Location-aware Deep Collaborative Filtering for Service Recommendation

Yiwen Zhang, Chunhui Yin, Qilin Wu, Qiang He, *Member*, *IEEE* and Haibin Zhu, *Senior Member*, *IEEE*

***Abstract*—With the widespread application of service-oriented architecture (SOA), a flood of similarly functioning services have been deployed online. How to recommend services to users to meet their individual needs becomes the key issue in service recommendation. In recent years, methods based on collaborative filtering (CF) have been widely proposed for service recommendation. However, traditional CF typically exploits only low-dimensional and linear interactions between users and services and is challenged by the problem of data sparsity in the real-world. To address these issues, inspired by deep learning, this paper proposes a new deep collaborative filtering model for service recommendation, named LDCF (Location-aware Deep Collaborative Filtering). This model offers the following innovations: 1) the location features are mapped into high-dimensional dense embedding vectors, 2) the Multi-Layer-Perceptron (MLP) captures the high-dimensional and non-linear characteristics, and 3) the similarity Adaptive Corrector (AC) is first embedded in the Output Layer to correct the predictive quality of service. Equipped with these, LDCF can not only learn the high-dimensional and non-linear interactions between users and services, but also significantly alleviate the data sparsity problem. Through substantial experiments conducted on a real-world web service dataset, results indicate that LDCF's recommendation performance obviously outperforms nine state-of-the-art service recommendation methods.**

***Index Terms*—Service recommendation, collaborative filtering, deep learning, similarity adaptive corrector**

Manuscript received at February 15, 2019; this work is partly supported by the National Natural Science Foundation of China (No. 61872002), Australian Research Council Discovery Project (No. DP180100212), the Anhui Key Research and Development Plan (No.201904a05020091) and the Natural Science Foundation of Anhui Province of China (No. 1808085MF197). Corresponding author: Qilin Wu.

Y. Zhang and C. Yin are with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. E-mail: zhangyiwen@ahu.edu.cn; yinchunhui.ahu@gmail.com.

Q. Wu is with the School of Information Engineering, Chaohu University, Chaohu, Anhui and the School of Management and Engineering, Nanjing University, Nanjing, China. E-mail: lingqiw@126.com.

Q. He is with the School of Software and Electrical Engineering, Swin-burne University of Technology, Melbourne, Australia and the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. Email: qhe@swin.edu.au.

H. Zhu is with the Department of Control and System Engineering, Nanjing University, Nanjing 210093, China and the Department of Computer Science and Mathematics, Nipissing University, North Bay, ON P1B 8L7, Canada. Email: haibinz@nipissingu.ca.

I. Introduction

With the advent of the era that everything is service (e.g., cloud services, micro-services, Internet of Things (IoT) services, etc.), services deployment is proceeding at a rapid pace. How to recommend services to users that meet their individual needs has become a critical and challenging issue. Growing amounts of data support the idea that users are more inclined to choose services that satisfy their personal preferences, drawing interest from researchers examining the development of service recommendations based on contextual information of users and services.

Predicting Quality of Service (QoS) is the primary challenge in service recommendations. Among the existing QoS prediction methods, collaborative filtering (CF) is the most widely used [1-4]. Some researches seek to combine time, trust, location and other contextual information to improve recommendation performance when applying CF technology to QoS prediction. However, traditional CF technologies have the following two shortcomings: 1) the similarity calculation method employed by traditional CF-based methods can only learn the low-dimensional and linear characteristics from the past interactions between users and services, and 2) the common data sparsity problem in the real-world significantly impacts their recommendation performance.

Some efforts have been devoted to the combination of deep neural networks with CF with the aim to overcome the limitations of CF [5, 6]. He *et al*. [5] combined Matrix Factorization (MF) with the Multi-Layer Perceptron (MLP) in deep learning, and proposed the neural collaborative filtering (NCF) framework to overcome the limitation of MF in low-dimensional latent spaces. The deep matrix factorization (DMF) framework proposed by Xue *et al*. [6] can extract features directly from the user-service interaction matrix, and consider explicit rating and implicit feedbacks for making Top-K recommendations. The existing research has raised two issues: 1) only the correlation between the user and the service is studied, but the robustness of the method is ignored; and 2) using only the identifier information of the user and the service does not reflect the location correlation between the user and the service.

This paper proposes the Location-aware Deep Collaborative Filtering (LDCF) model, which not only has strong robustness, but also reflect the location correlation between the user and the service. The main contributions of this work are as follows:

1) We propose the LDCF model that innovatively integrates Multi-Layer-Perceptron with a similarity Adaptive Corrector, designed to learn the high-dimensional and non-linear interactions and the location correlation between users and services.

2) We first introduce the Huber loss function in this model, which has strong robustness and achieves excellent performance on all evaluation metrics. Thus, LDCF has good adaptability and extensibility in exploiting contextual information such as locations.

3) Experiments have been conducted to evaluate the performance of our approach and compare it with nine other state-of-the-art alternatives. Results indicate that our approach not only achieves better recommendation performance, but also greatly alleviates problems caused by data sparsity.

The remainder of this paper is organized as follows. Related work is described in Section II. Section III supplies the motivation for this work. Section IV discusses the architecture of our proposed model. Section V presents experimental results and analysis. Conclusions appear in Section VI.

II. Related work

This section reviews related works based on traditional CF and the latest deep learning methods.

A. Collaborative Filtering Based Methods

The CF-based methods use historical information to recommend services for potential users. The CF-based method for service recommendation has been widely studied since the first use of CF by Shao *et al*. [7] for predicting QoS. CF can be further characterized as memory-based or model-based.

The memory-based approach includes user-based [8], item-based [2], and a combination of the two [9]. One of the main tasks of this CF technique is to predict missing QoS values for target users. The key step is to perform similarity calculations on users or items. In order to more accurately calculate the similarity of users or services, many improved memory-based works were proposed. For example, Wu *et al*. [10] proposed a neighborhood-based CF approach called ADF, in which the A-cosine approach, the data smoothing process, and the similarity fusion approach are adopted. Zhang *et al*. [11] combined the covering-based clustering algorithm with MF, and proposed a Covering-based via Neighborhood-aware MF (CNMF) method to fully utilize neighborhood information in service recommendations. To ensure the correct execution of the resulting composite service, Wang *et al.* [12] proposed a solution that included a graph search-based algorithm and two novel preprocessing methods. The concept of Generalized Component Services (GCSs) proposed by Wu *et al.* [13] is defined in a semantic manner to expand the scope of service selection. Ding *et al.* [14] addressed the issue of selecting and composing web services via a genetic algorithm (GA) and offered a QoS-aware selection approach. Wu *et al.* [15] proposed a ratio-based approach to calculate similarity to recommend services.

To further improve the accuracy of similarity calculations in memory-based CF methods, many researchers have begun to focus on contextual information, such as reliability, time, locations, and so on. For example, Chen *et al.* [16] considered the user's trust value and location for QoS prediction before the similarity calculation. Zheng *et al.* [17] proposed two personalized reliability predictions, which use past fault data to predict Web service failure probability. Hu *et al.* [18] used time information to improve similarity calculation for predicting QoS. Tang *et al.* [19] improved the accuracy of QoS prediction by integrating the locations of users and services into traditional similarity calculations. Liu *et al.* [20] proposed a Location-aware collaborative filtering method, which uses the locations of users and services to effectively improve recommendation performance. Tang *et al.* [21] proposed a network-aware method called NAMF for service recommendation by integrating MF with the network map. However, when facing a large amount of data, memory-based CF methods cannot propose recommendations in real-time due to the complexity of calculations involved.

Fortunately, model-based CF methods effectively solve this problem. For instance, Zhang *et al.* [22] proposed a WSPred model with embedded time information for predicting QoS. Yang [23] introduced location information into the Factorization Machine (FM) for QoS prediction. Although the contextual information contributes to the similarity calculation of CF, this kind of calculation can only learn the low-dimensional and linear features of users and services. When facing the real-world problem of data sparsity, feature learning is insufficient, thereby limiting recommendation performance. To address this issue, our proposed method uses MLP to capture the complex high-dimensional and non-linear relationships between users and services.

B. Deep Learning Based Methods

To the best of our knowledge, He *et al.* [5] first applied deep learning techniques to the field of recommendation systems. They proposed the NCF model, which solves the problem of poor representation of MF in low dimensions. Subsequently, many methods have been proposed, such as the DMF model proposed by Xue *et al.* [6], which extracts features directly from the user-item matrix as neural network inputs. This takes into account explicit and implicit feedback for Top-K recommendation. He *et al.* [24] proposed ConvNCF that used the Convolutional Neural Network (CNN) to study the high-dimensional correlation between local and global embedding dimensions in a hierarchical manner for Top-K recommendation. Kim *et al.* [25] proposed the ConvMF model to combine CNN with Probability Matrix Factorization (PMF) for QoS prediction. Wu *et al.* [26] developed a novel neural architecture CNSR that jointly incorporates the social network structure and user–item interaction in a unified model for social recommendations. Xiong *et al.* [27] proposed an online method based on Personalized Long and Short Time Memory Network (PLSTM), which can capture dynamic implicit feature representations of multiple users and services, and update the prediction model in time to process new data. Most of these studies rely mainly on the user identifier and the item identifier to achieve good performance in the field of film

recommendation.

Recently, Xiong *et al.* [28] proposed a Deep Hybrid Service Recommendation (DHSR) model that integrates MLP and text similarity to learn the non-linear relationship between mashups and services. Bai *et al.* [29] used denoising autoencoders (SDAE) to construct a deep learning framework DLTSR to solve the long tail network problem of service recommendation. Yuan *et al.* [30] proposed a deep learning model for healthcare service recommendation, which embeds the trust relationship and distrust relationship of the target user. It is worth noting that the above work often uses the user's or service's identifier information, but rarely considers location information that may be closely related to the quality of service. In contrast to the existing research, our method addresses the problem by embedding the similarity Adaptive Corrector of the user location and service location.

III. Motivation

In this section, we illustrate the motivation of our research according to Figs. 1 and 2. Specifically, Part A discusses why locations should be introduced and Part B analyzes the necessity of applying deep learning techniques.

A. Why Include Locations?

Developments in cloud and edge computing have given rise to a hybrid platform based on the edge infrastructure. This has become the focus of attention for many researchers. The Content Distribution Network (CDN) is an important part of this platform. It relies on edge servers, deployed in the local area, to enable personalized nearby user services through content distribution, load balancing and other technologies. This is done to alleviate network congestion, along with improved unified coordination and service capabilities, to enhance the user experience.

Fig. 1 shows a location-aware service recommendation scenario. The figure includes three users: u_1, u_2, u_3, one CDN central server c_0 and three edge servers: s_1, s_2, s_3 with coverage areas *region* 1, *region* 2, and *region* 3, respectively. Orange dotted lines represent data packet transmission paths. Black oval dotted lines represent edge server coverage area. Our goal is to examine the impact on QoS of location correlation between target users and services and then recommend suitable Maps services to users.

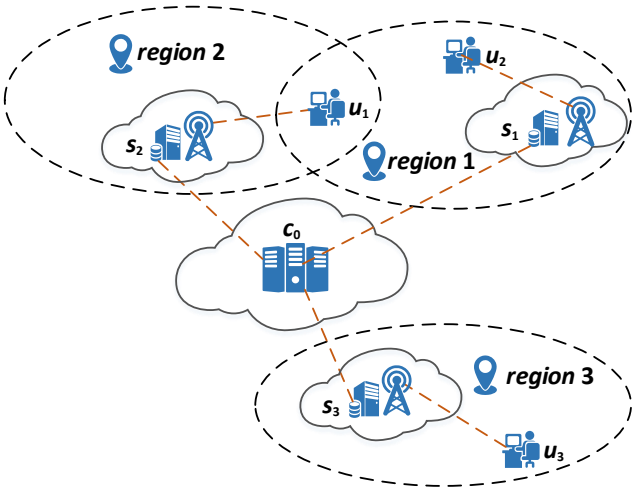


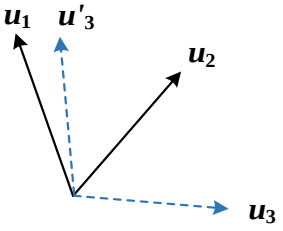
Fig. 1. A location-aware service recommendation scenario

QoS is largely dependent on bandwidth and the network distance between user and cloud server. Users can experience better QoS by calling services that are geographically close to them. As shown in Fig. 1, u_1, u_2, and u_3 send requests to c_0 to call Google Maps service. User u_1 is within the coverage areas of both s_1 and s_2. Since u_1 is closer to s_2, c_0 can use the global load balancing strategy to point u_1's access to s_2 instead of s_1. Improving user experience can be achieved by considering regional differences between them and services.

B. Why Use Deep Learning?

	s_1	s_2	s_3
u_1	0.70	1.63	0.33
u_2	0.54	0.31	0.71
u_3	2.47	0.04	0.93

(a)



(b)

Fig. 2. An example of CF similarity calculation

Fig. 2 illustrates how the similarity calculation limits the effectiveness of CF. CF-based methods employ similarity calculation for service recommendation based on similarity measurements such as cosine similarity, Euclidean similarity, Pearson correlation coefficient, etc. This limits the ability of CF-based methods in mining features effectively. Fig. 2 exemplifies this limitation with the cosine similarity.

From the above user-service innovation matrix presented in Fig. 2, we can obtain user u_1 and u_2's feature vectors: u_1= [0.70, 1.63, 0.33], u_2= [0.54, 0.31, 0.71]. The cosine similarity between u_1 and u_2 is: Sim (u_1, u_2) = 0.66. Fig. 2(b) demonstrates their geometric relationship in a 2D space. Let us assume a new user u_3 = [2.47, 0.04, 0.93]. There is Sim (u_1, u_3) = 0.44 < Sim (u_1, u_2) = 0.66 < Sim (u_2, u_3) = 0.81. This indicates that u_3 is more similar to u_2 than u_1. However, if a CF-based method places u_3 as the closest user to u_1 as demonstrated in Fig. 1(b), u_3 will be closer to u_1 than u_2, i.e., Sim (u_1, u'_3) > Sim (u_2, u'_3). This will lead to inaccuracy and misjudgment in user similarity evaluation. A similar issue has been raised and resolved in work [5]. To address this issue, in this work, we leverage the ability of deep learning to extract features effectively [31].

IV. Proposed Model

In this section, we first introduce the model, then describe its components. After that, we provide an explanation of the loss function and optimizer parameters applied in the model.

A. *Location-aware Deep Collaborative Filtering*

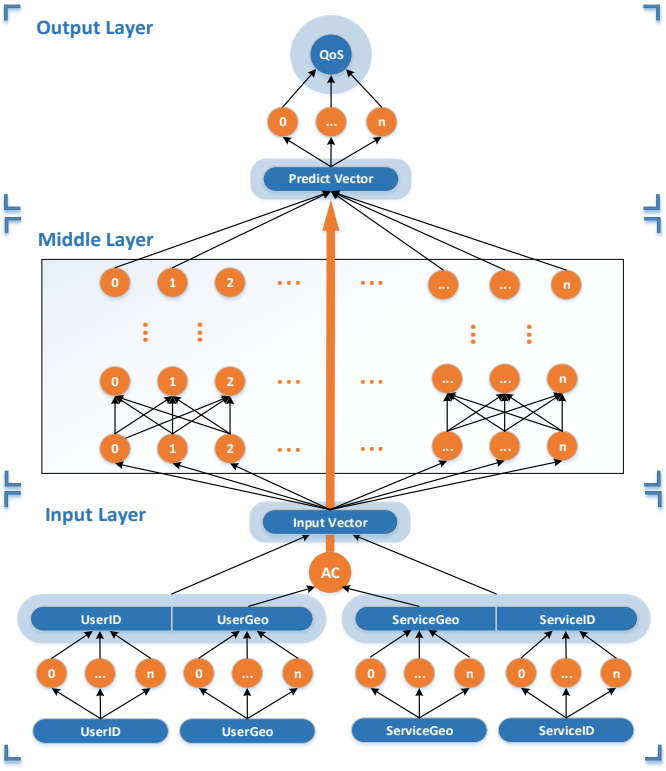


Fig. 3. Location-aware Deep Collaborative Filtering

As shown in Fig. 3. The LDCF architecture is a multi-layer feedforward neural network that includes three specific functional layers, i.e., the Input Layer, the Middle Layer, and the Output Layer. In its forward propagation process, the output of each layer is used as the input of the next layer. For example, we use the Input Layer to generate the input vectors required by the Middle Layer and the similarity required by the Output Layer. The Middle Layer is used for centralized training to obtain high-dimensional and non-linear features.

The basic meanings of each component in Fig. 3 are as follows: the orange circle represents the computing node (or calculation unit), which includes all neurons of the deep neural network and an Adaptive Corrector that calculates the similarity; the arrows represent data flow; the light-colored rounded oval rectangle represents the merge operation. Each functional layer will be described in detail below.

B. *Input Layer*

The Input layer is primarily used to process the original input. For neural networks to learn additional data characteristics, we input the user identifier, user's location information, service identifier, and service's location information into the Embedding Layer of Keras[1], which can be regarded as a special fully-connected layer without bias term. Specifically, Embedding performs one-hot encoding on the input to generate a zero vector with a specified dimension and the *i*-th position of the vector will be set to 1 [32]. Similar to Word2vec, Doc2vec, and GloVe, our embedding method uses dense vectors to represent words or documents, similar to Natural Language Processing [33-35]. Through this operation, the categorical features are mapped to the high-dimensional dense embedding vectors. The mapping process is shown in Equations (1) - (4):

$$I_u^k = f_1(P_1^T i_u + b_1) \tag{1}$$

$$G_u^k = f_1(P_1^T g_u + b_1) \tag{2}$$

$$I_s^k = f_1(Q_1^T i_s + b_1) \tag{3}$$

$$G_s^k = f_1(Q_1^T g_s + b_1) \tag{4}$$

where i_u and i_s , represent the user's and the service's identifier , respectively; g_u and g_s the original inputs of the user's and service's location; P_1 the user's embedding weight matrix; Q_1 the service's embedding weight matrix; b_1 the bias term initialized to zero; f_1 the activation function of this layer; and the standard identity function is selected in the paper. I_u^k and G_u^k are the *k*-dimensional user's identifier embedding vector and location embedding vector, respectively. Similarly, U and S are the *k*-dimensional service's identifier embedding vector and location embedding vector, respectively.

Finally, we combine identifier feature vector with the corresponding location feature vector to obtain user feature vector and service feature vector respectively. Then we concatenate these two feature vectors to get the input vector required for the middle layer. The formula is expressed as follows:

$$U = \Phi(I_u^k, G_u^k) = \begin{bmatrix} I_u^k \\ G_u^k \end{bmatrix} \tag{5}$$

$$S = \Phi(I_s^k, G_s^k) = \begin{bmatrix} I_s^k \\ G_s^k \end{bmatrix} \tag{6}$$

$$x = \Phi(U, V) = \begin{bmatrix} U \\ V \end{bmatrix} \tag{7}$$

where Φ represents the mergence operation, U and S the embedding vector of a user and a service, and x the input vector.

Here, we propose an Adaptive Corrector (AC), which performs similarity calculation between user location embedding and service location embedding. In recent years, many CF-based methods [11, 21, 23] have integrated user location similarity and service location similarity into Matrix Factorization to improve prediction accuracy. AC shares the similar methodology by integrating location similarity between users and services into the forward propagation process in the neural network. In this way, AC helps bridge the gap between deep learning and collaborative filtering. As shown in Fig. 3, the operation result of the AC is directly transmitted to the Output Layer without the Middle Layer. The AC can be adaptive and can be adapted to various similarity calculations such as cosine similarity and Euclidean similarity. The formula is expressed as either (8) or (9):

[1] https://keras.io/

$$o^{AC} = \text{cosine}(\mathrm{G}_u, \mathrm{G}_s) = \frac{\mathrm{G}_u \cdot G_s}{\|\mathrm{G}_u\| \|G_s\|} \quad (8)$$

$$o^{AC} = \text{euclidean}(\mathrm{G}_u, \mathrm{G}_s) = \sqrt{\mathrm{G}_u \cdot \mathrm{G}_s} \quad (9)$$

where o^{AC} represents the similarity output of AC, here the cosine similarity or the Euclidean similarity is available. If no special statement is presented, we use Equation (8) to obtain the cosine similarity result of AC in the experiments.

C. Middle Layer

The Middle Layer is used to process the input vector from the Input layer for capturing the non-linear features.

In this paper, a fully connected Multi-Layer Perceptron (MLP) structure is used to learn the high-dimensional non-linear relationship between users and services. First of all, we should choose the activation function. Through experiments, we found that the activation function of the Rectified Linear Unit (ReLu) has many advantages. For example, it can accelerate the convergence of a model and solve the problem of the disappearance of the sigmoid function gradient in the saturation zone. Furthermore, Relu is one-sided compared with the anti-symmetry of tanh, thus it has more biological plausibility. Therefore, ReLu is chosen as the activation function of the Middle Layer. Secondly, in order for the neural network to learn more features, the network architecture needs to follow the typical tower structure, i.e., the more the bottom neurons, the lower the top levels [5]. Finally, we use L2 regularization on weight to prevent overfitting. The forward propagation process of the input vector in the Middle layer is defined as follows:

$$o_2^{mlp} = f_2(W_2^T x + b_2) \quad (10)$$

$$o_i^{\text{mlp}} = f_i(W_i^T o_{i-1}^{mlp} + b_i), i = 3, 4, ..., n\text{-}1 \quad (11)$$

$$o^{mlp} = o_{n\text{-}1}^{mlp} \quad (12)$$

where o_i^{mlp} denotes the *i*-th layer of the Middle Layer' output, W_i the corresponding weight matrix, b_i the bias term corresponding to the Middle layer, and o^{mlp} the output of the Middle Layer.

D. Output Layer

The Output Layer is primarily used to generate the final prediction result. LDCF models users and services in two pathways. Inspired by [5, 6], we directly concatenate on the outputs of these two pathways. We combine the similarity output o^{AC} with o^{mlp} to construct a new output vector o. Lastly LDCF generate final predictions via a single-layer neural network. Since the output is a specific value, it can be regarded as a regression problem, and the identity function is also selected as the activation function. The parameter initialization of this layer uses Gaussian distribution, as shown in Equations (13) - (14):

$$o = \Phi(o^{AC}, o^{mlp}) = \begin{bmatrix} o^{AC} \\ o^{mlp} \end{bmatrix} \quad (13)$$

$$\hat{Q}_{u,s} = f_n(W_n^T o + b_n) \quad (14)$$

where $\hat{Q}_{u,s}$ represents the predictive QoS value of user u invoking service s, and f_n is a standard identity function that represents the activation function of the last layer of this layer.

E. LDCF Learning

In supervised learning, the learning of the neural network model can be considered as a process of comparing the predictive results with the real values and then continuously optimizing the target loss function to achieve a final fit. The selection of the loss function and optimizer has a non-negligible effect on the performance of the algorithm. In this section, we mainly describe the loss functions and optimizer used in the LDCF model.

1) Loss Function Selection

The loss functions currently applied in mainstream recommendation systems can be divided into two types: pointwise and pairwise. A pointwise loss function converts the recommendation problem into a multi-classification problem or regression problem, while a pairwise loss function converts the recommendation problem into a binary classification problem. According to applications, the loss functions of pointwise (e.g., root-mean-square loss, log loss, etc.) can be further divided into regression-based, classification-based and ordinal regression-based, and pairwise loss functions include BPR [36], AUC and so on. The LDCF model predicts the value of QoS and belongs to the regression problem. Thus, the binary cross information entropy [5, 6, 28] is no longer suitable for our model. The commonly used loss functions of pointwise for regression are square loss, absolute loss, and so on. In statistical theory, an absolute loss function is not differential at a specific point (origin), and may lead to an unbiased estimation of the arithmetic average. The square loss function is extremely sensitive to outliers and easily leads to a median unbiased estimation. In order for the LDCF model to perform well across all evaluation metrics, we have chosen the Huber loss function [37] that combines the advantages of the former two. In statistics, Huber loss is a loss function used in robust regression, and is less sensitive to outliers in data analysis than the squared error loss[2]. The Huber loss function is defined as follows:

$$L_\delta(Q_{u,s}, \hat{Q}_{u,s}) = \begin{cases} \frac{1}{2}(Q_{u,s} - \hat{Q}_{u,s})^2 & for \ \left|Q_{u,s} - \hat{Q}_{u,s}\right| \le \delta, \\ \delta\left|Q_{u,s} - \hat{Q}_{u,s}\right| - \frac{1}{2}\delta^2 & otherwise. \end{cases} \quad (15)$$

where $Q_{u,s}$ is the original QoS value of user u invoking service s, $\hat{Q}_{u,s}$ is the predictive QoS value of user u invoking service s, and δ is a threshold for switching and δ is set to 1.0 in this paper.

2) Optimizer Selection

The mini-batch Adaptive Moment Estimation (Adam) [38] optimizer has the advantages of high computational efficiency, smaller memory requirement, and strong interpretability, etc. Adam comprehensively considers the first moment estimation

[2] https://en.wikipedia.org/wiki/huber_loss

and the second moment estimation of the gradient to update the step size and effectively combines the advantages of the two optimization algorithms AdaGrad [39] and RMSProp [40]. Therefore, we choose the Adam optimizer.

3) Complexity Analysis

We further analyze the time complexity of the proposed approach. The pseudo code of the proposed algorithm is presented below.

Algorithm 1: LDCF Algorithm.

Input: user-service invocation matrix R,
matrix density d,
neural network topology structure t,
learning rate l, decay ratio r,
number of iteration i.

Output: Weight matrices and bias terms P_1, Q_1, W_2, W_3, …, W_n, b_1, b_2, …, b_n.

1. sparse R according to d;
2. generate training entries R_{train} and test entries R_{test};
3. generate input features i_u, g_u, i_s, g_s;
4. build neural networks according to t and Eq. (10)-(12);
5. initialize P_1, Q_1, W_2, …, W_n according to Gaussian distribution;
6. initialize b_1, b_2, …, b_n to 0;
8. **for** epoch = 1, 2, …, i do
9. **for** user and service in R_{train} do
10. generate embedding vectors through Eq. (1)-(4);
11. generate input vector through Eq. (5)-(7);
12. generate AC output through Eq. (8) or (9);
13. generate prediction $\hat{Q}_{u,s}$ through Eq. (13)- (14);
14. **end for**
15. pass l and r to Adam;
16. update model parameters by Adam minimizing Eq. (15);
17 **for** user and service in R_{test} do
18. evaluate model performance through Eq. (18)-(19);
19 **end for**
20. **end for**

In the LDCF algorithm, the time complexity of line 10 is $O(k)$, where k represents the dimension of the embedding vectors. The time complexity of lines 11 is $O(1)$ because the concatenation is conducted. The time complexity of line 12 is $O(l_u \times l_s \times k)$, where l_u and l_s represent the length of user location features and service locations feature respectively, k represents dimension of embedding vectors. Among these parameters, l_u and l_s are constants. The time complexity of line 13 is $O(1)$. Then, the algorithm repeats lines 10-13 until R_{train} is traversed. Therefore, the time complexity of the forward propagation process (lines 9-14) is $O(n) \times (O(k) + O(1) + O(l_u \times l_s \times k) + O(1)) = O(n \times k)$, where n is the number of entries in R_{train}.

The time complexity of line 15 is $O(1)$. The time complexity of line 16 is $O(n \times k)$ because the time complexity of backpropagation is the same as the forward propagation. The time complexity of line 18 is $O(1)$. Line 18 needs to be repeated until R_{test} is traversed. Thus, the time complexity of lines 17-19 is $O(m)$, where m is the number of entries in R_{test}. The time complexity of lines 15-19 is $O(1) + O(n \times k) + O(m) = O(n \times k)$. Then, lines 9-19 need to be repeated until all iterations are completed.

Overall, the time complexity of LDCF is $O(i) \times (O(n \times k) + O(n \times k)) = O(i \times n \times k)$, where i is the total number of iterations.

V. Experiments

In this section, we conduct extensive experiments aimed at answering the following research questions:

1) ***RQ1***: Can the LDCF model alleviate the data sparsity problem compared to existing classic recommendation algorithms? Is there a significant improvement in recommendation performance?

2) ***RQ2***: Can the introduction of locations be helpful for the learning of the LDCF model?

3) ***RQ3***: Can deep-learning acquire high-dimensional and non-linear characteristics of users and services? Can the depth help recommend performance?

4) ***RQ4***: Can the similarity Adaptive Corrector (AC) help improve performance? Can it be adaptable and extensible?

5) ***RQ5***: Can the Huber loss function achieve excellent performance?

A. Dataset

We conducted experiments on the WS-Dream dataset, a large-scale real-world Web services dataset collected and maintained by Zheng [4] *et al.*, which contains 1,974,675 QoS values of Web services collected from 339 users on 5,825 services. The dataset provides location information about users and services (such as countries, etc.). In this paper, the QoS dataset is represented in the form of a user-service matrix, where the row index represents the user identifier, the column index represents the service identifier, and each value in the matrix is described by the response time (RT) and throughput (TP). In the experiment, we used RT and TP as the input to LDCF.

B. Pre-processing

The network unit with the same Autonomous System Number (ASN) commonly has similar network environments [20]. This paper uses two geographically related attributes provided by the datasets: CN (Country Name) and ASN. Through dataset statistics, users are distributed among 30 countries and 136 autonomous systems, while services are distributed among 990 autonomous systems in 73 countries. For CN, we use the categorical encoding of the Sklearn[3] to transform the classified features into integer encodings, so that each classification feature is represented as the national code. For ASN, we use its numeric coding directly. After pre-processing, the data can be represented as:

$$U = (UID, UASN, UCN) \tag{16}$$

$$S = (SID, SASN, SCN) \tag{17}$$

where U and S indicate the input embedding vector of a user and a service, respectively. The UID indicates the identifier of the user, the UASN indicates the autonomous system numerical

[3] https://scikit-learn.org

TABLE I
EXPERIMENTAL RESULTS OF RESPONSE-TIME

Methods	*density*=0.05		*density*=0.10		*density*=0.15		*density*=0.20		*density*=0.25		*density*=0.30	
	MAE	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*
UMEAN	0.876	1.853	0.873	1.857	0.874	1.857	0.873	1.858	0.874	1.858	0.874	1.860
IMEAN	0.703	1.567	0.686	1.542	0.684	1.533	0.681	1.529	0.680	1.525	0.679	1.525
UPCC	0.634	1.377	0.553	1.311	0.511	1.258	0.483	1.220	0.467	1.189	0.454	1.170
IPCC	0.633	1.397	0.591	1.341	0.507	1.258	0.454	1.208	0.431	1.175	0.415	1.155
UIPCC	0.624	1.386	0.579	1.328	0.498	1.247	0.448	1.197	0.425	1.165	0.410	1.145
RegionKNN	0.594	1.641	0.577	1.637	0.569	1.627	0.569	1.617	0.562	1.619	0.563	1.618
LACF	0.682	1.500	0.650	1.468	0.610	1.416	0.582	1.381	0.562	1.357	0.546	1.332
PMF	0.568	1.537	0.487	1.321	0.451	1.221	0.430	1.171	0.416	1.139	0.409	1.125
NCF	0.440	1.325	0.385	1.283	0.372	1.253	0.362	1.205	0.349	1.138	0.337	1.123
LDCF	**0.402**	**1.277**	**0.367**	**1.233**	**0.345**	**1.169**	**0.331**	**1.138**	**0.331**	**1.110**	**0.312**	**1.107**

TABLE II
EXPERIMENTAL RESULTS OF THROUGHPUT

Methods	*density*=0.05		*density*=0.10		*density*=0.15		*density*=0.20		*density*=0.25		*density*=0.30	
	MAE	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*
UMEAN	54.333	110.296	53.947	110.345	53.971	110.201	53.906	110.190	53.862	110.194	53.841	110.261
IMEAN	27.342	65.844	26.962	64.843	26.757	64.266	26.669	64.069	26.595	63.873	26.558	63.715
UPCC	27.559	60.757	22.687	54.598	20.525	50.906	19.243	48.834	18.253	47.135	17.358	45.681
IPCC	27.102	62.665	26.270	60.479	25.487	57.561	23.726	54.564	22.286	52.293	21.301	50.602
UIPCC	27.070	60.510	22.440	54.506	20.256	50.585	18.888	48.238	17.863	46.392	16.965	44.832
RegionKNN	26.857	69.614	25.352	68.015	24.947	67.365	24.687	66.923	24.746	66.831	24.572	66.460
LACF	27.419	65.770	24.847	62.057	22.943	58.816	21.562	56.507	20.587	54.785	19.793	53.319
PMF	18.943	57.020	16.004	47.933	14.668	43.642	13.988	41.652	13.398	40.025	13.143	39.170
NCF	15.468	49.703	13.616	46.034	12.284	42.317	11.833	41.263	11.312	39.534	10.924	38.733
LDCF	**13.844**	**47.359**	**12.382**	**43.482**	**11.270**	**39.813**	**10.841**	**38.998**	**10.715**	**38.130**	**10.227**	**36.612**

code of the user, and the *UCN* represents the national code of the user. *SID*, *SASN* and *SCN* are similar to the above.

In the real world, the user-service matrix is usually very sparse, and users only invoke a very small number of services. In order to make the experiment more realistic, we randomly delete entries from the user-service matrix to make the matrix sparse at six different densities. For example, a matrix density (i.e., the ratio of non-zero entries) of 0.30 means that we randomly select 30% of the QoS entries as the training set for the model, and the remaining 70% are used as the test set to evaluate the accuracy of the model predictions. The matrix density is in steps of 0.05 and ranges from 0.05 to 0.30.

C. Parameter Setting

For methods of CF (e.g. UPCC, IPCC, LACF, RegionKNN, etc.), Top-K is set to 10, the learning rate is initialized to 0.001, the number of implicit feature factors (dimensions) is set to 10, the maximum number of iterations is set to 300, the regularization parameters are set to 0.1 and the random factor for rarefy matrix is set to 7.

For methods based on deep learning (e.g. NCF, LDCF[4]), we implement them on Keras (TensorFlow as the backend), where we use Gaussian distribution (*avg*=0, *stdev*=0.01) to initialize model parameters, use formula (15) to update parameter of the model. And we set the *batchsize* to 256, the learning rate to 0.0001, the number of MLP to 4, and use Adam for optimizing.

D. Evaluation Metrics

Two basic statistical accuracy metrics: Mean Absolute Error (*MAE*) and Root Mean Squared Error (*RMSE*) metrics are used to measure the recommendation performance of the selected methods. *MAE* and *RMSE* can be defined as:

$$MAE = \frac{\sum_{u,s} \left| Q_{u,s} - \hat{Q}_{u,s} \right|}{N} \tag{18}$$

$$RMSE = \frac{1}{N} \sum_{u,s} \left(Q_{u,s} - \hat{Q}_{u,s} \right)^2 \tag{19}$$

[4] https://github.com/ChunhuiYin/Location-aware_Deep_Collaborative_Filtering

where $Q_{u,s}$ is the QoS original value of the user u invoking the service s, $\hat{Q}_{u,s}$ the predictive QoS value of the user u invoking the service s, and N the total number of QoS.

The *MAE* calculates the average difference based on the number of values that need to be predicted, and measures all absolute differences between the labels and predicted values. *RMSE* gives a relatively high weight to outliers, and is sensitive to larger or smaller values within a set of predictions.

E. Comparison Methods

To verify the performance of the LDCF model in this paper, we compared nine of the most typical methods, including the traditional collaborative filtering methods and the latest deep learning methods:

1)**UMEAN** (User Mean): This method employs a user's average QoS value on the used web services for web service recommendation.

2)**IMEAN** (Item Mean): This method employs the average QoS value of the web service observed by other service users for service recommendation.

3)**UPCC** (User-based CF method using PCC) [1]: This method employs similar user behavior information for service recommendation.

4)**IPCC** (Item-Based CF Method using PCC) [2]: This method employs similar item attribute information for service recommendation.

5)**UIPCC** (User-based and Item-based CF) [4]: This method combines the similar users and similar web services adopted in UPCC and IPCC for service recommendation.

6)**RegionKNN** [40]: This method incorporates region factors into CF method for service recommendation, which is benchmark location-aware method.

7)**LACF** (Location-aware collaborative filtering) [19]: This method uses both locations of users and services for service recommendation, which is state-of-art location-aware method.

8)**PMF** (Probability Matrix Factorization) [42]: This method incorporates probabilistic factors into MF for service recommendation.

9)**NCF** (Neural collaborative filtering) [5]: This is an advanced neural network method that combines MLP and MF for recommendation.

Among the above, 1)-5) are traditional benchmark methods, 6)-7) are geolocation-based methods, and 8)-9) are model-based and deep learning-based methods, respectively.

F. Experimental Results and Analysis

We measured two evaluation metrics: response time and throughput at six different matrix densities. Each experiment was run 20 times and the average was taken as the final result. The experimental results are shown in Table I and Table II. As can be seen from Table I and Table II, our LDCF model has the smallest *MAE* and *RMSE* at any density, i.e., our LDCF is significantly better than the other existing methods. Especially when the sparsity is very large, our LDCF model has obvious advantages in recommending services.

Furthermore, the performance of NCF and LDCF based on deep learning is significantly better than that of Neighbor-based CF methods when the sparsity is large, indicating that deep learning is effective for high-dimensional and non-linear feature learning of relationship.

*1) Performance Comparison (**RQ1**)*

As shown in Fig. 4 (a)-(d), in addition to UMEAN, IMEAN, the performance of most of the chosen methods increases with the increase of density. Such a phenomenon is consistent with our intuitive experience that "the more data provided, the more accurate the CF similarity calculation, the more features the neural network can learn, and the higher the recommendation performance". Although the performance of the methods varies from different densities, the trends of these methods are similar, and the relationships under the same conditions are similar. To simplify the statement, we compare them at the same matrix density of 0.05. By observation, we can get the following relationships from the *MAE* performance comparison of RT and TP: LDCF < NCF < PMF < RegionKNN < UIPCC < IPCC < UIPCC < LACF < IMEAN < UMEAN. This indicates that the traditional CF method has poor accuracy of recommendation when the sparsity is large, but the deep learning-based method still performs well. Through analysis, we believe that traditional CF methods can only learn low-dimensional and linear features, hence their feature learning ability is greatly restricted, while deep learning can capture high-dimensional non-linear features, effectively compensate for data sparsity, and effectively make up for the limitation of CF feature learning by data sparsity.

In terms of *RMSE* performance, our LDCF model achieves the smallest *RMSE* at various densities, indicating that the LDCF model is very stable. Similarly, we can get the *RMSE* performance relationships of all the methods from Fig. 4: LDCF < NCF < PMF < UIPCC < UPCC < IPCC < LACF < IMEAN < RegionKNN < UMEAN. UPCC, IPCC, and UIPCC performed similarly. The performance of NCF was similar to PMF after the density became 2.5%. At a density of 0.2 and 0.25, NCF's response time was not as good as PMF's. Traditional CF methods can achieve good performance when the data is not sparse. NCF performs well when the data is sparse, but it only uses the identifier information. Fortunately, our method LDCF can not only learn the additional non-linear characteristics of users and services, but also incorporate location information to make up for the defects of CF method and NCF, so that at any density, the *RMSE* of our method LDCF is superior to that of other methods. Therefore, compared with the existing recommendation methods including traditional CF methods and the state-of-the-art deep learning methods, LDCF can effectively deal with data sparsity and achieve better recommendation performance.

*2) Impact of Location Information (**RQ2**)*

To examine the impact of location information, we compare the LDCF model with the NCF model, because LDCF incorporates locations, but NCF does not. In the experiment, both NCF and LDCF are trained under the same loss function and the same network parameters. To the best of our knowledge, the performance of neural network methods largely depends on the initialization of the machine, and the model gradually converges to the optimal situation as the number of iterations increases. To verify the impact of location information on deep collaborative filtering recommendations, we iterate 50 times with matrix density of 0.05, predicting RT and TP, respectively.

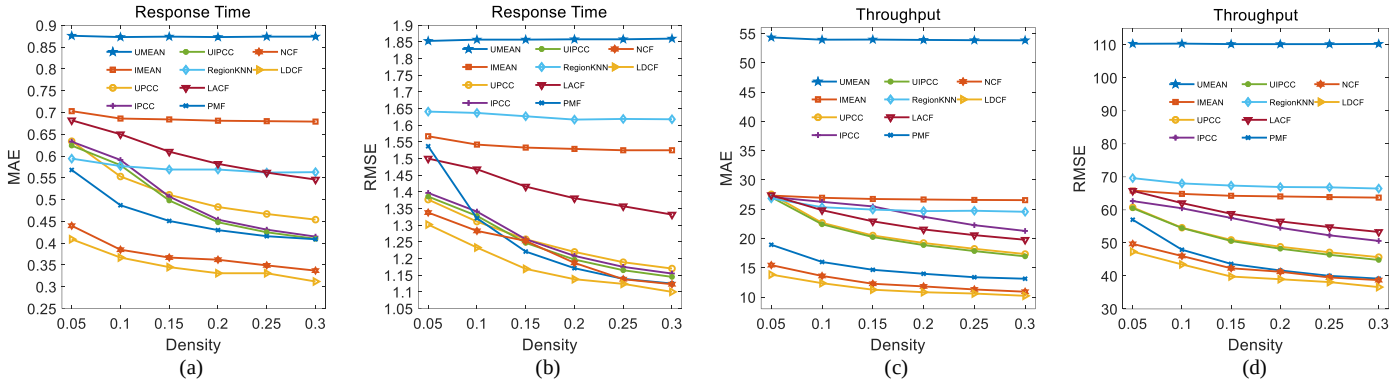


Fig. 4. Performance comparison (***RQ1***)

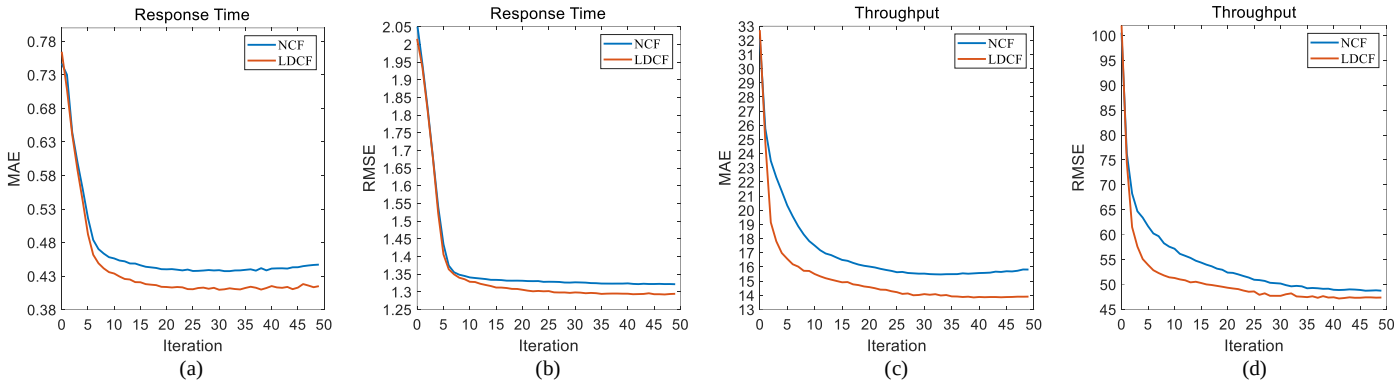


Fig. 5. Performance w.r.t. the number of iterations. (***RQ2***)

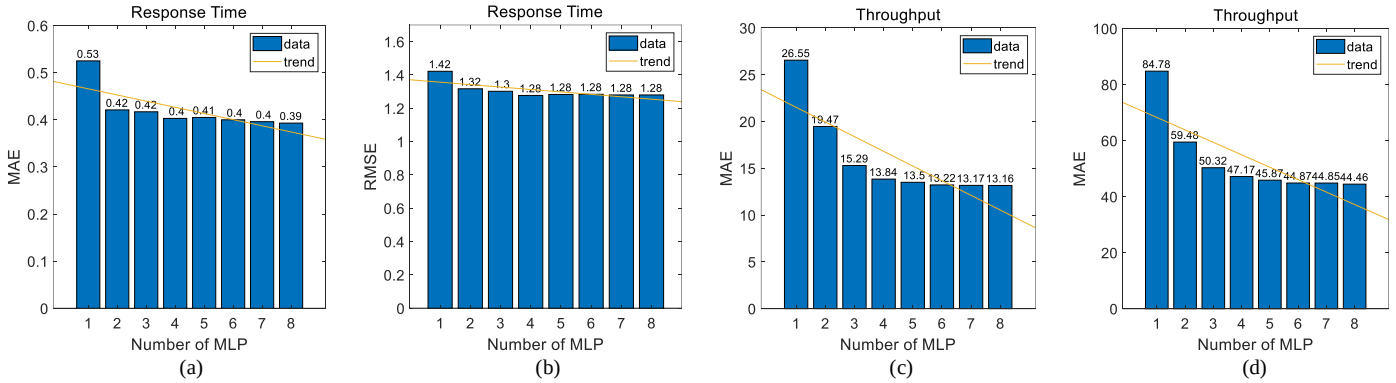


Fig. 6. Performance w.r.t. the number of MLP. (***RQ3***)

As shown in Fig. 5 (a)-(b), in the RT experiments, the two methods begin to converge after about 15 iterations and tend to be stable. LDCF is almost the same as NCF before convergence. As the number of iterations increases, the performance of the two methods is gradually improved. However, the performance of the LDCF model considering location information is better than the NCF model considering only the identifier information. Similarly, as shown in Fig. 5 (c)-(d), in the TP experiments, whether it is on *MAE* or *RMSE*, the performance of the LDCF model considering the location information is significantly better than the NCF model. Therefore, the introduction of location information of users and services can greatly contribute to the performance of the model.

*3) Impact of Depths (**RQ3**)*

It is well known that neurons are the smallest unit of arithmetic that constitutes a neural network framework, and their connection has an indispensable effect on neural networks. In this work, in order to allow the neural network to learn more non-linear features, we use the tower structure. Under the framework of the tower structure, we let the neural network with only 1 MLP have 8 neurons, and the network topology map formed by it can be expressed as <8>. As depth is equal to 2, the network topology map can be expressed as <16, 8>, which means that the first layer of MLP has 8 neurons and the second layer has 16 neurons. Similarly, we can state that, if the depth is *i*, the neural network topology map can be expressed as $[2^3*2^{i-1}, 2^3*2^{i-2}, \cdots, 2^3]$.

The experimental results are shown in Fig. 6 (a)-(d). It can be seen from the trend line that as the number of MLP changes from 1 to 8, the performance improves significantly at first, and then is gradually reduced, indicating that deep neural network can greatly improve the performance. However, as the non-linear features are limited, the performance improvement of more than 6 MLPs is not significant. We believe that the MLP can learn the high-dimensional and non-linear complex

feature, thus making up for limitations of CF for feature learning. Although the addition of more than 6 layers of MLP can only slightly improve the performance of the model, the performance of deep neural networks should not be underestimated.

*4) Impact of AC (**RQ4**)*

TABLE III
IMPACT OF AC (***RQ4***)

Methods	Response Time		Throughput	
	MAE	***RMSE***	***MAE***	***RMSE***
AC-MLP	0.419	1.322	18.353	59.735
AC-EUC	0.411	1.321	17.539	57.721
AC-COS	0.402	1.277	13.844	47.359

In Section III, we propose a similarity Adaptive Corrector at the Input Layer. Its main function is to calculate the similarity between the user's location and the location of the service, and then we transmit it to the Output Layer to correct the prediction results of the model.

Before verifying the effect of the AC on the prediction results under the Location-aware collaborative filtering framework, we define the following three sub-methods:

a) **AC-MLP**: This method uses only a multi-layer perceptron and does not transmit the similarity between the user's location and the location of the service to the output layer for correction.

b) **AC-EUC**: This method transmits the Euclidean similarity between the user's location and the location of the service to the output layer based on the AC-MLP.

c) **AC-COS**: This method transmits the cosine similarity between the user's location and the location of the service to the output layer based on the AC-MLP.

In order to study the influence of AC in depth, we conducted experiments on response time and throughput by setting the matrix density = 0.05. As shown in Table III, AC-MLP performance without the similarity is poor regardless of response time or throughput. After we added the similarity of the Adaptive Corrector (AC), performance improved significantly. For example, in terms of throughput, the *MAE* of AC-EUC is 17.539 and its *RMSE* is 57.721. As a comparison, the *MAE* of AC-MLP is 18.353 and its *RMSE* is 59.735. This phenomenon shows that AC is very helpful for improving the performance of the model.

Furthermore, in the comparison between AC-COS and AC-EUC, we can find that the performance of AC-COS is much better than that of AC-EUC. That is, the cosine similarity is more suitable for our model than the Euclidean one. Through the comparison analysis, we believe that Euclidean similarity can reflect the absolute difference of individual numerical features by measuring the linear distance between two spatial points, while cosine similarity can reflect the difference of angles between two vectors by measuring the directional distance between two spatial points. In our LDCF model, location information is represented as a vector, hence the cosine similarity is more suitable.

Moreover, our proposed similarity Adaptive Corrector can incorporate various CF similarity calculation methods into the deep learning framework to improve the performance of the model, as long as the similarity calculation has a tensor form of expression. It is enough to prove that our proposed similarity Adaptive Corrector (AC) has very strong adaptability and extensity.

*5) Impact of Huber Loss (**RQ5**)*

TABLE IV
IMPACT OF HUBER LOSS (***RQ5***)

Loss	Response Time		Throughput	
	MAE	***RMSE***	***MAE***	***RMSE***
L1	0.3717	1.337	17.128	56.399
L2	0.5224	1.3074	26.462	60.784
Huber	0.402	1.277	13.844	47.359

In this work, the Huber loss function defined as Equation (15) is first introduced. We have conducted experiments to compare the performance of the Huber loss function against the L1 loss (i.e., absolute error loss) and L2 loss (i.e., squared error loss) functions with 0.05 matrix density. The experimental results are summarized in Table IV.

In the comparison between the L1 loss and L2 loss functions, we can easily see that the L1 loss function wins the match on MAE but lost that on RMSE. The results show that the Huber loss function well balances between MAE and RMSE for the response time, and outperforms the L1 loss and the L2 loss function significantly in terms of both MAE and RMSE for the throughput.

6) Threats to Validity

In this subsection, we discuss the related threats to the validity of our evaluation of LDCF, including the construct validity, external validity, internal validity and conclusion validity.

Threats to construct validity. One of the main threats to the construct validity of our evaluation lies in the comparison of recommendation accuracy with the selected recommendation methods. The selected recommendation methods are based on CF and deep learning techniques, which are currently the most popular and widely used. Although other methods, such as the trust-based recommendation method [16], are not included in the evaluation, this threat is not particularly significant because our LDCF can be indirectly compared with those methods through inspecting the evaluation presented in the related literature using the methods included in the evaluation as reference methods. Another major threat to the construct validity of our experiments is the lack of consideration of time [18] and trust [16] during the recommendation. This threat is also not important, because although these aspects enhances the recommendation performance, they do not change the basic mechanism that improves the accuracy by adding contextual information. Although time can be included in LDCF in a manner similar to [43], a direct comparison between LDCF and other methods in the location-aware recommendation methods is more straightforward and representative.

Threats to external validity. The main threat to the external validity of our evaluation is the nature of the dataset being used. It may not be able to exactly represent all the real-world applications. To minimize this threat, we used the dataset that has been widely employed in experiments on recommendation methods for Web services, i.e., the WS-Dream dataset. In the

experiments, we generated matrices with different densities by randomly removing entries to simulate various recommendation scenarios in real word. In this way, we can compare LDCF with existing methods comprehensively and draw a conclusion that LDCF can alleviate the data sparsity problem properly. This also greatly reduces the threat to the external validity of our evaluation.

***Threats to internal validity*.** The main threat to the internal validity of our evaluation is its representativeness. Although the experiments under other parameter settings (e.g., more selections of loss functions and optimizers) can be conducted, we believe that this threat is not significant because we have examined the effects of three most popular loss functions (i.e., L1 loss, L2 loss and Huber loss) and the effects of widely used optimizer Adam. Thus, our evaluation is representative and the threat to the internal validity is not significant.

***Threats to conclusion validity*.** The main threat to the conclusion validity of our evaluation is the lack of statistical tests, e.g., chi-square tests. In fact, we could have conducted chi-square tests to draw conclusions while evaluating LDCF. Since we have performed 20 experiments in each run and averaged the optimal run results after the model converges, this will lead to a large number of test cases, which results in a small p-value in the chi-square tests and lowers the practical significance of the test results [44]. However, this number of runs is not even close to the number of observation samples concerned by Lin *et al.* [44]. Thus, the threat to the conclusion validity due to the lack of statistical tests may be possible but not significant.

VI. Conclusion

This paper proposes a new deep learning based model, i.e., the Location-aware Deep Collaborative Filtering (LDCF) model, which aims to solve the key problem of service recommendation—predicting the Quality of Service. The proposed model can learn the high-dimensional and non-linear relationships between users and services through Multi-Layer Perceptron, and incorporate the similarity Adaptive Corrector to correct the predictive values. Substantial experiments have been done on real-world Web service datasets to evaluate the performance of LDCF. Compared with traditional collaborative filtering methods and the latest deep learning methods, our approach can greatly improve the performance of Web service recommendation. In the future, we will further consider the role and impact of other contextual information (e.g., time, trust) on service recommendation.

References

[1] J. S. Breese, D. Heckerman, and C. M. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncer. Artifi. Intell.*, Madison, WI, 1998, pp. 43–52.

[2] B. Sarwar, G. Karypic, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. Int. World Wide Web Conf.*, Hong Kong, CHN, 2001, pp. 285–295.

[3] Z. Zheng, H. Ma, M. R. Lyu and I. King, "WSRec: a collaborative filtering based web service recommender system," in *Proc. Int. Conf. Web Serv.*, Los Angeles, USA, 2009, pp. 437-444.

[4] Z. Zheng, H. Ma, M.R.Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Trans. Serv. Comput.*, vol. 4, no. 2, pp. 140–152, Apr.–Jun. 2011.

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu and T.S Chua, "Neural collaborative filtering," in *Proc. Int. World Wide Web Conf.*, Perth, AU, 2017, pp. 173-182.

[6] H. Xue, X. Dai, J. Zhang and S. Huang, "Deep matrix factorization models for recommender systems," in *Proc. 6th Int. Joi. Conf. Artifi. Intell.*, Melbourne, AU, 2017, pp. 3203-3209.

[7] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS prediction for web services via collaborative filtering," in *Proc. Int. Conf. Web Serv.*, Salt Lake City, UT, 2007, pp. 439–446

[8] Z. Tan, and L. He, "An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance Principle," *IEEE Access*, vol. 5, pp. 27211-27228, Nov. 2017.

[9] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proc. ACM SIGIR Conf.*, Amsterdam, NL, 2007, pp. 39-46.

[10] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. C. Zhou and Z. Wu, "Predicting quality of service for selection by neighborhood-based collaborative filtering," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 43, no. 2, pp. 428-439, 2013.

[11] Y. Zhang, K. Wang, Q. He, F. Chen, S. Deng, Z. Zheng and Y. Yang," Covering-based Web Service Quality Prediction via Neighborhood-aware Matrix Factorization", *IEEE Trans. Serv. Comput.*, to be published.

[12] P. Wang, Z. Ding, C. Jiang and M. C. Zhou, "Constraint-aware approach to web service composition," *IEEE Trans. Syst., Man, Cybern. A, Syst.*, vol. 44, no. 6, pp. 770-784, 2014.

[13] Q. Wu, F. Ishikawa, Q. Zhu and D. H. Shin, "QoS-aware multi-granularity service composition: modeling and optimization," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 46, no. 11, pp. 1565-1577, 2016.

[14] Z. J. Ding, J. J. Liu, Y. Q. Sun C. Jiang and M. C. Zhou, "A transaction and QoS-aware service selection approach based on genetic algorithm," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 45, no. 7, pp. 1035-1046, 2017.

[15] X. Wu, B. Cheng, and J. Chen, "Collaborative filtering service recommendation based on a novel similarity computation method," *IEEE Trans. Serv. Comput.*, vol. 10, no. 3, pp. 352-365, 2017.

[16] K. Chen, H. Mao, X. Shi, Y. Xu and A. Liu, "Trust-aware and location-based collaborative filtering for web service QoS prediction," in *Proc. IEEE 41st Ann. Comput. Soft. Appli. Conf.*, Turin, IT, 2017, pp. 143-148.

[17] Z. Zheng, and M. R. Lyu, "Personalized reliability prediction of web services," *ACM Trans. Soft. Engi. Meth.*, vol. 22, no. 2, pp. 1-25, 2013.

[18] Y. Hu, Q. Peng, X. Hu and R. Yang, "Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering," *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 782-794, 2015.

[19] M. Tang, Y. Jiang, J. Liu and X. Liu, "Location-aware collaborative filtering for QoS-based service recommendation." in *Proc. Int. Conf. Web Serv.*, Salt Lake City, UT, Jun, 2012, pp. 202-209.

[20] J. Liu, M. Tang, Z. Zheng, X. Liu and S. Lyu, "Location-aware and personalized collaborative filtering for web service recommendation," *IEEE Trans. Serv. Comput.*, vol. 9, no. 5, pp. 686-699, 2016.

[21] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang and T. Zhang, "Collaborative Web Service Quality Prediction via Exploiting Matrix Factorization and Network Map", *IEEE Trans. Net. Ser. Manage., vol.* 13, *no.* 1, pp.126-137.

[22] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: a time-aware personalized QoS prediction framework for web services," in *Proc. IEEE Int. Sym. Soft. Rel. Engi.*, Hiroshima, JPN, 2011, pp. 210-219.

[23] Y. Yang, Z. Zheng, X., M. Tang, Y. Lu and X. Liao, "A location-based factorization machine model for web service QoS prediction," *IEEE Trans. Serv. Comput.*, to be published.

[24] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T. S. Chua, "Outer product-based neural collaborative filtering," in *Proc. 6th Int. Joi. Conf. Arti. Intell.*, Stock., Jul 13-19, 2018, pp. 2227-2233.

[25] D. Kim, C. Park, J., S. Lee and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. 10th ACM Conf. Rec. Sys.*, Boston, MA, USA, Sep. 7-8, 2016, pp. 233-240.

[26] L. Wu, P. Sun, R. Hong, Y. Ge and M. Wang, "Collaborative neural social recommendation," *IEEE Trans. Syst., Man, Cybern. A, Syst.*, to be published.

[27] R. Xiong, J. Wang, Z. Li, B. Li and P. C. K. Hung, "Personalized LSTM based matrix factorization for online QoS prediction," in *Proc. Int. Conf. Web Serv.*, Seattle, USA, Jun. 25-30, 2018, pp. 34-41.

[28] R. Xiong, J. Wang, N. Zhang and Y. Ma, "Deep hybrid collaborative filtering for Web service recommendation," *Exp. Sys. with Applica.*, vol. 110, pp. 191-205, 2018.
[29] B. Bai, Y. Fan, W. Tan and J. Zhang, "DLTSR: A deep learning framework for recommendation of long-tail web services". *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp.1-1, 2017.
[30] W. Yuan, C. Li, D. Guan, G. Han and A. M. Khattak, "Socialized healthcare service recommendation using deep learning," *Neu. Comput. Appli.*, vol. 30, no. 7, pp. 2071-2082, 2018.
[31] S. Zhang, L. Yao, A. Sun and Y. Tay, "Deep Learning based Recommender System: A Survey and New Perspectives", *ACM Comput. Surveys*, to be published.
[32] L. Zhang, T. Luo, F. Zhang and Y. Wu, "A recommendation model based on deep neural network," *IEEE Access*, vol. 6, pp. 9454-9463, 2018.
[33] M. Ailem, B. Zhang, A. Bellet, P. Denis and F. Sha. "A Probabilistic Model for Joint Learning of Word Embeddings from Texts and Images". In *Conf. Empi. Meth. Natu. Lan. Proce.*, Brussels, Belgium, Oct.-Nov., 2018, pp. 1478–1487.
[34] W. Sun, J. Cao, and X. Wan, "Semantic Dependency Parsing via Book Embedding," in *Associa., Comput. Lingu.*, Los Vancouver, Canada, Jul 30 - Aug 4, 2017, pp. 8285-838.
[35] R. Wang, A. M. Finch, M. Utiyama and E. Sumita, "Sentence Embedding for Neural Machine Translation Domain Adaptation," in *Associa., Comput. Lingu.*, Los Vancouver, Canada, Jul 30 - Aug 4, 2017, pp. 560-566.
[36] S. Rendle, C. Freudenthaler, Z. Gantner and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," *in Proc. 25th Conf. Uncer. Artif. Intell.*, Montreal, CAN, 2009, pp. 452-461.
[37] Huber, J. Peter, "Robust estimation of a location parameter", *The Ann. Math. Stati.*, vol. 35, no. 1, pp.73-101, 1964.
[38] D. P. Kingma, and J. Ba, "Adam: a method for stochastic optimization," in *Proc. the 3rd Int. Conf. Learn. Represent.*, San Diego, USA, 2014 pp.1-15.
[39] J. Duchi, H. Elad and Y. Singer, "Adaptive sub-gradient methods for online learning and stochastic optimization," *Jour. Mach. Lear. Res.*, vol. 12, Jul 12, pp.2121-2159, 2011.
[40] T. Tieleman and G. Hinton, "RMSProp: divide the gradient by a running average of its recent magnitude," Tor. University, Toronto, CAN, *Neu. Net. Mach., Lea., Tech. Rep*. 2012.
[41] X. Chen, X. Liu, Z. Huang and H. Sun, "RegionKNN: A Scalable Hybrid collaborative filtering Algorithm for Personalized Web Service Recommendation." in *Proc. Int. Conf. Web Serv.*, Florida, USA, Jul.5-10, 2010, pp. 9-16.
[42] R. Salakhutdinov, and A. Mnih, "Probabilistic Matrix Factorization," in *Proc. In Adv. neu .info. pro. sys*, Vancouver, CAN, 2007, pp. 1257-1264.
[43] D. Dong, X. Zheng, R. Zhang and Y. Wang, "Recurrent collaborative filtering for unifying general and sequential recommender," in *Proc. Int. Joi. Conf. Artifi. Intell.*, Stockholm, SWE, Jul.9-19, 2018, pp. 3350-3356.
[44] M. Lin, H. C. Lucas, and G. Shmueli, "Too Big to Fail: large samples and the p-value problem", *Info. Sys. Res.*, vol. 24, no. 4, pp.906-917, 2013.

Yiwen Zhang received his PhD degree in management science and engineering in 2013 from Hefei University of Technology. He is a professor in the School of Computer Science and Technology at Anhui University. His research interests include service computing, cloud computing and big data. More details about his research can be found at https://bigdata.ahu.edu.cn.

Chunhui Yin received his bachelor degree in computer science and technology in 2017 and now is a master student in the School of Computer Science and Technology at Anhui University. His current research interests include deep learning, recommenders and service computing.

Qilin Wu received his PhD degree in computer application technology in 2011 from Hefei University of Technology. He is a professor in the School of Information Engineering at Chaohu University. His research interests include resource allocation and optimization for wireless networks, edge computing and service computing.

Qiang He received his first PhD degree from Swinburne University of Technology, Australia, in 2009 and his second PhD degree in computer science and engineering from Huazhong University of Science and Technology, China. He is a senior lecturer at Swinburne. His research interests include service computing, software engineering, cloud computing and edge computing. More details about his research can be found at https://sites.google.com/site/heqiang/.

Haibin Zhu is a professor with the School of received the B.S. degree in computer engineering from the Institute of Engineering and Technology, Zhengzhou, China, in 1983, and the M.S. and Ph.D. degrees in computer science from the National University of Defense Technology (NUDT), Changsha, China, in 1988 and 1997, respectively.

He is a Full Professor with the Department of Computer Science and Mathematics and the Founder and the Director of Collaborative Systems Laboratory, Nipissing University, North Bay, ON, Canada, and a Visiting Professor with the Department of Control Science and Engineering, Nanjing University, Nanjing, China. He is serving and served as the Co-Chair of the Technical Committee of Distributed Intelligent Systems of the IEEE SMC Society, an Associate Editor for the IEEE Systems, Man, and Cybernetics Magazine and the International Journal of Agent Technologies and Systems, an Associate Editor-in-Chief for the International Journal of Advances in Information and Service Sciences, an Editorial Board Member for the International Journal of Software Science and Computational Intelligence, a Guest Editor for the IEEE Transactions on Systems, Man, and Cybernetics: Systems.