


RESEARCH

Open Access



# Mobility-aware personalized service recommendation in mobile edge computing

Hongxia Zhang<sup>\*</sup> , Yanhui Dong and Yongjin Yang

<sup>\*</sup>Correspondence:  
zhanghx@upc.edu.cn  
Department of Computer  
Science and Technology,  
China University  
of Petroleum, Qingdao, China

## Abstract

With the proliferation of smartphones and an increasing number of services provisioned by clouds, mobile edge computing (MEC) is emerging as a complementary technology of cloud computing. It could provide cloud resources and services by local mobile edge servers, which are normally nearby users. However, a significant challenge is aroused in MEC because of the mobility of users. User trajectory prediction technologies could be used to cope with this issue, which has already played important roles in service recommendation systems with MEC. Unfortunately, little attention and work have been given in service recommendation systems considering users mobility. Thus, in this paper, we propose a mobility-aware personalized service recommendation (MPSR) approach based on user trajectory and quality of service (QoS) predictions. In the proposed method, users trajectory is firstly discovered by a hybrid long-short memory network. Then, given users trajectories, service QoS is predicted, considering the similarity of different users and different edge servers. Finally, services are recommended by a center trajectory strategy through MPSR. Experimental results on a real dataset show that our proposed approach can outperform the traditional recommendation approaches in terms of accuracy in mobile edge computing.

**Keywords:** Mobile edge computing, Mobility, Edge service, Service recommendation, Quality of service(QoS)

## 1 Introduction

In mobile edge computing (MEC), a number of edge servers with computation and storage capabilities are deployed to construct a network, termed as the mobile edge network (MEN) [1, 2]. Services can be provided within close proximity of service subscribers for satisfying the high-workload and low-latency requirements through MEN. Application developers and content providers are able to deploy services on edge servers according to context information and mobile edge network information [3]. However, due to the increasingly growing popularity of mobile devices, a large number of mobile services have been developed for mobile devices. For high service qualities and efficiency, users should invoke their desired services through accessing edge servers rather than core network or the Internet [4, 5]. It is critical to recommend proper mobile services to users, taking both quality of services (QoS) and network optimizations into consideration.

Undoubtedly, user mobility has a detrimental impact on the QoS-based service recommendation [6]. On one hand, a user requests a cloud service by his/her mobile terminal device continuously, the service will be provided by different edge servers during physical location changes. On the other hand, QoS data of one edge server may prominently volatile due to user mobility, as the service has to switch among different edge servers frequently, resulting in higher network overhead. Although some approaches [7, 8] on service recommendation have already been proposed in MEC, these approaches directly overlook user mobility, or merely consider user mobility with unrealistic models, yielding inappropriate service recommendations. User mobility brings two main problems, which are not fully taken into account as follows,

- *Frequent handoff among edge servers* The handoff among different edge servers occur frequently caused by user mobility [9]. In such a case, a user request the desired service, which will be provided by different edge server at different moments, because of his dynamical location changes [10]. Each edge server is only able to provide the service for a moment due to the high QoS performance in a small area. The network load and server load can be increased by data transmission among servers. Besides, the QoS may be changed or even absent on servers. Therefore, user mobility leads to frequent handoff among edge servers and significantly degrade service performance [11].
- *Volatility of QoS data* In the light of extensive network changes, the QoS data are unstable. One active user invokes the service many times, and QoS data is different each time. Different users who invoke the same service are provided by different QoS data at the same time. This phenomenon is common in real life. Especially in MEC, the QoS data is different each time, because of different performance, surroundings and the number of users using services at that time. So the QoS data is highly volatile, it is more complicated when recommending better services for users.

To address these questions, a mobility-aware personalized service recommendation approach is proposed in this paper, it fully considers user mobility when recommending services, makes personalized service recommendations according to different users' location change, to improve the experience of users and reduce network load. Specifically, user trajectories are predicted through mobility patterns leveraging a hybrid long short-term memory (LSTM) network firstly, which helps to obtain candidate edge servers candidate servers. Next, we predict QoS data on candidate servers based on the trajectory by reducing the influence of the above two factors. Then, QoS predictions are carried out based on a location-based collaborative filtering approach, the service with the best QoS data will be recommended to the user. The major contributions of this paper are summarized as follows:

- A hybrid LSTM-based mobility prediction model (LSTMM) is proposed to predict user trajectories from long-range and sparse position data, in which server information as well as user characteristics and distances information are considered to improve the prediction efficiency.

- We put forward a mobility-aware personalized service recommendation approach, focusing on predicting the QoS data of servers along trajectories. Information of users similar to a specific user is applied to reduce the volatility of QoS data, which will improve prediction accuracy. Servers around predicted places on the trajectory will provide global optimal QoS values and reduce the network overload.

The rest of this paper is organized as follows: Sect. 2 summarizes the related work. Section 3 introduces a motivation scenario. Section 4 presents our mobility-aware service recommendation approach based on trajectory prediction and edge server similarity. Section 5 describes the implementation of our experiments and performance comparisons. Section 6 draws a conclusion.

## 2 Related work

Many literature over the past decade focused on developing service recommendation systems. Most of these works focus on collaborative filtering based recommendation [7, 12, 13], content-based recommendation [14, 15] and model-based recommendation [16, 17].

Shao et al. [18] proposed a user-based collaborative filtering method for predicting the web service QoS value and conducted experiments on 20 web services. Zheng et al. developed a web service recommendation system called WSRec [19–21] for collecting a large-scale real-world web service QoS information through a user-collaborative mechanism. They proposed a hybrid collaborative filtering approach to predict web service QoS value by combining the user-based Pearson correlation coefficient (PCC) approach and the item-based PCC approach [20]. Sun et al. [12] presented a normal recovery collaborative filtering approach to solve the personalized web service recommendation problem. They proposed a new similarity measure for web service similarity computation and proposed a novel collaborative filtering approach. Better prediction accuracy is achieved through fusing the information of similar users and similar web services. Wang et al. [7] performed the prediction via a set of multi-dimensional QoS measures through exploiting the structural relationships among multi-dimensional QoS data. They proposed an integrated QoS prediction approach to recommend efficient Web service for mobile clients. These approaches made a lot of effort on improving the accuracy of service recommendation and achieved great success in the traditional Web service recommendation. Due to users' mobility and QoS volatility, these approaches will lead to large deviations and inappropriate results when recommending services in mobile edge computing.

To address these problems, many studies have been studied on the location-based recommendation [22–24]. Zheng et al. [25] presented a user-centered collaborative location and activity filtering approach to cluster many users data, applied the collaborative filtering to find like-minded users and like-patterned activities at different locations. They modeled the user location-activity relations with a tensor representation and proposed a regularized tensor and matrix decomposition solution to address the sparse data problem in mobile circumstances. This approach only considered the users locations, not service locations. Chen et al. [26] proposed a location-aware Web service recommend system, which employed both location information and QoS values to cluster

users and services. They made personalized QoS prediction for users on the clustering results according to user locations and QoS values. Location-Aware and personalized collaborative filtering approach [27] leveraged locations of both users and Web services when selecting similar neighbors for the target user or service. The method presented an enhanced similarity measurement for both users and Web services, by taking into account the personalized influence of them. Wang et al. [7] indicated that user mobility often makes service QoS prediction values deviate from actual values in mobile edge computing networks. They proposed a service recommendation approach based on collaborative filtering considering user mobility. They divided service invocation into two cases and calculated user or edge server similarity separately. But the frequency of mobility did not take into consideration, which may bring immigration among different edge servers frequently.

Although the location is vital to service recommendation at the current time, it cannot explain movement trends, and it is not sufficient to predict QoS data only considering the next location. Therefore, these approaches will bring inaccurate even bad results in MEC. To address these problems, we propose an approach that both consider the user trajectory and the data volatility to provide high quality services.

### 3 Scenario and problem formulation

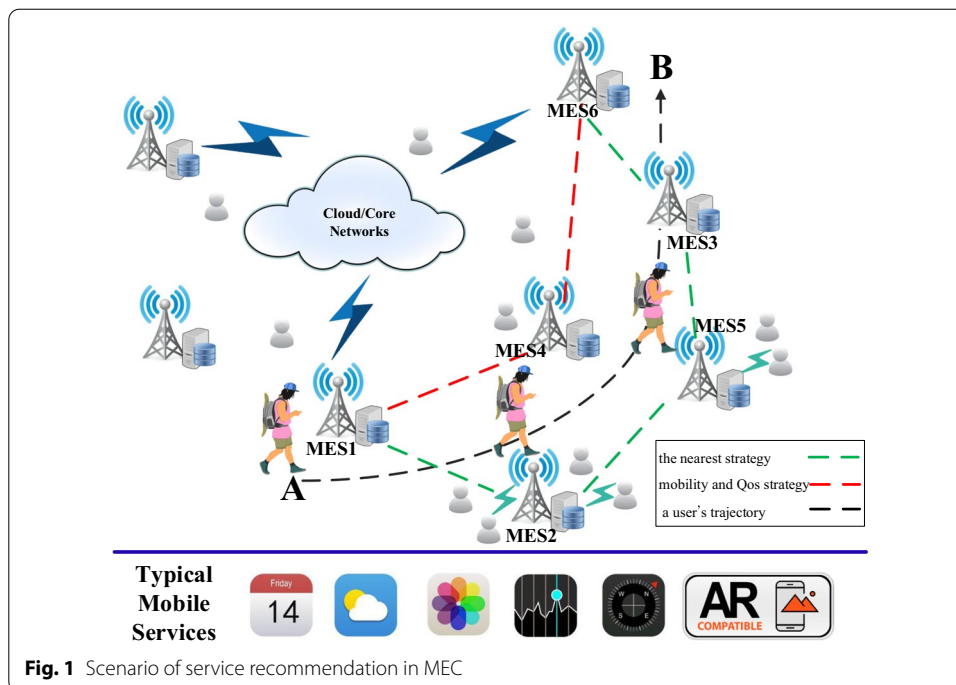
#### 3.1 Scenario

A mobile edge network consists of multiple mobile edge servers (*MESs*), which provide network access services for mobile users anytime in a certain range, such as airports, large shopping malls, museums, etc. Suppose that a mobile user named Jenny often uses services to obtain surrounding information in an airport, such as AR/VR, location, discount, etc.

Considering Jenny is invoking service  $s$  by accessing the nearby mobile edge server *MES1*. As Fig. 1 depicts, she uses service  $s$  continuously on her traveling from  $A$  to  $B$  along the trajectory as the black dotted line is shown. In this situation, how to predict the next accessing edge server and the corresponding QoS value of the service becomes an important issue. If service  $s$  is always provided by the nearest edge server, that is to say, the green-line option:  $MES1 \rightarrow MES2 \rightarrow MES5 \rightarrow MES3 \rightarrow \dots$  will provide the service successively. However, service  $s$  in *MES2* may have bad quality (e.g., the response time is 100ms) for accessing by many people at that time. After that, Jenny will access *MES5* with the *response time* is 95ms for the shortest distance. Meanwhile, the *response time* of service  $s$  is 95ms when accessing *MES4* during the entire moving process. Therefore, the red-line option:  $MES1 \rightarrow MES4 \rightarrow MES6 \rightarrow \dots$  will obtain the best experience globally, which has the same path direction as Jenny's trajectory. Therefore, the users experience can be improved and network delays can be reduced compared to the case where only the nearest edge server is recommended. In the above scenario, the trajectory defines user mobility more accurately than the current location.

#### 3.2 Problem formulation

In this section, we first define the mobility prediction model, then the mobility-aware personalized service recommendation problem is formalized. Trajectory data are the foundation of the mobility prediction, which characterize locations and times of



**Fig. 1** Scenario of service recommendation in MEC

moving objects. A trajectory is a chronologically ordered sequence of spatio points. A spatio point  $q_i$  is defined as a tuple of its coordinate  $\langle x_i, y_i \rangle$ . A sequence  $sq$  is a set of spatio point  $q_i$ , i.e.  $sq=q_1q_2\dots q_m(n \in D)$ . A subsequence of  $sq$  is defined as  $sub(sq_r)_l=q_{r-l}\dots q_{r-1}q_r$ , where  $r-l$  is the start position on the subsequence, and  $l$  is the number of spatiotemporal points.

Formally, given a sequence set  $SQ$  and an active user  $u$ , the trajectory set of user  $u$  is a chronologically ordered subsequence.  $Tr(u)=sub(sq_r)_l$  contains all the trajectories generated by user  $u$  with the sequence length  $l$ . A trajectory of user  $u$  which is recorded at the  $t$  time window can be defined as  $Tr(u)_t=sub(sq_t)_l=q_{t-l}\dots q_{t-1}q_t \subseteq Tr(u)$ .

If the current time window is  $t$ , then the trajectory history can be denoted as  $Tr(u)_1, Tr(u)_2, \dots, Tr(u)_{t-1}$ . For the convenience of calculation, we discretize continuous time interval  $t$  to hour of day and day of week.

The mobility prediction problem can be formalized as:

**Problem 1** Given a user's history trajectory set  $Tr(u)$  and the current trajectory  $Tr(u)_t$ , the mobility prediction is to predict the user's trajectory  $\widehat{Tr}(u)_{t+1}$  at the next time interval, the prediction length  $l=len(Tr(u)_{t+1})$ , that is, construct the predicting model  $f(Tr(u), Tr(u)_t) \mapsto \widehat{Tr}(u)_{t+1}$ .

Analysis results [28] show that people's trajectory with high probability, which confirms the existence of mobility patterns for users. If the mobility pattern can be obtained through the history trajectory, then the Problem 1 will be solved according to the related mobility pattern information.

**Definition (Mobility pattern)** Given a trajectory set  $Tr$  of users  $U$ , the user mobility pattern  $mp$  is a set of typical trajectories that can indicate the characteristics of users movement and its uniqueness.

Apparently, QoS properties, including response time, throughput, failure-probability, etc., are heavily dependent on network environments and users location [26]. As a result, in service recommendations, the QoS properties of MEC servers along trajectories should be considered rather than only selecting the nearest ones. Therefore, our motivating problem is to make more accurate QoS estimations based on the trajectory for personalized service recommendations in MEC to improve user experience and reduce the overload of networks. Therefore, the user mobility-aware personalized service recommendation problem in MEC can be formulated as follows:

**Problem 2** Let  $U$  be a set of users,  $S$  be a set of services,  $MES$  be a set of MEC servers, an active user  $u \in U$  is using a service  $s \in S$  providing by  $mes \in MES$  when roaming at  $m$ , then  $u$  leaves the service area of  $mes$  or the QoS of  $s$  gets worse gradually, the  $mes_i \in MES$  around the predicted trajectory  $\widehat{Tr}(u)_{t+1}$  will be recommended to provide service  $s$  for user  $u$  for better QoS. The problem is predicting the QoS data of service  $s$  on  $MES$  along the predicted trajectory  $\widehat{Tr}(u)_{t+1}$ , then the server  $mes_j \in MES$  with the best QoS data will recommend to  $u$ .

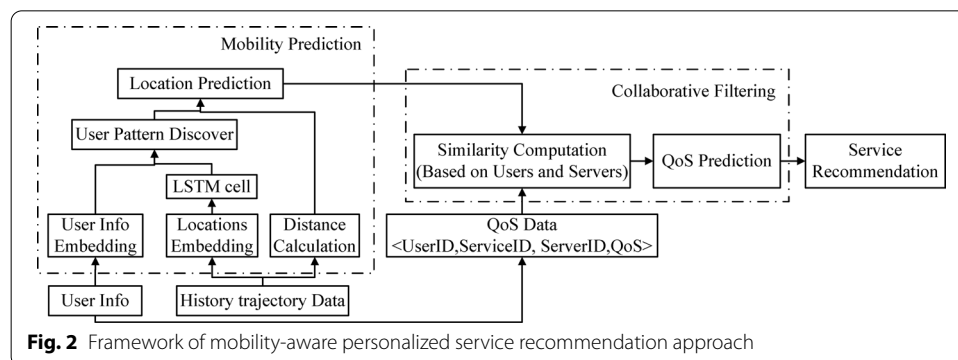
To reach this goal, several challenges must be addressed, including: (1) How to discover the user mobility pattern and predict the trajectory. (2) How to redefine the collaborative filtering algorithm to predict QoS data and perform service recommendation when considering edge servers information.

#### 4 Mobility-aware personalized service recommendation in MEC

In this section, we propose a two-stage mobility-aware personalized service recommendation approach. Firstly, a LSTM-based mobility prediction model(LSTMM) is presented to predict the user trajectory by combining the user information, historical trajectory information and distances between servers. Then a personalized server recommendation approach is presented to predict the QoS data on the mobile edge servers around the trajectory by weakening the volatility of QoS data.

As shown in Fig. 2, our proposed approach consists of three steps as follows:

1. Trajectory prediction. In this stage, user mobility patterns can be discovered by the historical trajectories and user information, then a fully connected classifier is built to predict the future trajectory by combining mobility patterns and the current position. The servers along the trajectory consist of candidate servers set  $CS$ , which considering the frequency mobility of users in the recommendation process.



**Fig. 2** Framework of mobility-aware personalized service recommendation approach

2. QoS prediction: The QoS data provided by the candidate server set  $CS$  will be predicted in this stage. We propose a personalized service recommendation approach, that considers the similarity of both users and servers to weaken the volatility of QoS data. QoS data on each edge server is personalized, which calculates on the basis of similar users and similar servers.
3. Service recommendation: According to predicted QoS data along the trajectory, we make service recommendations to meet user’s personalized needs.

#### 4.1 A hybrid LSTM-based mobility prediction model

In this section, textcoloredLSTMM is presented to discovery user mobility patterns and predict the user trajectory, then the edge servers around the trajectory will be the candidate servers recommended.

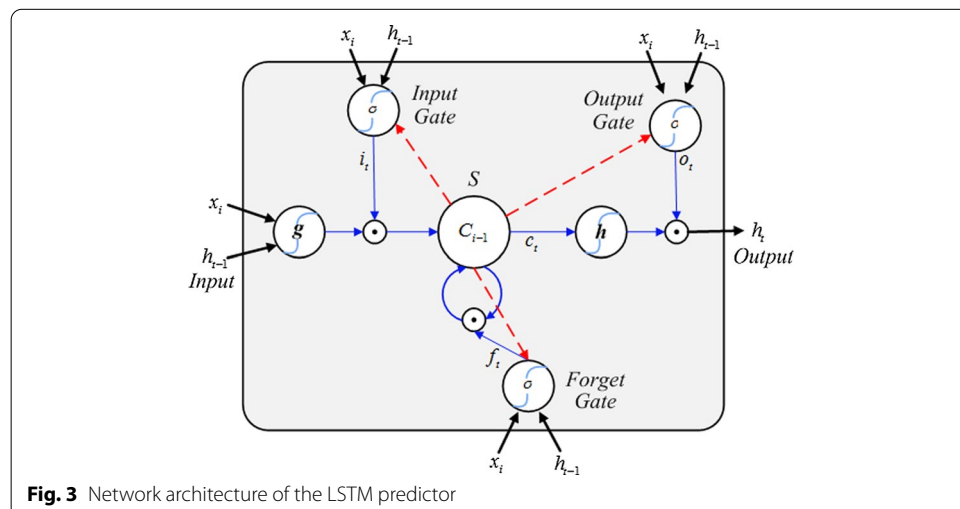
1. User mobility patterns discovery

Each trajectory has different spatio points and variable temporal distances. To process these long-term variable length trajectories, we employ LSTM network to discover user mobility patterns. LSTM is a particular implementation of recurrent neural network, which can learn long-term dependency information and avoid the problem of gradient disappearance. It is generally considered much suitable for time series. The network architecture of the LSTM predictor is schematically presented in Fig. 3.

Specially, we take the user information and the historical trajectories as the input to model the characteristic of mobility patterns. Before training the model, the spare representation of user and location information are converted into a dense vector. Position vector  $pos_i$  is embedded to  $\mathbf{p}_i$  by

$$p_i = V_p \cdot pos_i, \tag{1}$$

where  $V_p$  is an trainable embedding matrix for all positions, and  $pos_i$  is an one-hot vector of a spatio point. User information vector  $u_j$  is embedded to  $\mathbf{u}_j$  by



**Fig. 3** Network architecture of the LSTM predictor



$$u_j = V_u \cdot us_j, \tag{2}$$

where  $V_u$  is an trainable embedding matrix for all positions, and  $us_j$  is an one-hot vector of a user.

For the trajectory vector  $\mathbf{Tr}(\mathbf{u}_j)_t = \{\mathbf{p}_s, \mathbf{p}_{s+1}, \dots, \mathbf{p}_t\}$ , let  $h_t$  and  $h_{t+1}$  denote the previous and current embedding state respectively. The basic LSTM model, which takes  $\mathbf{p}_i$  and  $h_t$  as inputs, the implementation is as follows:

$$\begin{aligned} f_{t+1} &= \sigma(W_f[h_t, p_{t+1}] + b_f) \\ i_{t+1} &= \sigma(W_i[h_t, p_{t+1}] + b_i) \\ o_{t+1} &= \sigma(W_o[h_t, p_{t+1}] + b_o), \end{aligned} \tag{3}$$

where  $f_{t+1}, i_{t+1}, o_{t+1}$  are the forget gate, input gate and output gate respectively,  $\sigma$  stands for the standard sigmoid function,  $W$  are different gate parameters, and  $b$  is the bias. The memory cell  $c_{t+1}$  is updated by partially replacing the existing memory unit  $c_t$  with a new cell as:

$$c_{t+1} = f_{t+1}c_t + i_{t+1}\sigma(W_c[h_t, p_{t+1}] + b_c). \tag{4}$$

The embedding of the trajectory  $\mathbf{Tr}(\mathbf{u}_j)_{t+1}$  which expresses the mobility pattern information is then output by,

$$h_{t+1} = o_{t+1} \tanh(c_{t+1}). \tag{5}$$

## 2. Users trajectory Prediction

To predict the following trajectory of an active user, a fully connected encoder takes the encoded  $\mathbf{u}_j$  and  $\mathbf{Tr}(\mathbf{u}_j)_t$  as input and predicts the probability  $Pr_{t+l}$  of user  $u_j$  belonging to each place  $P_i$  in the future  $t+l$  time by

$$Pr_{t+l} = \text{softmax}(fc_{t+l}), \tag{6}$$

where  $fc_{t+l}$  is the output of feature data after fully connection operation in the  $l_t h$  branch,  $\sigma$  stands for the standard ReLU function [29]. Then, a softmax layer is used to normalizes the output value  $fc_{t+l}$  whose formula is given as,

$$fc_{t+l} = W_{i2}\sigma(W_{i1}[u_j, tr_j] + b_{i1}) + b_{i2}, 1 \leq i \leq seq \tag{7}$$

Besides, due to the limitation of users movement, the distance among the current server and the future servers is also an important factor during the trajectory prediction.  $dist(pos_i) = [d_1, d_2, \dots, d_j, \dots, d_{|P|}]$  is defined as a  $|P|_{th}$  vector represents the distance between the current server position  $pos_i$  and the positions of other edge servers. A typical geographic distance algorithm [30] is adopted to calculate the distance  $d_j(1 \leq j \leq |P|)$ , which means the distance between  $pos_i$  and  $pos_j$ , where  $pos_i$  and  $pos_j$  specified by the coordinates  $\langle x_i, y_i \rangle$  and  $\langle x_j, y_j \rangle$ ,

$$dist(pos_i)_j = 2R \arctan\left(\frac{\sqrt{\text{hav}(\theta)}}{1 - \sqrt{\text{hav}(\theta)}}\right), \tag{8}$$



with  $hav(\theta) = \sin 2(\frac{y_1 - y_2}{2}) + \cos y_1 \cos y_2 \sin 2(\frac{x_1 - x_2}{2})$ , where  $\theta$  is a central angle between the two edge servers and  $R$  is the radius of the Earth, which we assume to be 6371 km.

Considering the distance effect on the future position result, a parameter  $\alpha (0 \leq \alpha \leq 1)$  is employed to adjust the final results. The probability of position on  $t + l$  time can be formula by,

$$P_{t+l} = \text{softmax}((1-\alpha) \cdot Pr_{t+l} + \alpha \cdot \text{dist}(\text{pos}_i)). \quad (9)$$

### 3. Implementation

The model is trained by maximizing the likelihood of the true position being generated from the predicted value. Hence, we jointly learn all the parameters by minimizing the negative cross entropy loss  $L = - \sum_{l=1}^m y_l \log(P_l | W, b)$  for the future trajectory. An important aspect of the training phase is all parameters should be learned jointly, since the hybrid LSTM layers of the encoder and decoder units are trained in an end to end fashion. Thus, an adaptive gradient descent algorithm is used to backpropagate the loss for each trajectory at every time slot  $t$ . The pseudo-code of the trajectory prediction is shown as Algorithm 1.

---

#### Algorithm 1 Mobility Prediction

---

**Input:** The trajectory set  $Tr$ ; The user set  $U$ ; The trajectory length  $lag$ ; The prediction trajectory length  $seq$ ;  
**Output:** Trained model

- 1: // Stage 1. Data preprocessing.
- 2: Obtain all spatio points in  $Tr$ , calculate the distance matrix  $D$  between each points, and construct an empty train set  $TS$ ;
- 3: Construct train set  $TS$ ;
- 4: **for** each  $u \in U$  **do**
- 5:     Extract the sub-trajectory  $St(u)$  of length is  $(lag+seq)$  from  $Tr(u)$
- 6:     Split  $St(u)$  into  $tr=(p_1, \dots, p_{lag})$ ,  $tp=(pre_1, \dots, pre_{seq})$
- 7:     Get distance vector  $dist_p$  between the current location  $p_{lag}$  and all locations from the distance matrix  $D$
- 8:     Add  $(u, tr, tp, dist_p)$  to  $TS$
- 9: **end for**
- 10: // Stage 2. Model parameter training.
- 11: Initialize model parameters
- 12: **repeat**
- 13:     **for** each  $ts \in TS$  **do**
- 14:         Get  $(u, tr, tp, dist_p)$  from  $ts$
- 15:          $emb_u = emb(u)$  //According to Eq.(1)
- 16:          $emb(p_i) = emb(p_i)$ ,  $p_i \in tr$  //According to Eq.(2)
- 17:         Get trajectory vector representation  $v_{tr}$  of the trajectory  $tr$  by enter  $emb(p_i)$  to LSTM
- 18:          $join = \text{concat}(emb_u, v_{tr})$
- 19:         **for** each  $1 \leq l \leq seq$  **do**
- 20:             Calculate predict probability  $Pr_{t+l}$  by Formula (6)
- 21:             Calculate per time probability of belonging to each known location  $P_l$  by Formula (9)
- 22:             Calculate  $lossEntropy(P_l, \text{onehot}(pre_l))$
- 23:         **end for**
- 24:         Update model parameters by adaptive gradient descent
- 25:     **end for**
- 26: **until** convergence

---

Algorithm 1 takes the trajectory set  $Tr$ , the user set  $U$ , trajectory length  $lag$  and prediction length  $seq$  as input, and output is a trajectory predicted model with stable parameters. Line 1-9 is the data pre-processing stage, where an empty training set is constructed in line 2, and the distance matrix  $D$  between each spatio point is

calculated in Line 5-7. Lines 5 and 6 indicate that the sub-sequence  $St(u)$  of length  $(lag + seq)$  is extracted from the user trajectory  $Tr(u)$ , then the front  $lag$  locations of each sub-sequence are used as training data, and the  $seq$  locations are used as training labels, and line 7 get the distance  $dist_p$  between the current location  $p_{lag}$  and all locations from the distance matrix  $D$ . Line 10-26 is the stage of model training, line 11 initializes all trainable parameters of the model. The line 15 and 16 convert the sparse representation of the user and the location into a dense vector  $emb_u$  and  $emb(p_i)$  through the trainable embedding matrix, and line 17 inputs the sequence location vector into the LSTM-cell to obtain the vector representation encoder  $v_{tr}$  of the trajectory. Vector **join** is to connect the user's representation and the trajectory representation as a user trajectory pattern vector and as an input to the location prediction classifier in line 18. The line 20 and 21 calculate the output of the location prediction classifier and output the final prediction probability  $P_{t+l}$  in conjunction with the location distance according to the Formula (6) and (9). Line 22 calculates the training loss and line 24 update all trainable parameters by adaptive gradient descent.

#### 4.2 Mobility-aware personalized service recommendation

The number of QoS data on MEC servers is not large in MEC, because of the user mobility and the service quantity. When services are provided, it can be described by a user-service-server matrix, the majority of empty data in the matrix donates that the service has never been invoked by any user on the server before, which leads to the matrix density. Besides, the QoS data vary greatly on different servers, due to the variety of edge servers. The traditional similarity calculation methods [31] do not properly handle the QoS data in different vector spaces. To overcome these shortcomings, a location-based collaborative filtering (LCF) method is proposed to recommend services.

##### 1. Normalized similarity calculation

In our approach, we first measure the similarity between servers and users to predict QoS data on certain servers. The user-service-server matrix is decomposed into user-service matrix  $US$  and server-service matrix  $SS$  firstly. Then we normalized each row of the original  $US$  and  $SS$  to eliminate data variety.

Assume the server  $s$  and  $t$  co-provide services  $J$ ,  $sdist(s, t)$  calculates the Euclidean distance between  $s$  and  $t$  in the  $|J|$ -dimensional vector space, while  $sdis_{max}$  calculates the maximal Euclidean distance in the  $|J|$ -dimensional vector space. Therefore the formula to measure the similarity between two servers  $s$  and  $t$  is as follows:

$$\begin{aligned} Sim(s, t) &= 1 - \frac{sdis(s, t)}{sdis_{max}} \\ &= 1 - \frac{\sqrt{\sum_{j \in J} \left( \frac{r_{s,j} - r_{smin}}{r_{smax} - r_{smin}} - \frac{r_{t,j} - r_{tmin}}{r_{tmax} - r_{tmin}} \right)^2}}{\sqrt{|J|}}, \end{aligned} \quad (10)$$

where  $J = J_s \cap J_t$  is the set of services co-provided by servers  $s$  and  $t$ ;  $r_{s,t}$  is the value of service  $j$  provided by server  $u$  in the server-service matrix  $SS$ ;  $r_{smin}$  and  $r_{smax}$  denote the lowest and the highest values from server  $s$  in  $SS$ , respectively;  $r_{tmin}$  and  $r_{tmax}$  denote the

lowest and the highest values from server  $t$  in  $SS$ , respectively.  $Sim(s, t) = 0$  represents that two servers are dissimilar and  $Sim(s, t) = 1$  indicates that two servers are the same.

To calculate the similarity between users, similarly, relying on normalizing service QoS data, the original user-service matrix  $US$  is also mapped into the row-normal user-service matrix where each row is in the range of  $[0,1]$ . The formula to measure the similarity between two users  $u$  and  $v$  is as follows:

$$Sim(u, v) = 1 - \frac{\sqrt{\sum_{i \in I} \left( \frac{r_{u,i} - r_{umin}}{r_{umax} - r_{umin}} - \frac{r_{v,i} - r_{vmin}}{r_{umax} - r_{umin}} \right)^2}}{\sqrt{|I|}}, \tag{11}$$

where  $I = I_u \cap I_v$  is the set of services co-invoked by users  $u$  and  $v$ ;  $|I|$  is the number of  $I$ ;  $r_{u,i}$  is the QoS value of service  $i$  from user  $u$  in the user-service matrix  $P$ ;  $r_{umin}$  and  $r_{umax}$  denote the lowest and the highest values from user  $u$  in  $US$ , respectively;  $r_{vmin}$  and  $r_{vmax}$  denote the lowest and the highest values from user  $v$  in  $US$ , respectively.  $Sim(u, v) = 0$  represents that two users are dissimilar and  $Sim(u, v) = 1$  indicates that two users are the same.

## 2. Location-based Collaborative Filtering

Based on the normalized similarity measurement approach, we propose a Location-based Personalized QoS Prediction method. In our approach, the prediction value is calculated by the value of similar users and similar servers.

When predicting the unknown QoS value  $\hat{r}_{u,i}$  of user  $u$  on service  $i$ , our approach recover the original scale of user  $u$  or service  $i$ . In the user-based QoS value prediction, we define our user-based CF as follows:

$$\hat{r}_{u,i} = r_{umin} + (r_{umax} - r_{umin}) \frac{\sum_{u' \in U} Sim(u, u') * nr_{u',i}}{\sum_{u' \in U} Sim(u, u')}, \tag{12}$$

where  $U$  denotes the set of similar users to user  $u$ , who have invoked service  $i$ ;  $nr_{u',i}$  is the value of service  $i$  from user  $u'$  in the row-normal matrix  $US_{nu}$ ;  $r_{umin}$  and  $r_{umax}$  are the lowest and the highest values from user  $u$  in the original matrix  $US$ , respectively; and  $Sim(u, u')$  can be computed by formula (11).

In the server-based value prediction, we first cluster servers using K-means [32] to reduce the influence of the servers sparsity. Then we create server-based CF to predict the unknown QoS value  $\hat{r}_{s,i}$  of server  $s$  when provides service  $i$ , whose formula is given as:

$$\hat{r}_{s,i} = r_{smin} + (r_{smax} - r_{smin}) \frac{\sum_{s' \in S} Sim(s, s') * nr_{s',i}}{\sum_{s' \in S} Sim(s, s')}, \tag{13}$$

where  $S$  denotes the set of similar servers to server  $s$ , who have provided service  $j$ ;  $nr_{s',j}$  is the value of service  $j$  provided by server  $s'$  in the row-normal matrix  $SS_{nu}$ ;  $r_{smin}$  and  $r_{smax}$  are the lowest and the highest values from server  $s$  in the original matrix  $SS$ , respectively; and  $Sim(u, u')$  can be computed by formula (10).

To make use of the information from both similar users and similar servers, a parameter  $\lambda (0 \leq \lambda \leq 1)$  is employed to determine how much does the prediction rely

on user-based CF or server-based CF. Our approach makes prediction by employing the following equation:

$$\hat{r}_{s,u,i} = \lambda \times \hat{r}_{u,i} + (1 - \lambda) \times \hat{r}_{s,i}, \quad (14)$$

when  $\lambda = 1$ , the approach only uses information of similar users to make prediction. When  $\lambda = 0$ , it uses information of similar server for predicting the QoS data. When  $0 < \lambda < 1$ , it systematically combines the user-based CF approach and the item-based CF approach to fully utilize the information of both similar users and similar servers.

### 3. Service Recommendation

After predicting an active user trajectory, the mobile computing platform will choose servers around the trajectory and predict the QoS data on these servers. We assume that the best position is the center of the predicted trajectory which is the best distance to provide services for the user. servers around the best (within  $b$ km radius) will be considered to provide good QoS data for the active user. The mobile edge computing platform will calculate the QoS data of the service on these servers based on LCF, and recommend the active user to use that on the new edge server with the best QoS data or still use the service provided by the current edge server. In this way, personalized service recommendations can be achieved without conducting the expensive and time-consuming real-world service invocations on the edge server. The process of mobility-aware personalized service recommendation(MPSR) for an active user is shown in Algorithm 2.

---

#### Algorithm 2 MPSR Algorithm

---

**Input:** an active user  $u \in U$ ; History trajectories  $htr$  of  $u$ ; The service  $s$  invoked by  $u$ ; Trajectory length  $lag$ ; Prediction Trajectory  $seq$ ; Neighbor's search radius  $r$   
**Output:** Server with the best QoS

- 1:  $p\_servers = MobilityPrediction(u, htr, lag, seq)$  //Using the trained predicted locations model to obtain a prediction sequence of length  $seq$
- 2:  $center\_p = get\_center(p\_servers)$  // find servers center.
- 3:  $nb\_servers = get\_neighbors(center\_p, r)$  // Find neighbors near the center point within radius  $r$ .
- 4:  $QoS = dictionary()$  //Initialize a candidate set and save the QoS of each server.
- 5: **for** each  $s \in nb\_servers$  **do**
- 6:      $QoS[s] = Get\_QoS(u, b, s)$  //According to Eq.(14)
- 7: **end for**
- 8:  $best\_server = get\_best\_server(QoS)$  // find the best  $QoS$  server
- 9: **return**  $best\_server$

---

## 5 Results and discussion

In this section, we present experiments to verify the performance of our approach and compare the results with other recommendation methods. Our experiments are intended to (1) validate the rationality of our proposed approach; (2) compare our approach with other trajectory prediction methods; (3) compare our approach with other recommendation methods without considering the mobility; and (4) analyze parameters of our approach to achieve optimum performance.

### 5.1 Dataset description

In our experiments, we merge two real-world datasets on data fusion [33]: the WSDream dataset [20] and the Shanghai Telecom base station dataset [7]. The WSDream dataset contains the real-world QoS evaluation obtains from 142 users on 4500 web services in 64-time intervals, QoS evaluations contain response time, throughput, etc. The Shanghai Telecom dataset contains the information about 6357 users invoked on 3233 base station, it is regarded as user historical trajectories. The base stations are considered as edge servers in our experiment. A hybrid dataset is constructed according to the previous work [7], which contains both QoS data, edge server locations, and user trajectories.

### 5.2 Evaluation metric

Since our approach tries to predict top-K candidate positions for each user trajectory, we adopt the accuracy of top-K(ACC@K) [34] to evaluate the performance. ACC@K is the percentage of accurate predictions, i.e., correctly predicts the following positions for users, which is calculated as:

$$ACC@K = \frac{|y \text{ in top-K}|}{N}, \quad (15)$$

To evaluate the QoS value prediction accuracy, we adopt the commonly used Mean Absolute Error(MAE) [35] and Root Mean Squared Error(RMSE) [35] as the accuracy metrics. MAE is defined as follows:

$$MAE = \frac{\sum_{u,i} |r_{u,i} - \hat{r}_{u,i}|}{N}, \quad (16)$$

where  $r_{u,i}$  is the actual QoS value of user  $u$  using service  $i$ ,  $\hat{r}_{u,i}$  is the corresponding predicted QoS value,  $N$  is the total number of test samples. The RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{u,i} (r_{u,i} - \hat{r}_{u,i})^2}{N}}, \quad (17)$$

the meaning of the  $r_{u,i}$ ,  $\hat{r}_{u,i}$  and  $N$  are the same as the formula (16). For the position samples, the smaller the values of RMSE are, the more accurate the prediction.

### 5.3 Performance comparison on QoS prediction

For the sake of demonstrating the performance and effectiveness of our location-based collaborative filtering recommendation approach(LCF). We compare the MAE and RMSE with other approaches. The other approaches are as follows:

- **User-Based Collaborative Filtering Recommendation Approach (UCF):** This approach is to predict QoS values by user-based prediction algorithm using Pearson correlation coefficient, employs similar users for service recommendation;
- **Service-Based Collaborative Filtering Recommendation Approach(SCF):** This approach is to predict QoS values by item-based prediction algorithm using Pearson correlation coefficient, employs similar web services for service recommendation;

- Hybrid Collaborative Filtering Recommendation Approach (HCF):** This approach is to predict QoS values by combining the user-based and service-based prediction approach for service recommendation;

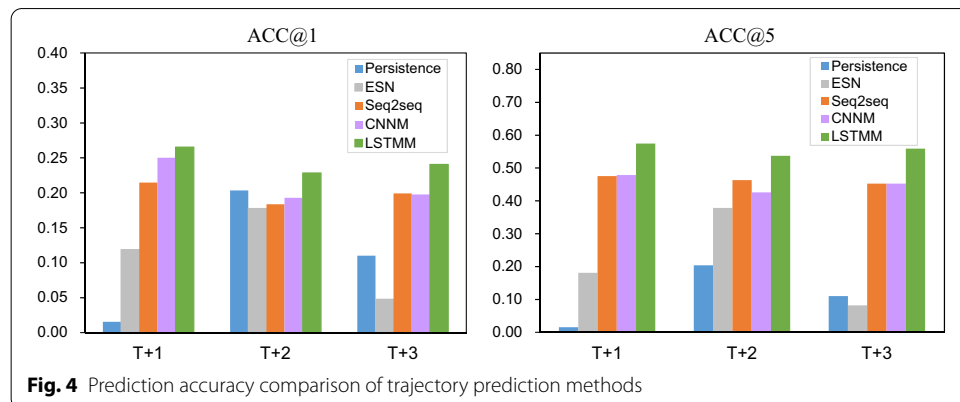
We make service recommendations in terms of qualities of service(response time and throughput) for randomly active users and compare our approach with others. The experimental results are shown in Table 1.

The results show that the MAE and RMSE values of our approach are all smaller than other approaches under different matrix density. We set the matrix density from 10% to 90%. The MAE and RMSE values increase gradually when the matrix density is set as 10% to 70%, which shows that prediction accuracy decreases. Because prediction results are in deviation when less information on similar users and services. The MAE and RMSE values increase dramatically when the matrix density increases 90%, because similar users and servers are rare. The MAE and RMSE values of our approach are smaller than other approaches, indicating that the prediction accuracy can be improved by our approach because we consider the volatility and the mobility. We take the similarities of both users and servers into consideration. In MEC environment, QoS data provided by different edge servers and various conditions, the QoS prediction provided by similar servers will eliminate volatility effectively. Besides, we normalize QoS data before calculating similarities.

#### 5.4 Performance comparison on trajectory prediction

For the sake of demonstrating the trajectory prediction accuracy of LSTMM. We compare the ACC@K with *persistent*(benchmark), *Seq2Seq* [36] and *CNNM* [37].we randomly select 20% data from our dataset to verify the trajectory prediction accuracy and compare our approach with other methods.

The length of the predicted trajectory is not long in terms of the prediction accuracy. The literature [10] shows that with the length of the predicted trajectory increases, the prediction accuracy will gradually decrease. In this experiment, we only predict the trajectory of the next  $(T + 3)_{th}$  time. Figure 4 shows the ACC@1 and ACC@5 values in different prediction trajectory length. The row labels  $T + 1$ ,  $T + 2$ , and  $T + 3$  express positions on the  $(T + l)_{th}$  time in the predicted trajectory. The difference between prediction accuracy in each position is small because they are predicted at the same time.



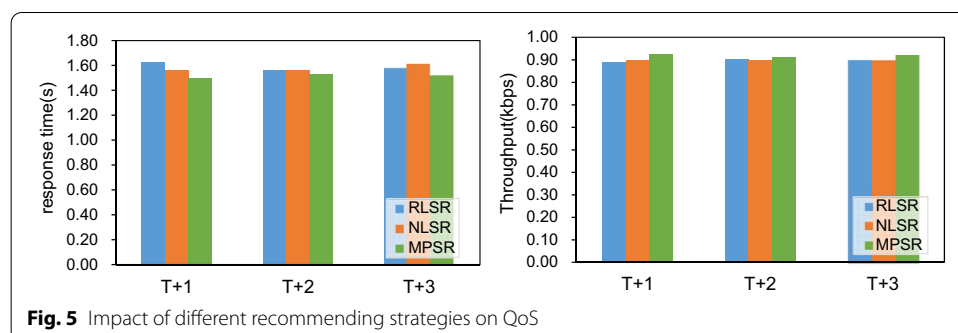
When the ACC@5 chooses the top 5 candidate position as the correct position, its value is much better than ACC@1, which only chooses the most likely position. Compared with other approaches, the ACC@1 and ACC@5 values of our approach are greater slightly. Compared with the Seq2Seq model only uses LSTM coding, the ACC@1 value of our approach increases by about 4% on average due to considering user characteristics and distances among servers. CNNM also results in poor prediction performance, because it is depend on the convolutional network, which is unsuitable to solve the long-term dependence problem. Therefore, our approach takes advantage of long-term and short-term dependence and considers more information, which lead to more accurate results than other approaches.

### 5.5 Impact of recommendation strategies

Trajectory prediction plays an important role in service recommendation. We analysis the impact from two aspects, first we compare the recommended QoS data with our approaches to other recommending strategies:

- **Random Location Service Recommendation(RLSR):** Randomly select a server around the current server, it will be the candidate server providing the service at the next time.
- **Nearest Location Service Recommendation(NLSR):** Select the nearest server to the current server as the candidate server providing the service at the next time.

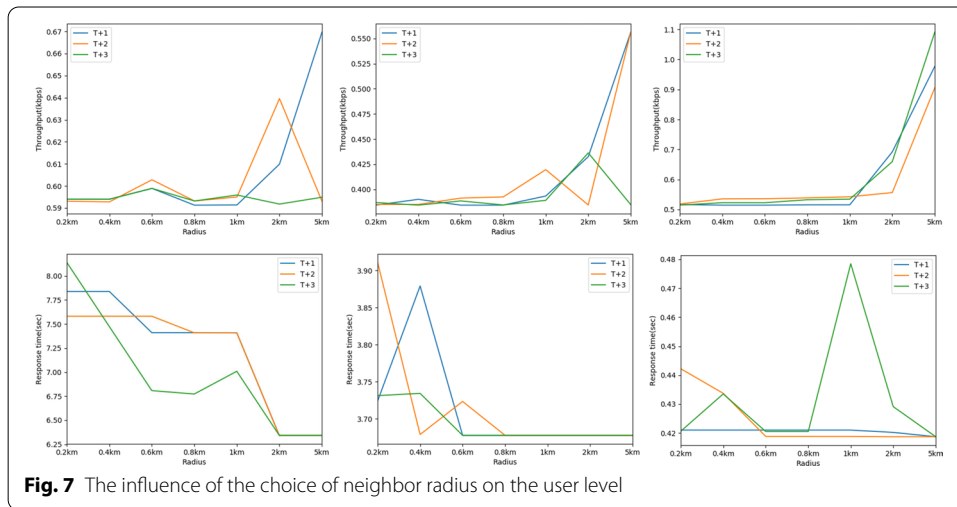
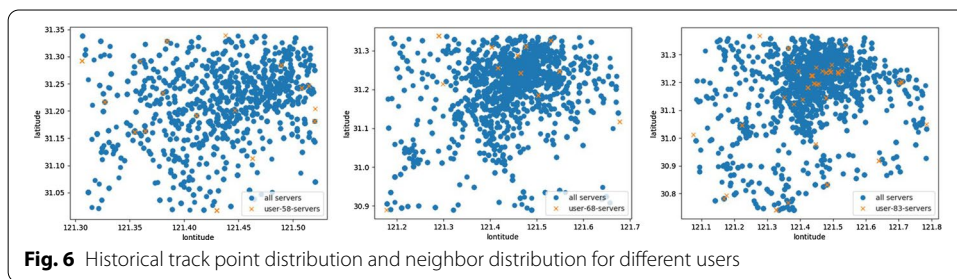
The experimental results are shown in Fig. 5. We set the prediction length of the trajectory to be next  $T + 1$ ,  $T + 2$ , and  $T + 3$  times. With the trajectory prediction times increasing, QoS data changes slightly in our approach, which shows that our approach can provide a consistent quality during the movement of users. The QoS changes impermanence in RLSR and NLSR, because of recommending services randomly. It also illustrates that the nearest server may not always provide the best QoS. Besides, our approach reduces the number of handoffs among servers. In our approach, we choose the server with the best QoS which around the center of predicted trajectory as the recommendation, therefore, we only handoff the service one time during prediction. Comparatively, the service has to transfer once when a user moves each time in RLSR and NLSR, which will increase the loading and decrease the performance of the network. The limitation of our approach is the prediction length cannot be too long, which will decrease the QoS with the expanding area.





### 5.6 Impact of MES distribution

Figure 6 shows the distribution of historical trajectory points for three typical active users randomly selected from the dataset. The distribution of each active user historical trajectory point and edge servers are very different. For user 58, the distribution of both trajectory points and the edge servers are well-proportioned. The distribution of trajectory points is well-proportioned, and the distribution of edge servers are disproportioned for user 68. Both of the historical trajectory points of users 83 and the surrounding servers are distributed disproportioned. Then, we analysis the recommendation results for the three active users shown in Fig. 7. The results show that the greater the QoS data provided, the greater the range of distances. There are also some slight



**Table 1** Impact of data sparsity on prediction accuracy

Methods	10%		30%		50%		70%		90%	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Response time										
UCF	3.9938	1.8100	4.0156	1.8179	4.0232	1.8260	4.0387	1.8337	4.0610	1.8665
SCF	3.4903	1.7540	3.5075	1.6987	3.5328	1.6317	3.5789	1.5558	3.6934	1.5177
TCF	3.6131	1.4923	3.6648	1.4838	3.7054	1.4795	3.7644	1.4994	3.8547	1.5740
LCF	<b>2.5627</b>	<b>1.2895</b>	<b>3.0548</b>	<b>1.4132</b>	<b>3.2744</b>	<b>1.4561</b>	<b>3.2966</b>	<b>1.4820</b>	<b>3.4647</b>	<b>1.7172</b>

The significance of "bold and italic values" is used to highlight the data in LCF is better than other methods

fluctuations, which are caused by the dis-proportioned distribution of edge servers. The throughput increases obviously when the range of distance is more than 1 km, and the response time reduces when the range of distance is between 0.8 km to 2 km. On the whole, the QoS data changes are less obvious with radius increases after 2 km. Hence, we predict QoS data in the range between 0.8 km and 2 km for active users to examine which distance ranges provided the most accurate service recommendations.

### 5.7 Impact of parameter $\lambda$

Different geographical areas may have different data correlation characteristics. The parameter  $\lambda$  makes our prediction method more feasible and adaptable to different data sets. We fuse information from both similar users and similar servers to predict the missing value for active users. To study the impact of the parameter  $\lambda$  to our hybrid collaborative filtering method, we vary the value of  $\lambda$  from 0 to 1 with a step value of 0.1. If  $\lambda = 1$ , we only consider information from similar users, and if  $\lambda = 0$ , we only consider information from similar servers. MAE and RMSE are compared to study the effect of the  $\lambda$  for the prediction accuracy. The experimental results are shown in the Table 2, which shows that MAE gains the best performance when  $\lambda = 0.8$  (corresponding to the bold and italic value in column MAE in Table 2) and RMSE gains the best performance when  $\lambda = 0.7$  (corresponding to the bold and italic value in column RMSE in Table 2). It means that user-based collaborative filtering has a greater influence on the prediction results on our dataset. Because in our experimental data, the number of servers is larger than the number of users, the average number of samples per user is much larger than the number of samples corresponding to each server, which results in the sparsity of the server-service matrix. Therefore,  $\lambda$  will prefer to set 0.7 as the fusion factor in the experiment.

## 6 Conclusion

In this paper, we investigate the problem of service recommendation in Mobile Edge computing for frequent mobility scenarios. We propose a two-stage approach for predicting QoS values on mobile edge servers by systematically combining user mobility and location-based Collaborative filtering. We first present a hybrid LSTM-based mobility prediction approach to obtain an active user trajectory by jointly consider server information, user characteristics and distance information. Then a mobility-aware personalized service recommendation approach is proposed to acquire the QoS values based on the mobility results. We conduct simulation experiments to study the performance of the proposed work base on the real dataset. Large-scale real-world experiments are

**Table 2** The effect of lambda taking different values on prediction accuracy

$\lambda$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Response time										
MAE	1.7542	1.6131	1.5177	1.4550	1.4168	1.3853	1.3620	<b>1.3413</b>	1.3462	1.3501
RMSE	3.8204	3.5451	3.2867	3.0712	2.9005	2.7829	<b>2.7253</b>	2.7314	2.8009	2.9292

The significance of "bold and italic values" highlights the best value in MAE and RMSE respectively, which is used to choose the best  $\lambda$

conducted and the comprehensive experimental results show the effectiveness and feasibility of our approach.

In our future study, we will research on more trajectory prediction method, e.g., social GAN and ESN, can be employed to predict the future position of users to further improve the QoS values prediction. And other factors should be considered, e.g., the impact of time, when the service is recommended in the mobile edge computing.

#### Abbreviations

MEC: Mobile edge computing; MPSR: Mobility-aware personalized service recommendation; QoS: Quality of service; MEN: Mobile edge network; LSTM: Long short-term memory; LSTMM: LSTM-based mobility prediction model; WSRec: Web service recommendation system; PCC: Pearson correlation coefficient; MESS: Multiple mobile edge servers; LCF: Location-based collaborative filtering; CF: Collaborative filtering; MAE: Mean absolute error; RMSE: Root mean squared error; UCF: User-Based collaborative filtering; SCF: Service-Based collaborative filtering; HCF: Hybrid collaborative filtering; RLSR: Random location service recommendation; NLSR: Nearest location service recommendation; GAN: Generative adversarial networks; ESN: Echo State Networks.

#### Acknowledgements

Not applicable.

#### Authors' contributions

HZ, YD, and YY conceived and designed the study. YD performed the simulations. HZ and YY wrote the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

#### Funding

This work was partially supported by the Major Scientific and Technological Projects of CNPC, China, under Grant ZD2019-183-004, and by the Fundamental Research Funds for the Central Universities, China, under Grant 20CX05019A.

#### Availability of data and materials

The details of experimental parameters are given in Section Experiment.

#### Declarations

#### Competing interests

The authors declare that they have no competing interests.

Received: 24 November 2020 Accepted: 17 November 2021

Published online: 04 December 2021

#### References

1. P. Mach, Z. Becvar, Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun. Surv. Tutor.* **19**(3), 1628–1656 (2017)
2. Y. M. Saputra, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, A novel mobile edge network architecture with joint caching-delivering and horizontal cooperation. *IEEE Trans. Mobile Comput.* **20**(1), 19–31 (2021)
3. L. Chiaraviglio, F. Cuomo, A. Gigli, M. Maisto, Y. Zhou, Z. Zhao, H. Zhang, A reality check of base station spatial distribution in mobile networks, in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2016* (2016), pp. 1065–1066
4. W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges. *IEEE Internet Things J.* **3**(5), 637–646 (2016)
5. K. Peng, H. Huang, S. Wan, and V. C. M. Leung, End-edgecloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment. *Wireless Netw.* **7**(4), 2622–2629 (2020)
6. Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, R. Wang, User mobility aware task assignment for mobile edge computing. *Future Gener. Comput. Syst.* **85**, 1–8 (2018)
7. S. Wang, Y. Zhao, L. Huang, J. Xu, C.H. Hsu, QoS prediction for service recommendations in mobile edge computing. *J. Parallel Distrib. Comput.* **127**, 134–144 (2017)
8. Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, Z. Mai, QoS prediction for service recommendation with deep feature learning in edge computing environment. *Mob. Netw. Appl.* **25**, 1–11 (2020)
9. X. Wu, H. Haas, Load balancing for hybrid LiFi and WiFi networks: to tackle user mobility and light-path blockage. *IEEE Trans. Commun.* **68**(3), 1675–1683 (2020)
10. R. Arshad, H. ElSawy, S. Sorour, T. Y. Al-Naffouri, M. Alouini, Handover management in dense cellular networks: a stochastic geometry approach, in *2016 IEEE International Conference on Communications (ICC)* (2016), pp. 1–7
11. B. Li, H. Zhang, H. Lu, User mobility prediction based on Lagrange's interpolation in ultra-dense networks. in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (2016), pp. 1–6
12. H. Sun, Z. Zheng, J. Chen, M.R. Lyu, Personalized web service recommendation via normal recovery collaborative filtering. *IEEE Trans. Serv. Comput.* **6**(4), 573–579 (2013)

13. Y. Li and Y. Guo, Cultural distance-aware service recommendation approach in mobile edge computing. *Sci Program*. **2018**, 1–12 (2018)
14. T. Rutkowski, J. Romanowski, P. Woldan, P. Staszewski, R. Nielek, L. Rutkowski, A content-based recommendation system using neuro-fuzzy approach, in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (2018), pp. 1–8
15. H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, X. Wang, Context-aware QoS prediction with neural collaborative filtering for internet-of-things services. *IEEE Internet Things J.* **7**(5), 4532–4542 (2020)
16. L. Yao, Q.Z. Sheng, A.H.H. Ngu, J. Yu, A. Segev, Unified collaborative and content-based web service recommendation. *IEEE Trans. Serv. Comput.* **8**(3), 453–466 (2015)
17. N. Nice, N. Koenigstein, S. Ben-Elazar, S. Keren, U. Paquet, Y. Finkelstein, Multilingual content based recommendation system (2018). <https://www.freepatentsonline.com/9898773.html>
18. L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, H. Mei, Personalized qos prediction for web services via collaborative filtering, in *IEEE International Conference on Web Services (ICWS 2007)* (2007), pp. 439–446
19. 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* (2009), pp. 437–444
20. Z. Zheng, H. Ma, R.M. Lyu, I. King, Qos-aware web service recommendation by collaborative filtering. *IEEE Trans. Serv. Comput.* **4**(2), 140–152 (2011)
21. Z. Zheng, H. Ma, R.M. Lyu, I. King, Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Trans. Serv. Comput.* **6**(3), 289–299 (2013)
22. R. Logesh, V. Subramaniaswamy, V. Vijayakumar, A personalised travel recommender system utilising social network profile and accurate gps data. *Electron. Gov. Int. J.* **14**(1), 90 (2018)
23. H. Wang, M. Terrovitis, N. Mamoulis, Location recommendation in location-based social networks using user check-in data, in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2013), pp. 374–383
24. M.-H. Park, J.-H. Hong, S.-B. Cho, Location-based recommendation system using bayesian user's preference model in mobile devices, in *Proceedings of the 4th International Conference on Ubiquitous Intelligence and Computing* (2007), pp. 1130–1139
25. V.W. Zheng, B. Cao, Y. Zheng, X. Xie, Q. Yang, Collaborative filtering meets mobile recommendation: a user-centered approach, in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (2010), pp. 236–241
26. 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), 1913–1924 (2014)
27. M. Tang, Y. Jiang, J. Liu, X. Liu, Location-aware collaborative filtering for qos-based service recommendation, in *2012 IEEE 19th International Conference on Web Services* (2012), pp. 202–209
28. E. Ben Zion, B. Lerner, Identifying and predicting social lifestyles in people's trajectories by neural networks. *EPJ Data Sci.* **45**(7), 1–27 (2018)
29. J. Schmidt-Hieber, Nonparametric regression using deep neural networks with relu activation function. *Ann. Stat.* **48**(4), 1875–1897 (2020)
30. S. Ramachandran, O. Deshpande, C.C. Roseman, N.A. Rosenberg, M.W. Feldman, L.L. Cavalli-Sforza, Support from the relationship of genetic and geographic distance in human populations for a serial founder effect originating in Africa. *Proc. Natl. Acad. Sci. U.S.A.* **102**(44), 15942–15947 (2005)
31. G. Coletti, B. Bouchon-Meunier, Fuzzy similarity measures and measurement theory, in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (2019), pp. 1–7
32. J.A. Hartigan, M.A. Wong, *A K-Means Clustering Algorithm: Algorithm AS 136*, vol. 28 (Blackwell Publishing, Hoboken, 1979), pp. 100–108
33. B. Khaleghi, A. Khamis, F.O. Karray, S.N. Razavi, Multisensor data fusion: a review of the state-of-the-art. *Inf. Fus.* **14**(1), 28–44 (2013)
34. Y. Wang, J. Wang, L. Li, Enhancing long tail recommendation based on user's experience evolution, in *2018 IEEE 22nd International Conference on Computer Supported* (2018), pp. 25–30
35. D. Ai-Lin, Z. Yang-Yong, S. Bai-Le, A collaborative filtering recommendation algorithm based on item rating prediction. *J. Softw.* **14**, 09 (2003)
36. A. Karatzoglou, A. Jablonski, M. Beigl, A seq2seq learning approach for modeling semantic trajectories and predicting the next location, in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2018), pp. 528–531
37. J. Lv, Q. Li, Q. Sun, X. Wang, T-conv: A convolutional neural network for multi-scale taxi trajectory prediction, in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)* (2018), pp. 82–89

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.