our goal is to minimize the loss function J , which can be expressed as:

$J = \frac{1}{N} \sum_{k=1}^N |Pred(x_k, \Theta) - Q_{x_k}| + \lambda \mathbb{L}_{reg}(\Theta)$, where N is the batch size, x_k denotes the k -th input (containing information about the corresponding user and Web service), Θ denotes all the parameters to be learned, $Pred$ denotes the function of RAHN that maps the input x_k to the predicted QoS value, and Q_{x_k} denotes the target QoS value. In addition, $\lambda \mathbb{L}_{reg}(\Theta)$ is the regularization term, which is used to prevent the model from overfitting.

RAHN uses the Adam optimizer [10] to update all parameters to be optimized as follows: $\Theta \leftarrow \Theta - \eta \frac{\partial J}{\partial \Theta}$, where η is the learning rate, which is used to control the rate at which the model converges.

V. EXPERIMENTS

A. Dataset

This paper conducts extensive experiments on the WS-DREAM [11] dataset with a large amount of real-world QoS data. WS-DREAM dataset contains 339 users, 5,825 Web services, and 2 user-service matrices (response time matrix and throughput matrix). In this paper, we use the user-service matrix of response time to evaluate the performance of RAHN and compare it with other baseline methods.

B. Evaluation Metrics

In this paper, MAE and RMSE are used as evaluation metrics. The related definitions are as follows: $MAE = \frac{\sum |Q_{ij} - \hat{Q}_{ij}|}{N}$, $RMSE = \sqrt{\frac{\sum (Q_{ij} - \hat{Q}_{ij})^2}{N}}$, where Q_{ij} denotes the true value, \hat{Q}_{ij} denotes the predicted value, and N denotes the number of entries in the test dataset. Smaller MAE and RMSE indicate higher QoS prediction accuracy.

C. Performance Comparison

TABLE I
PERFORMANCE COMPARISON OF RESPONSE TIME (SMALLER MAE AND RMSE MEANS HIGHER ACCURACY)

Methods	MAE					RMSE					
	MD=2%	MD=4%	MD=6%	MD=8%	MD=10%	Improve	MD=2%	MD=4%	MD=6%	MD=8%	MD=10%
PMF [12]	0.327	0.255	0.242	0.239	0.238	41.73%	0.529	0.461	0.460	0.457	0.454
CMF [13]	0.223	0.203	0.197	0.161	0.150	25.10%	0.412	0.395	0.388	0.346	0.339
RLMF [3]	0.185	0.162	0.162	0.156	0.147	16.99%	0.391	0.370	0.368	0.359	0.344
NeuMF [4]	0.163	0.150	0.148	0.149	0.147	12.18%	0.366	0.358	0.356	0.354	0.353
HSA-Net [15]	0.182	0.159	0.128	0.128	0.126	8.15%	0.558	0.495	0.470	0.448	0.442
PLRes [16]	0.174	0.143	0.135	0.125	0.122	5.90%	0.524	0.461	0.424	0.409	0.407
RAHN	0.156	0.134	0.125	0.118	0.115	-	0.366	0.348	0.343	0.337	0.335

To demonstrate the effectiveness of RAHN, we compare it with six other representative baseline methods. The details are presented as shown below: **1) PMF**: This MF-based method incorporates probability. **2) CMF**: This MF-based approach takes outliers into consideration. **3) RLMF**: This MF-based method calculates the user reputation using the Dirichlet distribution. **4) NeuMF**: This DL-based method fuses MF with multi-layer perceptron. **5) HSA-Net**: This DL-based method uses hidden state awareness techniques. **6) PLRes**:

This DL-based method reutilizes the historical call probability distribution of users and services with location features.

We initialized the parameters of every baseline method according to their paper to obtain the best performance. The parameters of RAHN are set to $N_u=5$, $N_s=15$, $N=2$ (Same as n in Eq.(8)), $PE=0$ (whether to use Position Embedding, 0 is false, 1 is true), $d=16$ (in Eq.(7)).

We randomly remove the QoS data from the user-service matrix as a training matrix to study the prediction performances under different Matrix Densities (MD). A test matrix is then created using the deleted data. MD is set to $\{2\%, 4\%, 6\%, 8\%, 10\%\}$. Outliers inevitably exist in historical QoS data, and if the model's predictions coincide with these outliers, this may lead to a situation where MAE and RMSE are small but the prediction accuracy is poor. Therefore, we exclude some outliers when calculating MAE and RMSE. In this paper, we removed 10% of the most obvious outliers in the test data for all methods according to the steps in [13]. As shown in Table I, when the matrix density is set in the range of 2% to 10%, RAHN has better prediction accuracy.

D. Impact of Parameters

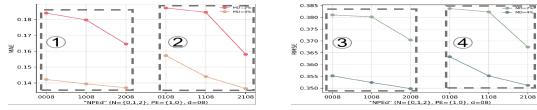


Fig. 2. Impact of N

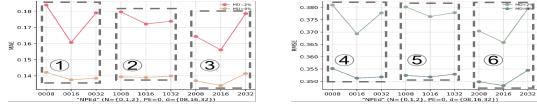


Fig. 3. Impact of d

In order to analyze how different parameter settings affect the QoS prediction of RAHN, in this paper, the same parameter settings except the parameters to be analyzed are as follows: $N_u=5$, $N_s=15$, $\eta=0.0005$. MD is set to 2% and 4%.

1) **Impact of N:** N denotes the number of QPHN. The size of N determines the network depth of RAHN. In order to explore the effect of N on the prediction results, we conducted experiments with N set to $\{0,1,2\}$ for $PE=\{1,0\}$ and $d=08$, respectively. As shown in Fig. 2, we can find from regions ① - ④ that the MAE and RMSE decrease with the increase of N. Considering the time and resource costs, we believe that RAHN predicts best at $N = 2$.

2) **Impact of d:** d denotes the user/service latent feature dimension. In order to explore the effect of d on the prediction results, we conducted experiments with d set to $\{08,16,32\}$ in the case of $N=\{0,1,2\}$ and $PE=0$, respectively. As shown in Fig. 3, we can find that both MAE and RMSE reach the minimum value at $d=16$ from regions ① - ⑥. Therefore, we believe that RAHN has the best prediction accuracy at $d = 16$.

VI. CONCLUSION

Previously, little or no work has been done to integrate reputation with DL techniques, while we propose a repu-

tation and DL based QoS prediction method, RAHN, with good predicted results. RAHN contains three modules: RCM, LFEM, and QPHN. RCM calculates user reputation and service reputation; LFEM extracts the initial latent feature vectors from reputation, region, and ID; QPHN extracts the initial latent feature vectors that incorporate different scales. To evaluate the effectiveness of RAHN, we conduct extensive experiments on real large-scale QoS datasets. Compared with the six baseline methods, RAHN reduces the MAE by an average of $5.9\% \sim 41.73\%$ and the RMSE by an average of $2.72\% \sim 23.35\%$. In future, we plan to extend RAHN as a time-aware QoS prediction based method.

REFERENCES

- [1] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, T. Zhang, Collaborative web service quality prediction via exploiting matrix factorization and network map, *IEEE Transactions on Network and Service Management* 13 (1) (2016) 126–137.
- [2] W. Zhong, Y. Du, C. Shan, H. Wang, F. Chen, Collaborative web service quality prediction via network biased matrix factorization., in: *SEKE*, 2022, pp. 418–423.
- [3] F. Chen, Y. Du, W. Zhong, H. Wang, Web service qos prediction based on reputation and location aware matrix factorization, in: *2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta)*, IEEE, 2022, pp. 1722–1729.
- [4] Q. Peng, B. Cao, X. Xie, S. Liu, G. Kang, J. Liu, Th-slp: Web service link prediction based on topic-aware heterogeneous graph neural network, in: *2023 IEEE International Conference on Web Services (ICWS)*, IEEE, 2023, pp. 465–474.
- [5] Z. Jia, L. Jin, Y. Zhang, C. Liu, K. Li, Y. Yang, Location-aware web service qos prediction via deep collaborative filtering, *IEEE Transactions on Computational Social Systems* (2022).
- [6] C. Wu, W. Qiu, Z. Zheng, X. Wang, X. Yang, Qos prediction of web services based on two-phase k-means clustering, in: *2015 ieee international conference on web services*, IEEE, 2015, pp. 161–168.
- [7] K. Su, L. Ma, B. Xiao, H. Zhang, Web service qos prediction by neighbor information combined non-negative matrix factorization, *Journal of Intelligent & Fuzzy Systems* 30 (6) (2016) 3593–3604.
- [8] Y. Yin, Z. Cao, Y. Xu, H. Gao, R. Li, Z. Mai, Qos prediction for service recommendation with features learning in mobile edge computing environment, *IEEE Transactions on Cognitive Communications and Networking* 6 (4) (2020) 1136–1145.
- [9] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.
- [10] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).
- [11] Z. Zheng, Y. Zhang, M. R. Lyu, Distributed qos evaluation for real-world web services, in: *2010 IEEE International Conference on Web Services*, IEEE, 2010, pp. 83–90.
- [12] A. Mnih, R. R. Salakhutdinov, Probabilistic matrix factorization, *Advances in neural information processing systems* 20 (2007).
- [13] F. Ye, Z. Lin, C. Chen, Z. Zheng, H. Huang, Outlier-resilient web service qos prediction, in: *Proceedings of the Web Conference 2021*, 2021, pp. 3099–3110.
- [14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.
- [15] Z. Wang, X. Zhang, M. Yan, L. Xu, D. Yang, Hsa-net: Hidden-state-aware networks for high-precision qos prediction, *IEEE Transactions on Parallel and Distributed Systems* 33 (6) (2021) 1421–1435.
- [16] W. Zhang, L. Xu, M. Yan, Z. Wang, C. Fu, A probability distribution and location-aware resnet approach for qos prediction, *Journal of Web Engineering* 20 (4) (2021) 1251–1290.