Future Generation Computer Systems 100 (2019) 46–57

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Spatio-temporal context-aware collaborative QoS prediction

Qimin Zhou [a], Hao Wu [a,*], Kun Yue [a], Ching-Hsien Hsu [b]

[a] School of Information Science and Engineering, Yunnan University, No. 2 North Green Lake Road, Kunming 650091, China
[b] Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan

HIGHLIGHTS

- We develop neural network based methods for the task of context-aware QoS prediction.
- The hierarchical structure is flexible to model the interactions of spatio-temporal features.
- We use the attention mechanism to merge spatial features into a single vector representation.
- More accurate prediction results can be obtained by optimizing the L_1 loss function with L_1 regularization.

ARTICLE INFO

Article history:
Received 30 December 2018
Received in revised form 1 April 2019
Accepted 6 May 2019
Available online 11 May 2019

Keywords:
QoS prediction
Spatio-temporal context-aware
Attention mechanism
Neural model

ABSTRACT

With the exponential growth of Web services, various collaborative QoS prediction methods have been suggested to make an efficient evaluation of quality-of-services (QoS) and assist users selecting appropriate services. It is still a technical challenge to be taken into account the impact of complex spatio-temporal contexts of service invocations and make use of their characteristics in the forecasting process. To this end, we propose two universal spatio-temporal context-aware collaborative neural models (STCA-1 and STCA-2) to make QoS prediction by considering invocation time and multiple spatial features both of service-side and user-side. Our proposed models utilize hierarchical neural networks to realize the embedding expression of original features, the generation of second-order features, the fusion of first-order and second-order features, the interaction between spatial features and temporal features layer by layer. In particular, attention mechanism is introduced to automatically assign weights to spatial features and realize the discriminative application in feature fusion. Experiments on a large-scale dataset demonstrate the effectiveness of the proposed method: (1) The prediction error can be significantly reduced in comparison with the baseline methods particularly in the case of sparse training data, where our models achieve a performance improvement by about 10.9–21.0% in term of MAE and NMAE, and by 2.4–7.8% in term of RMSE. (2) Attention mechanisms enable us to give intuitive explanations of the effectiveness of feature fusion more reasonably and thus strengthen the interpretability of the prediction models.

1. Introduction

Web services are self-described software designed to support interoperable machine-to-machine interaction over the Internet via standard interfaces and communication protocols such as SOAP and REST [1]. They are widely used in service-oriented architecture (SOA) because of their low coupling, ease of reuse and composability. More and more Web services are utilized to build Web applications with the integration of Cloud computing, Internet of Things and SOA [2]. Generally, it is very difficult for inexperienced users to select the most appropriate ones as faced with various candidate services of similar functionalities. In this case, QoS-based service evaluation has become more important as users can now take a decision on choosing appropriate services with distinguishable quality values of candidate services. To obtain accurate client-side quality values for individual users, client-side service evaluations are usually needed [3]. Due to constraints on time, cost and other factors, service providers cannot dispose a large number of software sensors in the network environments to monitor QoS information for every service [4]. To deal with this challenge, many efforts have been done to build models for collaborative QoS prediction by drawing lessons from the recommender systems, where memory/model-based collaborative filtering [5,6] are used to exploit the known QoS data of service invocations and predict the unknown quality values given active users and services. This treatment exploits crowd intelligence to aid the QoS assessment and avoid intuitive data

* Corresponding author.
E-mail addresses: zqmynu@gmail.com (Q. Zhou), haowu@ynu.edu.cn (H. Wu), kyue@ynu.edu.cn (K. Yue), chh@cs.ccu.edu.tw (C.-H. Hsu).

https://doi.org/10.1016/j.future.2019.05.024
0167-739X/© 2019 Elsevier B.V. All rights reserved.

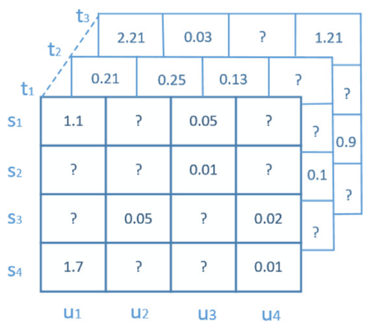


Fig. 1. The structure of a three dimension matrix in terms of user, service and time.

measurement, thereby save time and economic costs for both providers and users of services [7].

Currently, only a few of the existing works take spatio-temporal contexts into consideration in the process of QoS prediction [8,9]. Spatio-temporal context-awareness is a potential demand for the task of QoS prediction. On the one hand, the quality of Web services is highly related to invocation temporal factors, since the service status (e.g., workload) and the network environment (e.g., congestion) change over time [10]. Therefore, the observed QoS data record is actually a tensor structure in morphology (illustrated by Fig. 1). On the other hand, service invocations involve complex spatial contexts which contain not only distributed interactions between client hosts and service hosts, but also the interactions between spatial environments of the two ends, as service invocations occur not only the client hosts and the server hosts, but also across communication paths composed in tandem by different subnets, autonomous systems and geo-locations. Without considering these complexities, it is very challenging for traditional prediction methods to achieve more accurate results and really meet users' requirements on service selections. In this regard, there have been some exploratory works [9]. However, they exploit the characteristic of contexts for one specific dimension, e.g., time or location, and do not take full advantage of various contextual features. In particular, many features are associated with each other, resulting in difficulties in feature selection and utilization. Consequently, it continues to be a technical challenge to realize collaborative QoS prediction with spatio-temporal contexts.

Guided by these, we propose two general neural models (STCA-1 and STCA-2) for the task of spatio-temporal context-aware collaborative QoS prediction by considering multiple spatial features of users and services and temporal feature simultaneously. In our methods, we use a hierarchical structure to model the interactions of features in the spatial dimension, and the interaction between the spatial dimension and temporal dimension in turn. As there are many distinct features in the spatial dimension in comparison with the temporal dimension, we try to use neural networks to merge different spatial features into a single vector representation in the first-order and second-order cases respectively. In particular, we utilize the attention mechanism to automatically assign weights to different features and quantify their contributions to the final vector representation of spatial dimension. In addition, the linear combination of all spatial and temporal features is added to the final framework, to have the best of both non-linear and linear prediction components. With historical QoS data of service invocations, our models will be trained to optimize the unknown parameters and then the trained models will give an estimated quality value given an active user, a target service and an invocation time as well as associated spatial features. Regarding the data structure in Fig. 1, our models aim at completing any unknown entries against all known ones.

The main contributions of this article are summarized as follows:

(1) We develop neural network based methods for the task of context-aware QoS prediction and propose two novel variants as STCA-1 and STCA-2. The hierarchical structure provides a flexible manner to model the interactions of features in the spatial dimension, the interaction between the spatial dimension and temporal dimension.
(2) We utilize the attention mechanism to merge different spatial features into a single vector representation in the first-order and second-order cases respectively. Meanwhile, it enables us to automatically assign weights to different features and quantify their contributions to the final vector representation of spatial dimension.
(3) We conduct large-scale experiments on the task of response time prediction to study the proposed models, demonstrating their effectiveness and accuracies.

The remainder of this paper is organized as follows. Section 2 reviews the most relevant works to ours. We elaborate architecture and details of our proposed models in Section 3. Section 4 shows our extensive experimental results and Section 5 concludes this article.

2. Related works

Shao et al. first propose memory-based collaborative filtering for the QoS prediction with similarity mining [11]. Following works basically focus on how to improve the accuracy of similarity computation. Zheng et al. integrate user-based and item-based methods linearly and improve the TopK algorithm [12]. Wu et al. exploit adjusted-cosine-based similarity calculation to find neighbors of users and services [13]. Sun et al. exploit the characteristics of QoS values in the process of similarity measurement for memory-based collaborative filtering [14]. Chen et al. utilize QoS values and users' locations to make personalized QoS prediction [15]. Tang et al. compute neighborhoods of users and services based on their locations and combine user-based and service-based collaborative filtering techniques [16]. Xu et al. identify similar neighbors of users based on the geographical information and services based on affiliation information [17].

These methods mentioned above still have some weaknesses. Firstly, the selection of similarity measurements obviously affects the accuracy of QoS prediction. Secondly, the performance of these models can be cut dramatically when the data is sparse, because it is struggling for these models to compute the similarities accurately in sparse data. While matrix factorization can easily utilize additional information related to users and services to mitigate these problems to some extent. Zheng et al. use the probabilistic matrix factorization (PMF) to decompose the user–item matrix [4]. He et al. and Lo et al. design a matrix factorization model for taking the geographical information of users and services into consideration respectively [18,19]. Ryu et al. propose a location-based matrix factorization using a preference propagation method to address the cold-start problem [20]. Wu et al. propose a context-sensitive matrix factorization approach by making use of multiple contextual factors of users and services [8].

There are also some works concentrate on time-series-oriented QoS prediction. Hu et al. propose a time-aware approach which integrates time information into both similarity measurement and the final QoS prediction [22]. Zhang et al. provide a time-aware personalized QoS prediction method which analyzes latent

Table 1
Summarization of related works.

Models	Principle of algorithm	Context-Awareness		
		Spatial	Temporal	Users/Services
MD_{pr} [11]	Memory-based CF	×	×	√
WSRec [12]	Memory-based CF	×	×	√
ADF [13]	Memory-based CF	×	×	√
NRCF [14]	Memory-based CF	×	×	√
LoRec [15]	Memory-based CF	√	×	√
COFILL [16]	Memory-based CF	√	×	√
LE-MF [17]	Memory-based CF	√	×	√
QP_VCM [21]	Memory-based CF	×	√	√
TAWSRec+HRW [22]	Memory-based CF	×	√	√
CASR-TSE [23]	Memory-based CF	√	√	√
NIMF [4]	Matrix Factorization	×	×	√
HMF [18]	Matrix Factorization	√	×	√
LoNMF [19]	Matrix Factorization	√	×	√
LMDC [24]	Matrix Factorization	√	×	√
LMF-PP [20]	Matrix Factorization	√	×	√
CSMF [8]	Matrix Factorization	√	×	√
WSPred [10]	Tensor Factorization	×	√	√
HDOP [25]	Tensor Factorization	×	√	√
TLACF [9]	Tensor Factorization	√	√	√
LASSO [26]	LASSO	√	√	√
DNM [7]	Neural Network	√	×	√
STCA (Our Works)	Neural Network	√	√	√

features of users, services and time by performing tensor factorization [10]. Ma et al. propose a multi-valued collaborative QoS prediction via time series analysis for potential users [21]. Wang et al. propose a method which unifies the modeling of multi-dimensional data via tensor factorization but only utilizes time information in the QoS prediction task [25]. Wang et al. propose a novel spatial–temporal QoS prediction approach for time-aware Web service recommendation where the geo-location of web service is employed to detect neighbors of user–service pairs and the QoS prediction is modeled as a Lasso regression problem [26]. Fan et al. propose CASR-TSE method by modeling the effectiveness of spatial correlations between the user's location and the service's location, also exploiting temporal decay model incorporating the weighted rating effect in similarity computation [23].

To investigate the quality of Web services from multiple dimensions, Yu and Huang propose a time-aware and location-aware collaborative filtering algorithm [9]. In this paper, the users set and services set are divided into many clusters according to their location information, then, it seeks similar users and services within smaller and highly similar clusters. And the time factor is utilized to improve the prediction accuracy. However, it is still a technical challenge to be taken into account the influence of complex spatio-temporal contexts of service invocations and make use of their characteristics in the forecasting process. As many contemporary QoS prediction methods exploit the characteristic of contexts for one specific dimension, e.g., time or location, and do not take full advantage of various contextual features. Also, most time-aware methods focus on the sequential QoS prediction, i.e. where the sequences of the observed QoS values are used to forecast the QoS values of the next points given a user–service pair.

With the flourish of deep learning techniques, some research works based on the neural network in the field of QoS prediction have been conducted in recent years. Xiong et al. introduce deep learning model to learn the hidden features from context and compute the similarity between users and services based on the hidden features [24]. Wu et al. propose an universal deep neural model to make multi-attribute QoS prediction with multiple contextual features [7]. The summarization of related works is shown in Table 1.

Compared to the existing works, our works attempt to exploit neural model for QoS prediction by modeling the spatial and temporal features simultaneously, and the proposed models are easy to be expanded to other features or dimensions. Spatial features along with temporal feature are mapped to embedding vectors to improve the capacity of semantic expression. Then the high order interaction of spatial features can be captured by the interaction layer. And we assign different weight to each spatial feature automatically by attention mechanism and to distinguish their contributions to the prediction results. Finally, the linear combination of spatial features and temporal feature is taken into account. Additionally, different from the sequential prediction of QoS values, our methods concentrate on completing a tensor structure of user–service-time QoS values without directly modeling the temporal dependencies between different time slices.

3. Neural models for spatio-temporal context-aware QoS prediction

Our goal addresses modeling and predicting the quality of services by incorporating spatial features and temporal feature into the process of QoS prediction. Quality values of services are modeled as the function of spatial features of users and services, as well as temporal feature. To define the features uniformly, we just view the ID of users and services as two kinds of special spatial features. And then we can formally define the predicting process as a function: $\hat{Q}_{xt} = f(X, T)$, where $\hat{Q}_{xt}$ represents the estimated quality value from a user u to a service s at a time slice $t \in T$, and $x \in X$ indicates the overall combination of spatial context from u to s. Once the function is estimated, this predictor can make QoS prediction with given spatial features of users/services and an invocation time. In the next, we will detail the architecture of our proposed models (as shown in Fig. 2) for implementing the prediction function.

3.1. Input layer

In the Input Layer, we represent all the spatial features associated with service invocations in input vectors. For ease of illustration, we just consider ID number, IP address, city and autonomous system for each user or service. Then, an input vector x will be an instance of the spatial feature collection: $X = \{uid, uip, ucity, uas, sid, sip, scity, sas\}$. In the same way, we

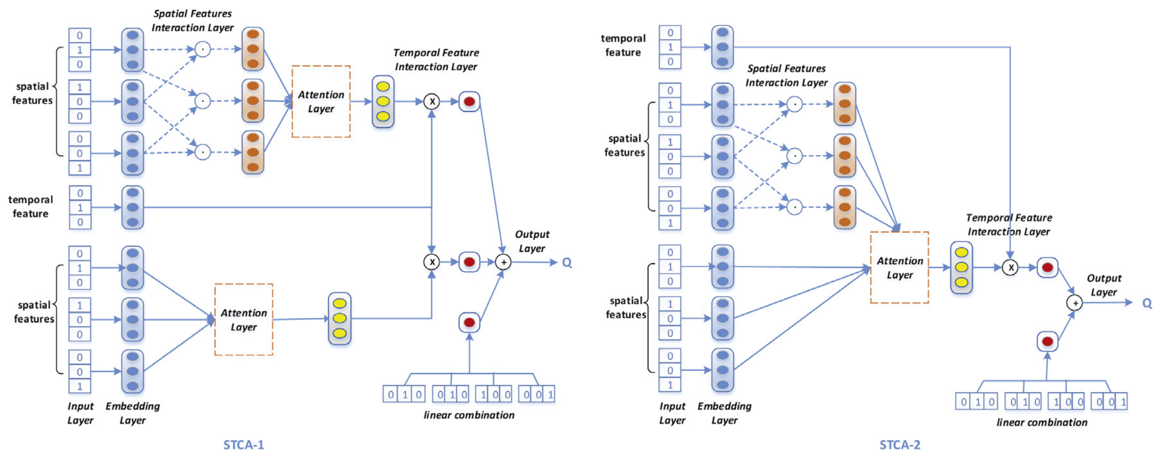


Fig. 2. The architecture of spatio-temporal context-aware collaborative QoS prediction model. Noted that spatial features are shared in the fusion of second-order and the first-order features. The left part is the architecture of STCA-1 and the right part is the architecture of STCA-2.

convert the time t when the service invocation happens into a vector, and the collection of temporal feature T is described as: $T = \{t\}$.

To make it easier to understand, an example is given as follows: suppose there are n=12 spatial features and m=3 temporal features in the whole dataset. If an invocation occurs from user u_1 to service s_2 at time slice t_3, then the input features are denoted as a list: $[uid_1, uip_1, ucity_1, uas_1, sid_2, sip_2, scity_2, sas_2, t_3]$. To formally define the input vector of spatial features x and the input vector of temporal feature t, we assign each different feature a local identifier (for spatial features and temporal feature respectively) which indicates the index of the feature, then the identifiers of spatial features could be described as: [0, 2, 4, 5, 6, 8, 10, 11] and the identifier of temporal feature could be [2]. If the ith feature exists in this instance, $x_i = 1$; otherwise, $x_i = 0$. Correspondingly, the input vector of spatial features can be defined as: $x = [101011101011]$. Similarly, the input vector of temporal features is $t = [001]$.

3.2. Embedding layer

In the *Embedding Layer*, each feature is mapped into a dense vector in a d-dimensional latent space $\mathbf{R}^d$. This is similar to the latent representation of users and services in matrix factorization [8]. In this way, all features can be characterized by the common latent factors in $\mathbf{R}^d$, and then it becomes easier to distinguish the differences of features in the same coordinate system.

Formally, let $\vec{e} \in \mathbf{R}^d$ be the embedding vector used to characterize a spatial feature, after embedding operation, we will get a collection of embedding vectors for the spatial features: $E = \{x_0\vec{e}_0, \dots, x_i\vec{e}_i, \dots, x_{n-1}\vec{e}_{n-1}\}(0 \le i \le n-1)$, where n is the number of distinct spatial features. Likewise, let $\vec{v} \in \mathbf{R}^d$ be the embedding vector of a temporal feature, we will get another collection of embedding vectors: $V = \{t_0\vec{v}_0, \dots, t_j\vec{v}_j, \dots, t_{m-1}\vec{v}_{m-1}\}(0 \le j \le m-1)$, where m is the number of distinct temporal feature values.

While all distinct feature values cannot appear in one input sample simultaneously. In our system, only eight distinct spatial feature values will appear in an input sample, which includes ID number, IP address, city and autonomous system of users and services respectively. We can only consider the embedding vectors of those existed features as $E = \{x_i\vec{e}_i | x_i \neq 0\}$ and $V = \{t_j\vec{v}_j | t_j \neq 0\}$. As for the above-mentioned example, $E = \{x_0\vec{e}_0, x_2\vec{e}_2, x_4\vec{e}_4, x_5\vec{e}_5, x_6\vec{e}_6, x_8\vec{e}_8, x_{10}\vec{e}_{10}, x_{11}\vec{e}_{11}\}$ and $V = \{t_2\vec{v}_2\}$.

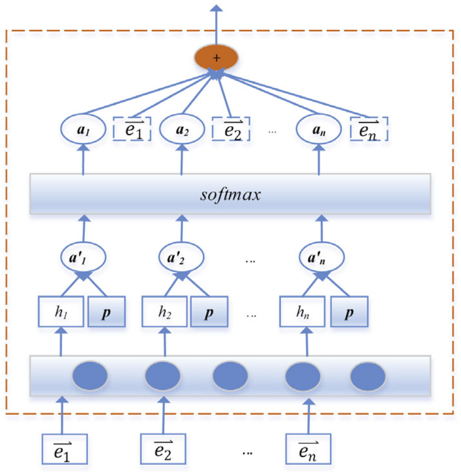


Fig. 3. The architecture of Attention module.

3.3. Spatial features interaction layer

The *Spatial Features Interaction Layer* is used to capture the second-order features which cannot be intuitively represented by the first-order spatial features above the Embedding Layer. For instance, a communication path cannot be directly portrayed by a single spatial feature, but it can be designated by combining a user address and a service address. This function can be realized by conducting the element-wise product on the embedding vectors of different spatial features [27,28]. When the collection E is fed into this layer, these embedding vectors will be converted into another set of vectors:

$$E^{(1)} = \{x_i\vec{e}_i \cdot x_j\vec{e}_j \mid i = 0 : n-2, j = i+1 : n-1\} \tag{1}$$

where $\cdot$ denotes the element-wise product of two vectors which have the same dimension and $x_i\vec{e}_i \in E$. Totally, there are $N = \frac{n(n-1)}{2}$ distinct features in $E^{(1)}$ and every element is also a d-dimensional vector that encodes an interaction between two spatial features [29].

For the variant model (SCTA-2), we directly aggregate the second-order spatial features and the first-order spatial features into a new collection: $E^{(2)} = E^{(1)} \cup E$, and there are totally $M = \frac{n(n+1)}{2}$ distinct features in this collection.

3.4. Attention layer

Attention mechanism in deep learning is essentially similar to the selective attention mechanism of human vision, and the core goal is to select more critical information from a large amount of information for the current task goal. Attention mechanism has been shown effective for the task of feature weighting and selection in various research works such as machine translation and image caption [30,31]. In our task, to distinguish the contribution of so many features, attention mechanism is utilized to assign different weights to each spatial feature automatically by following [32]. Higher weight indicates that the corresponding features are more informative for the consequence. Instead, lower weight indicates that the corresponding features are less informative. When feeding a collection of embedding vectors into the Attention layer, all the embedding vectors in the collection will be converted to one d-dimensional vector with different weights as showed in Fig. 3. As for the collection $E^{(1)}$, we will get a fused vector,

$$\vec{e}^{(1)} = \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} a_{ij}(x_i\vec{e}_i \cdot x_j\vec{e}_j) \tag{2}$$

where a_{ij} is the weight of a second-order spatial feature which is obtained by attention mechanism [33,34] as follows:

$$\begin{aligned} a'_{ij} &= \mathbf{p}_1^T ReLU(\mathbf{W}_1(x_i\vec{e}_i \cdot x_j\vec{e}_j) + \mathbf{b}_1) \\ a_{ij} &= \mathbf{softmax}(a'_{ij}) = \frac{\exp(a'_{ij})}{\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}\exp(a'_{ij})} \end{aligned} \tag{3}$$

where $\mathbf{W}_1 \in \mathbf{R}^{h\times d}$, $\mathbf{b}_1 \in \mathbf{R}^h$, $\mathbf{p}_1 \in \mathbf{R}^h$ are initialized model parameters in the first attention module, and h denotes the dimensionality of hidden factors. a'_{ij} is the original weight generated from attention module. Specifically, we use a multi-layer perceptron (MLP) and an inner product operation to compute these weights, then employ *ReLU* (Rectified Linear Units) as the *activation function* in the MLP which shows good performance empirically [35]. Finally, the original weights are normalized through the *softmax* function.

In the same way, the collection E can be compressed as a d-dimension vector as follows:

$$\begin{aligned} \vec{e}^{(2)} &= \sum_{i=0}^{n-1} a_i(x_i\vec{e}_i) \\ a'_i &= \mathbf{p}_2^T ReLU(\mathbf{W}_2(x_i\vec{e}_i) + \mathbf{b}_2) \\ a_i &= \frac{\exp(a'_i)}{\sum_{i=0}^{n-1}\exp(a'_i)} \end{aligned} \tag{4}$$

where $\mathbf{W}_2 \in \mathbf{R}^{h\times d}$, $\mathbf{b}_2 \in \mathbf{R}^h$, $\mathbf{p}_2 \in \mathbf{R}^h$ are the initialized model parameters in the second attention module.

It is useful to emphasize that we employ two independent attention modules to respectively deal with the weight allocation and feature aggregation of first-order and second-order spatial features in the model STCA-1. As for the variant model (STCA-2), we just need one attention module for $E^{(2)}$ to fuse all the spatial features into a d-dimensional vector $\vec{e}^{(3)}$ with the same operation as in Eq. (4). This is the main difference between two prediction models. However, according to experimental results, independent aggregation of the first-order and second-order features contributes to strengthening the model flexibility and thus reduce the prediction errors.

3.5. Temporal feature interaction layer

In reality, the same users may observe distinctive quality of services even if they invoke the same services just at discrete time slices, because the global network status is not always stationary. Fig. 4 shows the response time's variation tendency during 50 time intervals of two user–service pairs, they are (user 0, service 54) and (user 1, service 23) respectively. Apparently, QoS experience of the same user–service pair in discrete time slices is different. Hence, it is reasonable to take temporal feature into consideration in the process of QoS prediction.

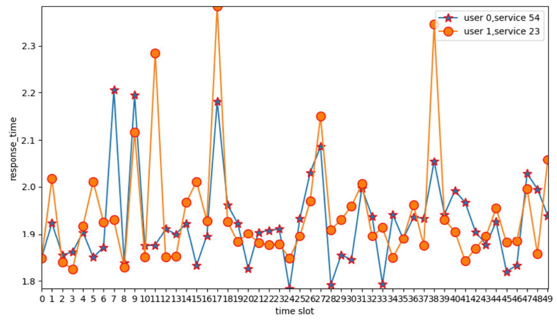


Fig. 4. The response time's variation trend during 50 time intervals of two user–service pairs.

Traditional time series prediction only considers the temporal dependence of end-to-end paths, so it is hard to describe the state of the whole network at different times. In our works, the time dimension is discretized and each time slice is characterized by a latent feature vector to globally describe the current state of the whole network and is shared by the spatial features of the whole network. When modeling the interaction between these temporal features and the spatial features with the inner product of vectors, all end-to-end QoS predictions at discrete time slices will become aware of these global states. Therefore, we set up a *Temporal Feature Interaction Layer* to realize this principle. Formally, it is defined as:

$$\begin{aligned} q_1 &= \vec{e}^{(1)} \times t_j\vec{v}_j \\ q_2 &= \vec{e}^{(2)} \times t_j\vec{v}_j \\ q_3 &= \vec{e}^{(3)} \times t_j\vec{v}_j \end{aligned} \tag{5}$$

where $\times$ denotes the inner product of two vectors and $t_j\vec{v}_j$ is the embedding vector of temporal feature in a certain sample. Clearly, the output of this operation is a scalar value. q_1 is a result of the interaction between temporal feature and the second-order spatial features and q_2 is a result of the interaction between temporal feature and the first-order spatial features. Both q_1 and q_2 are achieved in the STCA-1 model, while q_3 is an intermediate result in the STCA-2 model.

3.6. Output layer

Inspired by linear regression [36] and factorization machine (FM) [29], the linear regression part of FM is adopted in our models. After that, the output is the sum of two parts, one is the interaction between spatial features and temporal feature, the other is the linear combination of all features. It can be described as follows (Eq. (6) is the result of the first model STCA-1 and Eq. (7) is the result of the second model STCA-2):

$$\hat{Q}_{xt} = b_0 + \sum_{i=0}^{n-1}\alpha_i x_i + \sum_{j=0}^{m-1}\beta_j t_j + q_1 + q_2 \tag{6}$$

$$\hat{Q}_{xt} = b_0 + \sum_{i=0}^{n-1}\alpha_i x_i + \sum_{j=0}^{m-1}\beta_j t_j + q_3 \tag{7}$$

where $b_0 + \sum_{i=0}^{n-1} \alpha_i x_i + \sum_{j=0}^{m-1} \beta_j t_j$ represents the linear combination of all features, and b_0 is the global bias, α_i denotes the weight of the ith spatial feature, β_j denotes the weight of the jth temporal feature. As the example mentioned in Section 3.1, the input vector of spatial features is [101011101011] and the input vector of temporal feature is [001], then for STCA-1, $\hat{Q}_{xt} = b_0+\alpha_0+\alpha_2+\alpha_4+\alpha_5+\alpha_6+\alpha_8+\alpha_{10}+\alpha_{11}+\beta_2+q_1+q_2$. Similarly, for STCA-2, $\hat{Q}_{xt} = b_0+\alpha_0+\alpha_2+\alpha_4+\alpha_5+\alpha_6+\alpha_8+\alpha_{10}+\alpha_{11}+\beta_2+q_3$.

3.7. *Objective function and model learning*

To estimate model parameters of STCA-1 and STCA-2, we are required to specify an objective function to optimize. There are two commonly used loss functions are the least absolute deviations (L_1) and least square errors (L_2), which can respectively minimize the absolute differences and the squared differences between the estimated value and the original value. L_1 is more robust and generally not affected by outliers in comparison with L_2 on the QoS data [7]. If a sample is an outlier, L_2 loss will be adjusted to minimize the outlier at the expense of many other common samples. Since the QoS data usually contains many outliers and L_2 loss squares the errors, the model will see more amplified errors than L_1 loss. As a whole, the model will be more sensitive to the QoS data. Consequently, we adopt L_1 loss to form the objective function. To alleviate overfitting, we apply a L_1 regularizer in the objective function as follows:

$$\mathcal{L} = \min_{\phi} \sum_{x\in X, t\in T} |\hat{Q}_{xt} - Q_{xt}| + \lambda_1 \|\theta\|_1 \tag{8}$$

where $\hat{Q}_{xt}$ is the predicted value of a given sample x, t, and Q_{xt} is the corresponding true value. The goal of this function is to narrow the gap between estimated value and true value to optimize the parameters in $\phi = \{\vec{e}_*, \vec{v}_*, W_*, b_*, p_*, \alpha_*, \beta_*\}$. $\theta = \{\vec{e}_*, \vec{v}_*, W_*, b_*, p_*\}$ are the parameters which need to be regularized including the embedding vectors of all features and part of the parameters in attention modules. $\lambda_1\|\theta\|_1$ is the L_1 regularizer, consists of L_1 *norm* of parameters in scalars/vectors/tensors and λ_1 is the regularization parameter [37]. We also utilize L_2 loss to optimize model parameters as contrastive experiments, and a L_2 regularizer is utilized to alleviate the overfitting problem in the objective function:

$$\mathcal{L} = \min_{\phi} \sum_{x\in X, t\in T} \frac{1}{2}(\hat{Q}_{xt} - Q_{xt})^2 + \lambda_2 \|\theta\|_2^2 \tag{9}$$

where $\lambda_2\|\theta\|_2^2$ is the L_2 regularizer, consists of squared L_2 *norm* of parameters θ and λ_2 is the regularization parameter [37]. In mathematical form, $\|\theta\|_1$ is the sum of absolute values of all parameters and $\|\theta\|_2^2$ is the sum of squares of all parameters. For the sake of simplicity, we ignore these details.

As for model learning, the objective functions can be optimized by stochastic gradient descent (SGD) and its variants like Adagrad, Adam and Momentum [38]. General update equation is:

$$\phi = \phi - \eta \frac{\partial \mathcal{L}}{\partial \phi}, \tag{10}$$

where η is the learning rate. Moreover, a *mini-batch* gradient descent, which processes a small subset of the training set in each iteration, is used to speed up training on the large-scale datasets. Typical deep learning library like TensorFlow provides automatic differentiation with *mini-batch* strategy and hence we omit the gradient equations which can be computed by chain rule in back-propagation (BP).

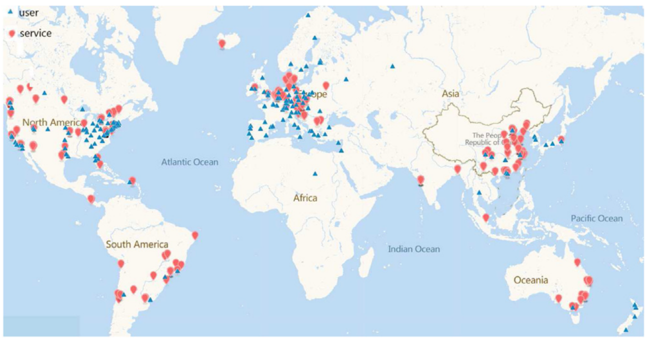


Fig. 5. The distribution of users and services in the dataset all over the world.

Table 2
Statistics of dataset.

Statistics	Values
#Users (uid)	420
#Services (sid)	1000
#Invocation time (t)	32
#Service Invocations	13,440,000
#Users' IP Address (uip)	354
#Users' City (ucity)	169
#Users' Autonomous System (uas)	147
#Services' IP Address (sip)	318
#Services' City (scity)	163
#Services' Autonomous System (sas)	179
Range of response-time	0–20 s

4. Experiments

4.1. *Datasets*

To evaluate the performance of the proposed model, we use one of the WSDream datasets.[1] The dataset is collected by using the PlanetLab platform[2] and comprises of a large-scale invocation results between 420 geo-distributed users and 1000 real-world Web services over 480 consecutive time slices at an interval of 30 minutes. Fig. 5 shows the distribution of users and services all over the world.

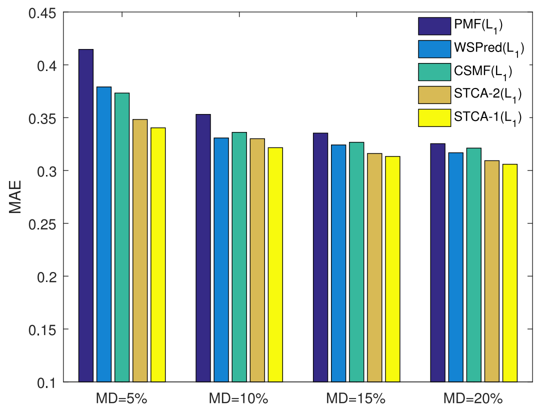
For our experiments, we further extend the dataset by adding spatial information. For each user or service, we find its IP address and city according to it's URL through a website called IP Location Finder[3] and then obtain autonomous system through Team Cymru.[4] Owing to the limitation of computing resource, we just use the first 32 time slices (0–31). After that, there are 420 users, 1000 services, 32 time slices in our experiments. Other statistics of the dataset are given by Table 2.

To study the impact of data sparsity on the accuracy of QoS predictions, we randomly split the QoS dataset into the training set and the test set by a given ratio, the ratio is also denoted as matrix density (MD). For example, MD=4% means for each time-slice, we randomly choose 4 percent records from the QoS matrix as the training subset, and take the remainders as the test subset. Training set is used to train our models and optimize parameters of our models, and test set is utilized to verify the prediction accuracy and performance of our methods.

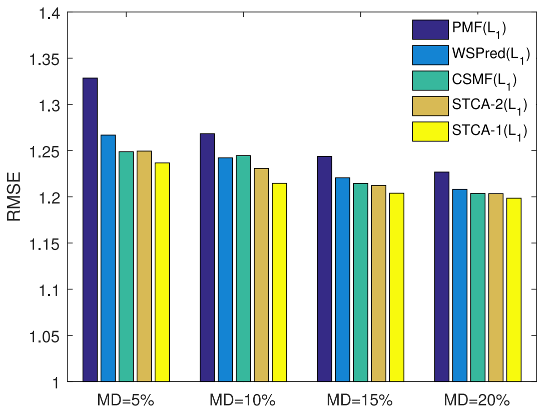
1 http://wsdream.github.io/#dataset.
2 https://www.planet-lab.org.
3 https://www.iplocation.net/.
4 http://www.team-cymru.com/.

Table 3
Performance comparisons of response-time prediction using different matrix densities. Best values for both baseline methods and STCA models are marked in bold font. Gains are calculated based on these values and all of them have a confidence level of 99%. The variances of experimental results are shown in Table 5.

Methods	MD=1%			MD=2%			MD=3%			MD=4%		
	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE
UMEAN	0.7546	1.5362	0.0378	0.7289	1.4599	0.0365	0.7201	1.4949	0.0360	0.7155	1.4897	0.0358
IMEAN	0.7692	1.6547	0.0396	0.7447	1.5707	0.0373	0.7248	1.5370	0.0362	0.6908	1.5200	0.0357
UPCC	0.7541	1.5388	0.0377	0.7101	1.5308	0.0355	0.6262	1.4037	0.0313	0.5656	1.2840	0.0283
IPCC	0.7911	1.6334	0.0396	0.7391	1.5517	0.0370	0.6835	1.5016	0.0342	0.6139	1.3583	0.0307
UIPCC	0.7517	1.6049	0.0376	0.7093	1.5193	0.0355	0.6601	1.4318	0.0330	0.5970	1.3347	0.0299
LASSO	0.8264	1.5926	0.0394	0.8008	1.5639	0.0382	0.7920	1.5523	0.0378	0.7846	1.5417	0.0374
PMF(L_2)	0.9081	1.8066	0.0433	0.6427	1.5944	0.0305	0.6065	1.4654	0.0287	0.5973	1.4046	0.0285
PMF(L_1)	0.8410	1.7453	0.0395	0.5897	1.5217	0.0281	0.5245	1.4161	0.0250	0.4569	1.3617	0.0215
WSPred(L_2)	0.9142	1.8038	0.0435	0.6706	1.5697	0.0319	0.6180	1.4606	0.0283	0.5529	1.3551	0.0263
WSPred(L_1)	0.8461	1.7390	0.0403	0.5967	1.4579	0.0284	0.4829	1.3348	0.0230	0.4040	1.2821	0.0192
CSMF(L_2)	0.6833	1.5059	0.0341	0.6421	1.4272	0.0319	0.6085	1.3925	0.0299	0.5753	1.3602	0.0280
CSMF(L_1)	**0.5710**	**1.4477**	**0.0272**	**0.4666**	**1.3391**	**0.0222**	**0.4156**	**1.2905**	**0.0198**	**0.3876**	**1.2645**	**0.0184**
STCA-2(L_2)	0.5612	**1.3341**	0.0267	0.4943	1.2752	0.0236	0.4811	1.2581	0.0229	0.4637	1.2403	0.0221
STCA-2(L_1)	**0.4523**	1.3609	**0.0215**	0.3897	1.2928	0.0186	0.3667	1.2766	0.0175	0.3584	1.2694	0.0171
STCA-1(L_2)	0.5468	1.3507	0.0260	0.4850	1.2755	0.0223	0.4684	1.2535	0.0223	0.4552	**1.2347**	0.0216
STCA-1(L_1)	0.4561	1.3462	0.0217	**0.3782**	**1.2707**	**0.0180**	**0.3542**	**1.2478**	**0.0169**	**0.3453**	1.2410	**0.0164**
Gains	20.8%	7.8%	21.0%	18.9%	5.1%	18.9%	14.8%	3.3%	14.6%	10.9%	2.4%	10.9%



(a)



(b)

Fig. 6. Performance comparisons of response-time prediction with selected methods in the case of dense training data.

4.2. Evaluation metrics

Mean absolute error (MAE) and root mean squared error (RMSE) metrics are two basic accuracy metrics which used to evaluate the performance of our proposed models. MAE is defined as:

$$MAE = \frac{\sum_{x\in X,t\in T}|\hat{Q}_{xt} - Q_{xt}|}{N_{sample}} \tag{11}$$

where $\hat{Q}_{xt}$ and Q_{xt} are respectively the predicted value and true value of sample x, t, and N_{sample} is the number of samples in the dataset. RMSE measures the error of the predicted value by quadratic scoring the error between the predicted and true value which is defined as:

$$RMSE = \sqrt{\frac{\sum_{x\in X,t\in T}(\hat{Q}_{xt} - Q_{xt})^2}{N_{sample}}} \tag{12}$$

RMSE represents a relatively high weight for a large difference because differences are squared before they are divided by the number of samples. So RMSE is more sensitive to those larger or smaller values compared to the MAE. To measure the performance of our models from multiple aspects, we also adopt normalized mean absolute error (NMAE) as one of the evaluation metrics which is defined as :

$$NMAE = \frac{MAE}{max_{Q_{xt}} - min_{Q_{xt}}} \tag{13}$$

where $max_{Q_{xt}}$ and $min_{Q_{xt}}$ respectively represent the maximum observed value and minimum observed value among all samples. Smaller value of MAE, RMSE and NMAE indicate better performance.

4.3. Comparison methods

To evaluate the proposed method and show the performance of our method, it is compared with existing state-of-the-art methods. The methods implemented for comparison which often used as baseline methods are as follows:

1. *UMEAN* takes the average QoS value known by user u as the predictive QoS value of user u to service s;
2. *IMEAN* takes the average QoS value observed from service s as the predictive QoS value of user u to service s;
3. *UPCC* is user-based collaborative prediction model. Top-N neighbors of users are found using PCC-based similarity [11];
4. *IPCC* is item-based collaborative prediction model. Top-N neighbors of services are found using PCC-based similarity [39];

5. *UIPCC* combines the user-based and item-based collaborative prediction approaches and employs both the similar users and similar services for the QoS value prediction [39];
6. *PMF* uses probabilistic matrix factorization [40] to factorize user–service QoS matrix for the prediction [4];
7. *LASSO* focuses on predicting the next QoS value, so it cannot directly compare with other methods. For our scenario, we utilize the longitude and latitude of users/services to detect neighbors of a user–service pair, and exploit a Lasso regression model to perform QoS prediction in each time slot [26];
8. *CSMF* is a context-sensitive matrix-factorization method to make collaborative QoS prediction by modeling the interactions of user-to-service and environment-to-environment simultaneously [8];
9. *WSPred* is a time-aware QoS prediction approach based on tensor factorization algorithm, the user-specific, service-specific and time-specific latent features can be extracted from the past QoS experiences [10].

Noted that all these methods are implemented with Tensorflow on a computer configured with Intel Core i7 processor, 16 GB memory with GPU support of Nvidia GTX 1080Ti.

4.4. Performance comparisons

To compare the QoS prediction performance of selected methods, we use following configurations: (a) For memory-based CF methods, like IPCC, UPCC and UIPCC, we fix the neighborhood size of users at 10 and the neighborhood size of services at 50. (b) For learning-based models like PMF, CSMF, WSPred, STCA-1 and STCA-2, the dimensionality of embedding vectors is $d = 64$, the regularization parameters are respectively λ_1=0.001 and λ_2=0.1 for L_1 loss and L_2 loss, the *mini-batch* size is 1024 and the maximum number of iterations is fixed at 100. (c) Specially, the number of hidden factors in the Attention layer for our proposed models is $h = 8$. (d) To compare the effects of L_1 loss and L_2 loss on the learning-based models, we build the corresponding objective functions by following Eqs. (8) and (9). Correspondingly, we have PMF (L_1), CSMF (L_1), WSPred (L_1), STCA-1 (L_1), and STCA-2 (L_1) which respectively combine a L_1 loss and a L_1 regularizer; PMF (L_2), CSMF (L_2), WSPred (L_2), STCA-1 (L_2) and STCA-2 (L_2) which respectively combine a L_2 loss and a L_2 regularizer. (e) We choose AdamOptimizer as the optimizer and the learning rate is initially as η=0.0005. We compare average of the best prediction results in five random iterative processes.

We first compare the selected methods in the case of sparse training data, where we let *MD*=1%–4%. The experimental results are shown in Table 3. Generally, traditional memory-based CF like UPCC, IPCC, UIPCC perform poorly than the model-based CF like PMF, WSPred and CSMF. From an overall perspective, CSMF (L_1) outperforms all other baselines thanks to the use of spatial contexts of service invocations. As the volume of training data increases, WSPred also shows a significant performance improvement, due to the consideration of the global impact of temporal factors. However, without considering the influence of complex spatio-temporal factors at the same time, they cannot compare with our proposed methods. As for our methods, both STCA-1 and STCA-2 significantly outperform the baseline methods, which confirms the effectiveness of our proposed methods in the case of sparse training data. Regarding to LASSO, it is specially designed for the sequential prediction of QoS values, and thus fails in our scenario.

When considering the effects of L_1 loss and L_2 loss, we can easily find that optimizing L_1 loss yields a better predictor than optimizing L_2 loss. As for PMF, WSPred, CSMF, STCA-1 and STCA-2, the variants based on L_1 loss consistently performs superior to those variants based on L_2 loss in term of MAE and NMAE, while optimizing L_2 loss can obtain lower RMSE in some circumstances. Totally, our models achieve a performance improvement by about 10.9–21.0% in term of MAE and NMAE, and by 2.4–7.8% in term of RMSE with optimizing L_1 loss. This is compatible with the previous theoretical analysis. Therefore, in practical scenarios, we recommend building the objective function of the prediction models based on L_1 loss, so as to reduce the negative effects of outlier in the QoS data.

We then compare the selected methods in the case of dense training data, where we let *MD*=5%–20%. The experimental results are presented in Fig. 6. With the increase of training data volume, parameters in learning-based models have more opportunities to get optimized, so simple models can also achieve significant performance improvements. Accordingly, the impact of spatial features on prediction performance is weakened. As an evidence, WSPred can exceed CSMF in term of MAE when MD increases to 10%–20%. Nevertheless, our models are still better than the baseline models in term of both MAE and RMSE. Furthermore, the conclusion that STCA-1 is superior to STCA-2 can be confirmed by combining the results of Table 3 and Fig. 6. Since STCA-1 separates the first-order spatial features from the second-order spatial features, we can find that this treatment gives the model more flexibility in feature selection and utilization, thus results in performance gains.

Finally, we conduct a case study to observe the effect of individual QoS predictions. To compromise the cases of sparse and dense data, we choose MD = 5%. We randomly choose five user–service pairs and observe two time slots for each pair. The results are shown in Table 4. According to the experimental results, STCA-1 is greater than other methods. Because CSMF only considers spatial features, WSPred only thinks temporal features, and thus their prediction accuracy is obviously limited. This result is also consistent with the former observations of macro-indicators.

4.5. Parameters analysis

In the following sections, we conduct experiments to investigate how key configurations affect the prediction performance of STCA-1 (L_1), as it performs superior to other models.

4.5.1. Impact of dimensionality in embedding layer

Dimensionality of embedding vectors determines how many latent factors are utilized to characterize the spatial features and temporal feature in the embedding layer. To examine the impact of dimensionality on the performance, we vary the value of dimensionality d by 16 to 64 with a step value of 16 and let the matrix density equal to 1%, 2%, 3% and 4% respectively in our experiments. Fig. 7 shows the influence of dimensionality on prediction accuracy in term of MAE, RMSE and NMAE, the experiments are conducted on the response time dataset.

According to Fig. 7, the prediction accuracy increases along with the increment of dimensionality in term of MAE, RMSE and NMAE. This observation is coinciding with the intuition that a relatively larger value of dimensionality expresses more implicit information and generates much better prediction accuracy [4]. On the response time dataset, the prediction accuracy is dramatically improved as d increases from 16 to 32 and *MD* increase from 1% to 2%, while it changes gradually with the increase of d after $d > 32$ and $MD > 3\%$. And MAE, RMSE and NMAE are all optimal when $MD = 4\%$ and $d = 64$.

Table 4
The cases study on the spatio-temporal context-aware QoS prediction with MD=5%.

Cases of service invocations			$\hat{Q}_{xt}$				
Spatial features	Time slot	Q_{xt}	STCA-1 (L_1)	STCA-2 (L_1)	CSMF (L_1)	WSPred (L_1)	PMF (L_1)
uid=0, *sid*=727, *uip*=75.130.96.12, *ucity*=Worcester, *uas*=AS10326, *sip*=202.203.209.15, *scity*=Kunming, *sas*=AS4538	$t=0$	0.399	0.416	0.416	0.425	0.456	0.455
	$t=1$	0.398	0.415	0.426	0.421	0.503	0.430
uid=13, *sid*=49, *uip*=151.97.9.224, *ucity*=Catania, *uas*=AS137, *sip*=141.8.224.183, *scity*=Zurich, *sas*=AS40034	$t=1$	2.186	2.089	2.051	2.025	2.315	1.068
	$t=5$	2.478	2.559	2.705	2.378	2.269	1.092
uid=334, *sid*=486, *uip*=155.54.239.65, *ucity*=Murcia, *uas*=AS766, *sip*=52.45.115.20, *scity*=Ashburn, *sas*=AS14618	$t=10$	0.323	0.344	0.353	0.364	0.410	0.401
	$t=11$	0.308	0.330	0.337	0.365	0.406	0.367
uid=228, *sid*=771, *uip*=131.188.44.100, *ucity*=Erlangen, *uas*=AS680, *sip*=43.241.8.62, *scity*=Beijing, *sas*=AS9802	$t=20$	1.534	1.352	1.509	1.360	1.314	1.133
	$t=21$	1.621	1.350	1.442	1.342	1.309	1.195
uid=187, *sid*=933, *uip*=128.227.150.11, *ucity*=Gainesville, *uas*=AS6356, *sip*=81.30.226.144, *scity*=kraj, *sas*=AS15935	$t=1$	0.085	0.086	0.069	0.088	0.069	0.102
	$t=2$	0.086	0.072	0.071	0.063	0.092	0.070

Table 5
The performance variance of different models.

Methods	MD=1%			MD=2%			MD=3%			MD=4%		
	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE
PMF(L_2)	4.44E−05	1.28E−05	9.76E−08	3.00E−04	7.73E−05	1.54E−07	1.35E−04	2.70E−05	1.84E−08	4.86E−07	1.58E−07	9.60E−09
PMF(L_1)	1.26E−04	9.60E−05	2.46E−07	1.34E−03	1.89E−03	3.06E−06	2.00E−03	9.13E−04	4.41E−06	4.24E−05	9.53E−06	1.02E−07
WSPred(L_2)	2.43E−05	2.17E−05	6.64E−08	1.87E−05	5.34E−05	4.56E−08	1.65E−05	2.19E−05	3.44E−08	4.34E−05	8.57E−05	1.06E−07
WSPred(L_1)	4.26E−05	3.71E−05	9.36E−08	1.04E−04	1.64E−04	2.34E−07	6.78E−05	3.72E−05	1.50E−07	3.22E−05	3.61E−05	7.84E−08
CSMF(L_2)	1.17E−03	1.43E−04	3.76E−07	7.96E−04	5.19E−04	2.02E−07	6.48E−04	2.97E−05	1.56E−08	1.79E−04	8.68E−05	6.89E−08
CSMF(L_1)	5.44E−05	2.59E−06	1.27E−07	3.12E−05	1.85E−06	8.00E−08	2.03E−05	9.49E−07	4.67E−08	1.59E−05	1.09E−06	3.56E−08
STCA-2(L_2)	1.25E−05	2.96E−05	2.96E−08	4.66E−05	2.81E−05	9.84E−08	7.93E−06	6.65E−05	1.20E−08	1.48E−05	4.69E−05	3.76E−08
STCA-2(L_1)	5.79E−06	7.59E−06	1.76E−08	5.59E−06	1.58E−05	1.04E−08	2.77E−06	6.98E−05	5.60E−09	1.89E−06	2.43E−05	5.60E−09
STCA-1(L_2)	4.97E−06	5.39E−05	1.36E−08	5.58E−05	2.15E−05	1.32E−07	2.40E−05	1.86E−05	5.20E−08	2.83E−06	1.84E−05	1.04E−08
STCA-1(L_1)	5.34E−05	1.05E−04	1.18E−07	1.09E−05	1.02E−04	2.00E−08	3.13E−06	1.39E−05	5.60E−09	2.84E−06	1.11E−05	6.40E−09

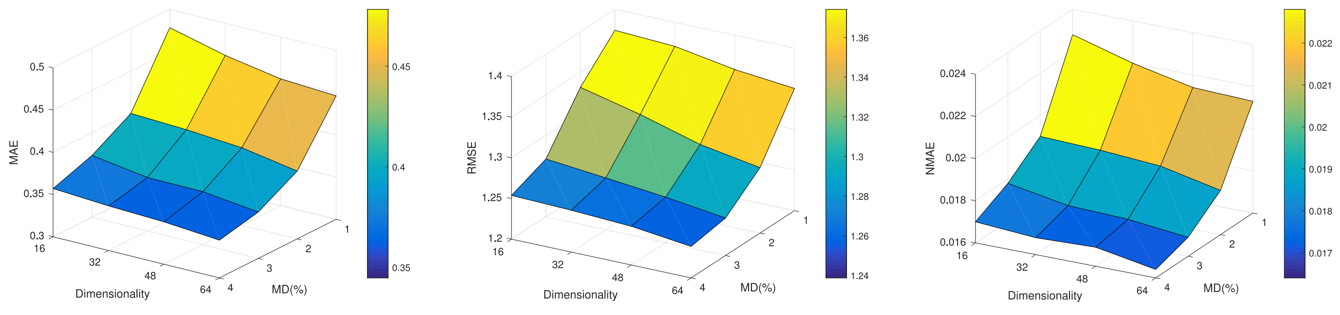


Fig. 7. Performance of STCA-1 (L_1) with different dimensionality and data density in term of MAE, RMSE and NMAE.

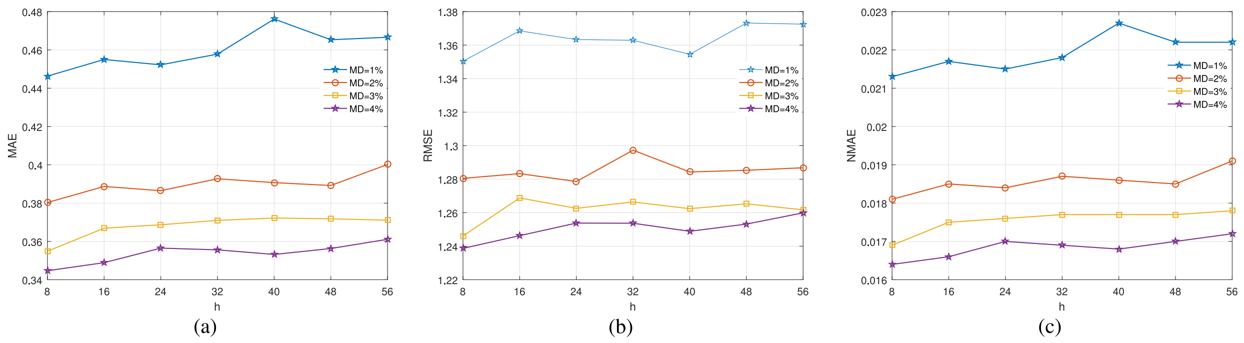


Fig. 8. Impact of hidden factors in attention mechanism on STCA-1 (L_1) in term of MAE, RMSE and NMAE.

4.5.2. Impact of hidden factors in attention mechanism

We conduct experiments to investigate the impact of hidden factor size h in attention mechanism on the accuracy of our models. Fig. 8 shows the prediction performance of STCA-1 (L_1) with different number of hidden factors and matrix densities. We vary the value of h by 8 to 56 with a step value of 8. The accuracy fluctuates with the change of h. More specifically, STCA-1(L_1) achieves the best performance when $h = 8$, while larger values like 16 can potentially reduce the prediction accuracy. Too many hidden factors will increase the difficulty of training

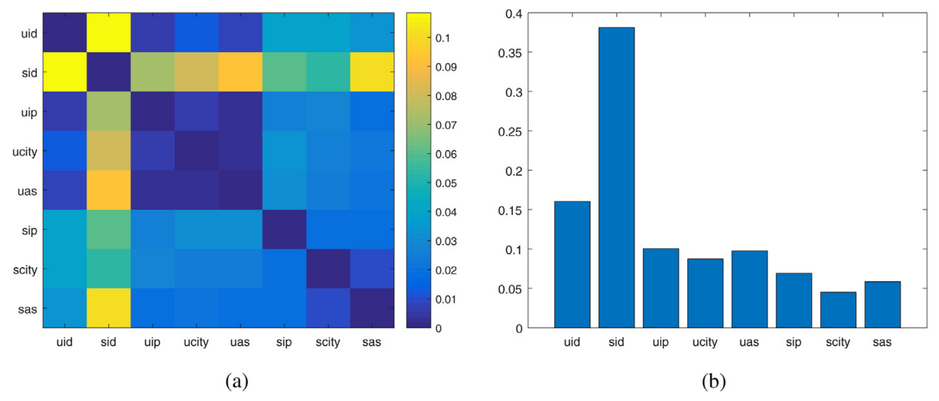


Fig. 9. The weights of second-order and first-order spatial features assigned through attention mechanism.

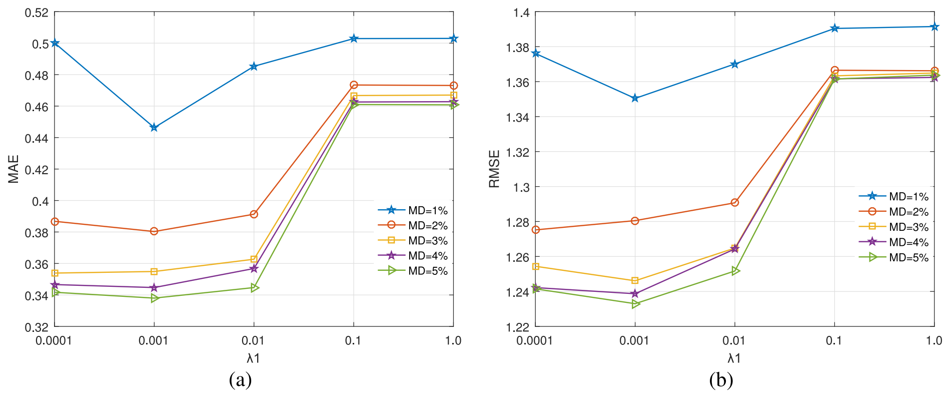


Fig. 10. Impact of regularization parameter in term of MAE and RMSE.

and result in slower convergence. It is worth noting that h may reflect the size of some visual field. The optimal value of h may change with the configurations of available contextual features and embedding dimensionality.

4.5.3. Visual analysis of impact of spatial features

In our works, attention mechanism is used to automatically assign weights to spatial features in each prediction, which plays a role in feature selection. In the process of prediction, the weight given to each feature is not invariable and will change dynamically with different prediction objectives. To analyze the influence of spatial features based on attention mechanism from a macro perspective, we randomly choose 1024 training records to average the assigned weights of features. We show the weights of the second-order spatial features by *heatmap* in Fig. 9a (The heatmap is symmetric about the main diagonal because the interactive feature of feature A and feature B and the interactive feature of feature B and feature A are equivalent.) and represent the weights of the first-order spatial features by *bar diagram* in Fig. 9b.

Service invocations occur not only the client hosts and the server hosts, but also across communication paths on the Internet, so the observed quality values are dependent on the configuration of hosts, the status of servers and the network conditions. The second-order spatial features are generated to model the interactions between client hosts and service hosts as well as the interactions between end-user environment and service environment. Depending on Fig. 9, the interaction between a user and a service (corresponds to second-order interactive features uid-sid) contributes the most to the prediction objective, followed by the second-order interactive features of sid-sas and sid-uas. The results accord with the intuitive observation. Because the interaction between uid and sid depicts more aspects of the interactive panorama. For instance, it may well describe a complete one-to-one communication path. Additional second-order features only describe part of the scene. For instance, the feature of sid-sas and sid-uas describes part of the communication path, respectively.

As for the first-order features, sid and uid have relatively greater influence, among which sid is more important. This is because the QoS values depend more on the service capabilities provided by service providers, which largely determine the quality of the customer's perception. Although other features play moderately weak roles, the use of these features will help to improve the prediction accuracy, while counteract the drawbacks of data sparsity. In addition, combined with Fig. 9a and Fig. 9b, network features (uip, uas, sip, sas) are more effective than geographic features (ucity, scity). After all, service invocations occur in cyberspace rather than physical space.

It is worth emphasizing that it is difficult to interpret the prediction results because of the black-box characteristics of neural networks [41]. However, interpretability is an important aspect of recommender systems as users expect a system to give a reason for its predictions, rather than facing black-box recommendations. Attention mechanisms allow us to give intuitive explanations of the effectiveness of feature fusion more reasonably and thus strengthen the interpretability of the prediction models.

4.5.4. Impact of regularization parameter

As for STCA-1(L_1), the use of L_1 regularizer can avoid overfitting problems, also play a role in feature selection to some extent. The parameter λ_1 is used to penalize larger values in the embedding vectors of features and the parameters of the attention modules. If λ_1 is approaching to zero, STCA-1(L_1) will

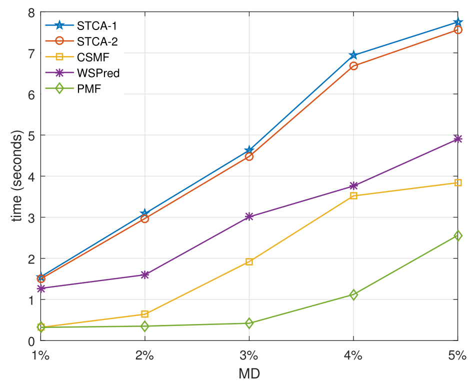


Fig. 11. The training overheads of different models in one iteration.

be often suffered from overfitting problem and exploding gradient problem. If λ_1 is increased, it means that the regularization terms will gradually affect the settings of parameters. With the increasing of λ_1, more values in the regularized parameters will approach zero to make the objective function as small as possible. This is why L_1 regularization induces sparsity in parameters and thus works in feature selection [37].

The experimental results are presented in Fig. 10 basically validate this principle. When λ_1 takes the value of 0.0001, the prediction performance in term of MAE and RMSE decreases to some extent because of over-fitting. When the value of parameter λ_1 increases, the positive effect of regularization gradually becomes prominent, we can observe that both MAE and RMSE achieve best results when λ_1 is set to 0.001. However, a larger λ_1 (e.g. $\lambda_1 > 0.01$) will make the parameters too sparse (many of them tend to be zero), resulting in under-fitting. At this time, model learning ability is insufficient to learn true distribution of data, resulting in high prediction bias.

4.6. Analysis of computational overheads

In this section, we observe the computational efficiency with respect to STCA-1 and STCA-2, and take PMF, CSMF, WSPred into account to make a comparison as all build upon machine learning. Generally, the computational cost of a machine learning model is dependent on the number of iterations and the number of model parameters. However, the number of iterations is determined largely by the learning rate or a presetting value, and thus we analyze the computational costs of these methods from the perspective of the number of parameters.

To indicate the number of parameters uniformly, we let n_u, n_s and n_t be the number of users, services and time slices respectively, and let n_c represents the number of spatial features, including the IP address, city and autonomous domain of users and services in our proposed models. As for PMF, it has $d(n_u+n_s)$ parameters. And WSPred has $d(n_u + n_s + n_t)$ parameters, CSMF has $d(n_u+n_s+n_c)$ parameters for the reason that CSMF has taken context features into consideration. Our proposed model STCA-1 and STCA-2 have more parameters than others because we adopt attention mechanisms and linear regression part of factorization machine. For the attention modules in STCA-1 model, there are $2(h \times d + 2h)$ parameters, and for linear regression component, there are $n_u + n_s + n_t + n_c$ parameters. Hence, there are total $(d+1)(n_u+n_s+n_t+n_c)+2(h\times d+2h)$ parameters. As for STCA-2 which has only one attention module, the number of parameters is $(d + 1)(n_u + n_s + n_t + n_c) + (h \times d + 2h)$.

To compare these methods intuitively, we further count their training costs in one iteration and display the results in Fig. 11 where everyone is optimized based on L_1 loss and L_1 regularizer. Specially, the time cost is accumulated by 32 time slots for PMF and CSMF. For all the models, the computational overheads are basically linear with the matrix density. Although our models are more complex than CSMF, WSPred and PMF, the overall efficiency gap is not as large as imagined, with the help of hardware acceleration such as GPU. In practice, both STCA-1 and STCA-2 spent about 20+ times of iterations to complete model training, about 50+ seconds to complete the task of predictions for more than ten million records in our experiments.

5. Conclusions and future works

We have proposed two neural models for the task of spatio-temporal context-aware QoS prediction. And intensive experiments and analysis indicate the effectiveness of our models, particularly our models achieve a performance improvement by about 10.9–21.0% in term of MAE and NMAE, and by 2.4–7.8% in term of RMSE in comparison with the baseline methods. The advantages of our works lie in the following points: (1) It realizes the integration of spatial–temporal contexts in the task of collaborative QoS prediction. (2) Hierarchical neural network structure is easy to add fresh features and advanced neural components to realize the prediction of tasks with different nature. (3) Attention mechanisms enable us to give intuitive explanations of the effectiveness of feature fusion more reasonably and thus strengthen the interpretability of the prediction models. (4) By optimizing the L_1 loss function with L_1 regularization, more accurate prediction results are achieved. The main shortcomings of our models lie in: (1) The models only consider the global correlation of the QoS data in time series, but ignore the local correlation; (2) The models are not robust enough, because they perform well under the condition of sparse data, but show relatively obvious performance degradation as the data becomes dense.

There are several directions to expand our works: (1) By applying new feature selection modules in the output layer of STCA models, it can fulfill synchronous prediction of multiple QoS attributes, such as response time and throughput; (2) The models can accurately fill in the tensor structure needed for dynamic QoS prediction, so it can realize the prediction of future QoS values by combining with time series analysis methods; (3) By subdividing spatial features and introducing multi-level structure into attention module, it is expected to achieve more fine selection and utilization of features; (4) Additionally, it is easy for researchers to employ our models for other prediction tasks like traffic flow prediction.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61562090, 61562092, U1802271), China Postdoctoral Science Foundation funded project (2016M592721), partially supported by the Project of Innovative Research Team of Yunnan Province (2018HC019). The authors would like to thank Dr. Weiwei Qiu from College of Computer Science and Technology, Zhejiang University, China for the contribution to the dataset. Besides that, the authors are grateful to reviewers for their constructive comments which contribute substantially to the improvement of this article. The authors declare that there is no conflict of interests regarding the publication of this article.

References

[1] A. Kobusińska, C.-H. Hsu, Towards increasing reliability of clouds environments with restful web services, Future Gener. Comput. Syst. 87 (2018) 502–513.

[2] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud computing and internet of things: a survey, Future Gener. Comput. Syst. 56 (2016) 684–700.

[3] W. Tsai, X. Zhou, Y. Chen, X. Bai, On testing and evaluating service-oriented software, IEEE Comput. 41 (8) (2008) 40–46.
[4] Z. Zheng, H. Ma, M.R. Lyu, I. King, Collaborative web service QoS prediction via neighborhood integrated matrix factorization, IEEE Trans. Serv. Comput. 6 (3) (2013) 289–299.
[5] Y. Koren, R.M. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Comput. 42 (8) (2009) 30–37.
[6] C.C. Aggarwal, Neighborhood-based collaborative filtering, in: Recommender Systems, Springer, 2016, pp. 29–70.
[7] H. Wu, Z. Zhang, J. Luo, K. Yue, C.-H. Hsu, Multiple attributes QoS prediction via deep neural model with contexts, IEEE Trans. Serv. Comput. (2018).
[8] H. Wu, K. Yue, B. Li, B. Zhang, C.-H. Hsu, Collaborative QoS prediction with context-sensitive matrix factorization, Future Gener. Comput. Syst. 82 (2018) 669–678.
[9] C. Yu, L. Huang, A web service QoS prediction approach based on time- and location-aware collaborative filtering, Serv. Orient. Comput. Appl. 10 (2) (2016) 135–149.
[10] Y. Zhang, Z. Zheng, M.R. Lyu, Wspred: A time-aware personalized QoS prediction framework for web services, in: Software Reliability Engineering, ISSRE, 2011 IEEE 22nd International Symposium on, IEEE, 2011, pp. 210–219.
[11] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, H. Mei, Personalized QoS prediction for web services via collaborative filtering, in: 2007 IEEE International Conference on Web Services, ICWS 2007, Salt Lake City, Utah, USA, 2007, pp. 439–446.
[12] Z. Zheng, H. Ma, M.R. Lyu, I. King, WSRec: A collaborative filtering based web service recommender system, in: 2009 IEEE International Conference on Web Services, ICWS 2009, Los Angeles, CA, USA, 2009, pp. 437–444.
[13] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. Zhou, Z. Wu, Predicting quality of service for selection by neighborhood-based collaborative filtering, IEEE Trans. Syst. Man Cybern. B 43 (2) (2013) 428–439.
[14] H. Sun, Z. Zheng, J. Chen, M.R. Lyu, Personalized web service recommendation via normal recovery collaborative filtering, IEEE Trans. Serv. Comput. 6 (4) (2013) 573–579.
[15] X. Chen, Z. Zheng, Q. Yu, M.R. Lyu, Web service recommendation via exploiting location and QoS information, IEEE Trans. Parallel Distrib. Syst. 25 (7) (2014) 1913–1924.
[16] M. Tang, T. Zhang, J. Liu, J. Chen, Cloud service qos prediction via exploiting collaborative filtering and location-based data smoothing, Concurr. Comput.: Pract. Exper. 27 (18) (2015) 5826–5839.
[17] Y. Xu, J. Yin, S. Deng, N.N. Xiong, J. Huang, Context-aware QoS prediction for web service recommendation and selection, Expert Syst. Appl. 53 (2016) 75–86.
[18] P. He, J. Zhu, Z. Zheng, J. Xu, M.R. Lyu, Location-based hierarchical matrix factorization for web service recommendation, in: Web Services, ICWS, 2014 IEEE International Conference on, IEEE, 2014, pp. 297–304.
[19] W. Lo, J. Yin, Y. Li, Z. Wu, Efficient web service QoS prediction using local neighborhood matrix factorization, Eng. Appl. AI 38 (2015) 14–23.
[20] D. Ryu, K. Lee, J. Baik, Location-based web service QoS prediction via preference propagation to address cold start problem, IEEE Trans. Serv. Comput. (2018).
[21] H. Ma, H. Zhu, Z. Hu, W. Tang, P. Dong, Multi-valued collaborative QoS prediction for cloud service via time series analysis, Future Gener. Comput. Syst. 68 (2017) 275–288.
[22] Y. Hu, Q. Peng, X. Hu, A time-aware and data sparsity tolerant approach for web service recommendation, in: 2014 IEEE International Conference on Web Services, ICWS, IEEE, 2014, pp. 33–40.
[23] X. Fan, Y. Hu, Z. Zheng, Y. Wang, P. Brezillon, W. Chen, CASR-TSE: Context-aware web services recommendation for modeling weighted temporal-spatial effectiveness, IEEE Trans. Serv. Comput. (2018) 1–1.
[24] W. Xiong, Z. Wu, B. Li, Q. Gu, A learning approach to QoS prediction via multi-dimensional context, in: Web Services, ICWS, 2017 IEEE International Conference on, IEEE, 2017, pp. 164–171.
[25] S. Wang, Y. Ma, B. Cheng, R. Chang, et al., Multi-dimensional QoS prediction for service recommendations, IEEE Trans. Serv. Comput. (2016).
[26] X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, M.R. Lyu, A spatial-temporal QoS prediction approach for time-aware web service recommendation, TWEB 10 (1) (2016) 7:1–7:25.
[27] X. He, T.-S. Chua, Neural factorization machines for sparse predictive analytics, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2017, pp. 355–364.
[28] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, J. Wang, Product-based neural networks for user response prediction, in: Data Mining, ICDM, 2016 IEEE 16th International Conference on, IEEE, 2016, pp. 1149–1154.
[29] S. Rendle, Factorization machines, in: Data Mining, ICDM, 2010 IEEE 10th International Conference on, IEEE, 2010, pp. 995–1000.
[30] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, arXiv preprint arXiv:1409.0473.
[31] A. Show, Tell: Neural image caption generation with visual attention, Kelvin Xu et al., 83 (2015) 89, arXiv pre-print.
[32] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, T.-S. Chua, Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2017, pp. 335–344.
[33] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T.-S. Chua, Attentional factorization machines: Learning the weight of feature interactions via attention networks, 2017, arXiv preprint arXiv:1708.04617.
[34] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, 2014, arXiv preprint arXiv:1406.1078.
[35] Y. Goldberg, A primer on neural network models for natural language processing, J. Artificial Intelligence Res. 57 (2016) 345–420.
[36] X. Yan, X. Su, Linear Regression Analysis: Theory and Computing, World Scientific, 2009.
[37] A.Y. Ng, Feature selection, L1 vs. L2 regularization, and rotational invariance, in: Proceedings of the Twenty-First International Conference on Machine Learning, ICML '04, ACM, New York, NY, USA, 2004, p. 78.
[38] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: a system for large-scale machine learning, in: OSDI, vol. 16, 2016, pp. 265–283.
[39] Z. Zheng, H. Ma, M.R. Lyu, I. King, QoS-Aware web service recommendation by collaborative filtering, IEEE Trans. Serv. Comput. 4 (2) (2011) 140–152.
[40] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Advances in Neural Information Processing Systems, NIPS2007, vol. 20, Vancouver, British Columbia, Canada, 2007, pp. 1257–1264.
[41] W. Samek, T. Wiegand, K.-R. Müller, Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models, 2017, arXiv preprint arXiv:1708.08296.

Qimin Zhou received the B.Sc. degree in computer science from Yunnan University, Kunming, China, in 2017. She is currently working toward the postgraduate degree at Yunnan University. Her current research interests mainly include deep learning and service computing.

Hao Wu received the bachelor's degree in computer science from Zhengzhou University, in 2001, master and Ph.D. degrees in computer science from Huazhong University of Science and Technology in 2004 and 2007, respectively. Now, he is an associate professor at School of Information Science and Engineering, Yunnan University, China. He has published more than fifty papers in peer-reviewed international journals and conferences. His research interests include service computing, information filtering and recommender systems.

Kun Yue received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Yunnan University (Kunming, China), Fudan University (Shanghai, China) and Yunnan University (Kunming, China) in 2001, 2004 and 2009, respectively. He is currently a professor at Yunnan University, Kunming, China. His current research interests mainly include massive data analysis and uncertainty in artificial intelligence.

Ching-Hsien Hsu is a Professor in the department of computer science and information engineering at National Chung Cheng University, Taiwan; His research includes high performance computing, cloud computing, parallel and distributed systems, big data analytics, ubiquitous/pervasive computing and intelligence. He has published 100 papers in top journals such as IEEE TPDS, IEEE TSC, IEEE TCC, IEEE TETC, IEEE System, IEEE Network, ACM TOMM and book chapters in these areas. Dr. Hsu is serving as editorial board for a number of prestigious journals, including IEEE TSC, IEEE TCC. He has been acting as an author/co-author or an editor/co-editor of 10 books from Elsevier, Springer, IGI Global, World Scientific and McGraw-Hill. Dr. Hsu was awarded nine times distinguished award for excellence in research from Chung Hua University. He is vice chair of IEEE TCCLD, executive committee of IEEE TCSC, Taiwan Association of Cloud Computing, a Fellow of IET and an IEEE senior member.