An Empirical Framework for Recommendation-based Location Services Using Deep Learning

Vinita Rohilla  
Department of Computer Science and Engineering, School of Engineering and Technology, Sharda University, Greater Noida, India and Department of CSE, Maharaja Surajmal Institute of Technology, New Delhi, India  
vinita_rohilla@msit.in

Mandeep Kaur  
Department of Computer Science and Engineering  
School of Engineering and Technology  
Sharda University  
Greater Noida, India  
mandeep.kaur@sharda.ac.in

Sudeshna Chakraborty  
Department of Computer Science and Technology  
Lloyd Institute of Engineering and Technology  
Greater Noida, India  
sudeshna2529@gmail.com

Received: 8 June 2022 | Revised: 8 July 2022 | Accepted: 10 July 2022

Abstract-The large amount of possible online services throws a significant load on the users' service selection decision-making procedure. A number of intelligent suggestion systems have been created in order to lower the excessive decision-making expense. Taking this into consideration, an RLSD (Recommendation-based Location Services using Deep Learning) model is proposed in this paper. Alongside robustness, this research considers the geographic interface between the client and the service. The suggested model blends a Multi-Layer-Perceptron (MLP) with a similarity Adaptive Corrector (AC), which is meant to detect high-dimensional and non-linear connections, as well as the location correlations amongst client and services. This not only improves recommendation results but also considerably reduces difficulties due to data sparseness. As a result, the proposed RLSD has strong flexibility and is extensible when it comes to leveraging context data like location.

Keywords-location services; recommendation system; deep learning; similarity corrector

I. INTRODUCTION

Numerous corporations and organizations have started to wrap their created business apps into more and more widespread multiple online services [1-4]. Many lightweight recommendation methods have also been created to reduce the expenses of consumer decision-making. Generally, by evaluating available client data, a recommender system can deduce a person's approximated tastes and preferences and then offer reliable suggestions [5-9]. It efficiently captures the relevant information and preferences of service clients. Location is an important component in decision recommendations since the Quality of Service (QoS) is sometimes significantly dependent on client or service location. Moreover, location-aware QoS data frequently contain sensitive user information, such as a user's most recently requested service set, the amount of time it takes a user to summon a services located in a nation, etc. Service-Oriented Architecture (SOA) is commonly employed in decentralized computing environments [10], including cloud computing and mobile computing. Location-based services are more successful to provide recommendations based on user's interests [12, 13].

Collaborative Filtering (CF) is a popular way for creating a recommendation system. In recommendation systems, the current user-based CF mechanism seeks for a comparable user set, the amount of time it takes a user to summon a services located in a nation, etc. Service-Oriented Architecture (SOA) is commonly employed in decentralized computing environments [10], including cloud computing and mobile computing. Location-based services are more successful to provide recommendations based on user's interests [12, 13].

Corresponding author: Vinita Rohilla
• The performance of the suggested strategy was assessed and compared with 9 advanced options.

• The proposed technique not only improves recommendation performance, but it also significantly reduces difficulties due to data sparseness.

II. LITERATURE SURVEY

Authors in [14] focused mostly on the cold-start dilemma for service advice depending on user and service locations. To resolve this concern, they offered a service recommendation approach based on CF and geo location, which takes into account the characteristics of services at the edges, movement, and customer requests over time. To begin, they used a multidimensional weighted approach to synthesize the service quality for every dimension in different temporal chunks. In respect of recommendations accuracy, the empirical findings suggest that their multidimensional inverse similarity recommendation method (MDITCF) built on time-aware CF surpasses the Inverted CF recommendation method. Authors in [15] suggested a novel deep CF approach to service recommendations, dubbed Location-aware Deep CF (LDCF). These main advancements included in this approach are: 1) the location attributes are located into high-dimensional dense imbed matrices. MLP encapsulates the HD and non-linear features. Similarity ACs are integrated in the output nodes to first rectify the forecasting QoS. With this, LDCF not only understands the high-dimension and non-linear correlation among the user and services, but it can also dramatically reduce data sparseness. Extensive studies on a real-world Web service [15] database show that LDCF’s recommendations efficiency clearly exceeds 9 state-of-the-art services recommendations approaches. Authors in [16] used location information in the factorization machine for QoS predictions, while contextualized information helps the CF similarity calculation. These computations understand the lower dimension and linear characteristics of customers and service providers. Authors in [17] suggested a tailored location-aware CF approach for web service recommendations. Picking comparable neighbors for the user based service the suggested technique takes advantage of both customer location and online services. The technique also incorporated an improved similarity metric for user and web services through accounting for their customized contribution. The test findings demonstrate that this methodology greatly increases QoS predictive performance and computing efficiency when compared to earlier CF-based systems.

III. PROPOSED METHODOLOGY

The major problem in recommending services is estimating QoS. CF is the most extensively utilized QoS predicting technology currently available. Conventional CF techniques, on the other hand, suffer from two flaws: 1) The similarity modeling approach used by conventional CF-based techniques can only gain knowledge of low-dimension and linear characteristics or features from previous networks of user and service and 2) the commonly used data sparseness dilemma in reality has a massive effect on recommendation systems’ efficiency. This section provides a detailed description of the proposed model and methodology. The section briefly describes the loss function, the optimizer, and the algorithm used in this work.

A. Deep Neural Networks

A Deep Neural Network (DNN) is one type of Neural Network with more than 2 layers: an input layer, at least one hidden layer, and an output layer. In a process known as “feature hierarchy,” every layer conducts certain kinds of sorting and ordering. Handling with unorganized or unstructured input is among the most important applications of highly powerful neural network models [18].

The RLSD model is a multiple-layered feed forward neural network with 3 main layers. An Input Layer is used to create the input variables needed from the Intermediate Layer and the similarities needed for the Output Nodes. The Intermediate Layer is employed for centralized training to ensure getting high-dimension and non-linear characteristics.

1) Input Layer

The Input layer’s primary function is to regulate the initial Input. The Input contains user identification, location-based info, service identifiers, and service geo locations in KerasI’s Embedding Layer, that may be thought as a special fully-connected layer lacking bias terms. Essentially, the Embedded Layer uses one-hot encoding on the inputs to build a ‘0’ vector with a given dimension, while the vector’s i-th point is set to 1 [19]. The category characteristics are translated to high-dimension fully connected embedding vectors using this procedure. The mapping procedure is depicted in (1) - (4):

\[ I_u^T = f_1(P_it_iu + b_1) \]  
\[ G_u^T = f_1(P_cg_u + b_2) \]  
\[ L_u^T = f_1(Q_i^T i_s + b_3) \]  
\[ L_u^T = f_1(Q_i^T g_s + b_4) \]

where \( i_u \) and \( i_s \) are the user’s identifier and the service’s identifier, correspondingly, \( g_u \) and \( g_s \) are the initial inputs for the user’s and service’s location, \( P_i \) and \( Q_i \) are user’s and service’s embedding weighted matrices, \( b_1 \) is a bias term.
initialized to 0, \( f_1 \) is the embedding layer’s activation function, which is considered as the identity function for this work, \( l^k_0 \) and \( G^k_0 \) are the \( k \)-dimensional user_id and location embedding vector correspondingly, and \( U \) and \( S \) are the \( k \)-dimensional service_id and location embedding vector correspondingly.

These identifying feature vectors representations are integrated with the appropriate location features set to generate the user feature map and service feature map. Next, we append both feature maps to generate the intermediate layers input vector sequence. The equation is written in the following form:

\[
U = \Phi(l^k_0, G^k_0) = \begin{bmatrix} l^k_0 \\ G^k_0 \end{bmatrix} \tag{5}
\]

\[
S = \Phi(l^k_0, G^k_0) = \begin{bmatrix} l^k_0 \\ G^k_0 \end{bmatrix} \tag{6}
\]

\[
X = \Phi(U, V) = \begin{bmatrix} U \\ V \end{bmatrix} \tag{7}
\]

where \( \Phi \) is the mergence operation, \( U \) and \( S \) are user and service embedding vectors, and \( X \) is the input vector.

An Adaptive Corrector (AC) that computes similarities among user-location embedding and service-location embedding is present. The AC employs a similarity approach, including location similarities among consumers and services into the neural network’s forward propagation phase. As a result, the AC bridges gaps across deep learning with CF. The AC’s operation outcome is immediately transferred to the Output Nodes, bypassing the Intermediate Layer. The AC is adaptable and may be used for all other similarity computations like cosine and Euclidean similarity. Equations (8) and (9) describe the procedure:

\[
O^{AC} = \text{cosider}(G_u, G_s) = \frac{G_u \cdot G_s}{\|G_u\| \|G_s\|} \tag{8}
\]

\[
O^{AC} = \text{cosider}(G_u, G_s) = \sqrt{G_u \cdot G_s} \tag{9}
\]

where \( O^{AC} \) is the AC similarity output. Cosine and Euclidean similarities are both present in this. When no specific comment is made, (8) is applied to determine the cosine similarity outcome of the AC.

2) Intermediate Layer

The Intermediate Layer is responsible for processing the input vectors from the Input Nodes in order to capture non-linear information. The high-dimensional non-linear correlation among consumers and services is learned using a densely integrated MLP architecture in this work. First but foremost, we must select the non-linear activation. The activation function SeLU (Scaled Exponential Linear Unit) presents numerous benefits. It allows building a map featuring qualities that led to SNNs (Self-normalized Neural Networks). Secondly, in order for the DNN to learn and train many attributes, the network structure must adhere to the standard tower architecture [20]. Lastly, to avoid overfitting, we employ L2 regularization on weights. In the Intermediate Layer, the forward propagating of the input vector sequence is described by:

\[
o_{m_{\text{2}}} = f_2(W^T_{2}x + b_2) \tag{10}
\]

\[
o_{m_{\text{0}}} = f_1(W^T_{1}f_{2}o_{m_{n-1}} + b_1), I = 3, 4, \ldots, N-1 \tag{11}
\]

\[
o_{m_{n_{\text{2}}}} = \alpha_{m_{n_{\text{2}}}} \tag{12}
\]

where \( o_{m_{n_{\text{2}}}} \) is the \( i \)-th layer of the Intermediate Layer’s output, \( b_1 \) the Intermediate Layer bias term, and \( o_{m_{n_{\text{2}}}} \) the Intermediate Layer’s output.

3) Output Layer

The Output Layer is largely responsible for producing the overall forecast results. The FRLS paradigm divides users and services into two paths. We immediately combine the outcomes of these two paths, as in [20, 21]. We mix the similarity outcome \( O^{AC} \) with the output vector \( o_{m_{n_{\text{2}}}} \) to create a new output vector sequence \( o \). Finally, FRLS makes the final projections using a single-layered neural network. Gaussian distribution is used to initialize the parameters of this layer, as illustrated in (13)-(14):

\[
o = \Phi(O^{AC}, o_{m_{n_{\text{2}}}}) = \begin{bmatrix} o^{AC} \\ o_{m_{n_{\text{2}}}} \end{bmatrix} \tag{13}
\]

\[
\tilde{Q}_{u,s} = f_u(W^T_{n}o + b_n) \tag{14}
\]

where \( \tilde{Q}_{u,s} \) is the predicted QoS values of user \( u \) raising the \( s \) service, and \( b_n \) is the ordinary identity function that signifies the activation function of the last layer.

We get a typical matrix factorization network base Gaussian Matrix Factorization (GMF) distribution on one side and an MLP network on the other.

B. Multi-Layer Perceptron Networks

An MLP network is simply a formal moniker for the most fundamental type of DNN. An MLP network is made up of 5 dense hidden layers and the SeLu activation function. We began by embedding the user and items, then flattened and appended them. Afterwards, we added the first dropout layer, followed by 4 hidden layers with 64 neurons, 32 neurons, 16 neurons, 8 neurons, and finally one densely integrated neuron. Then a dropout and batch normalization layer was added between hidden layers 1 and 2 and 2 and 3. Lastly, we got our final output layer, which is nothing more than a one fully-connected neuron. We utilized log cosh as the error function and Nadam as optimizer.

C. Matrix Factorization Networks

A Gaussian MF network is essentially a conventional point-wise MF that uses Stochastic Gradient Decent (Adam with this context) to approximate the factoring of a matrix (user x item) into two matrices, i.e. user x latent feature and latent feature x item.

D. Combining the two Networks: NeuMF

Finally, the Matrix factorization and MLP network are merged into the NeuMF network. Essentially, this merely appends the results of two networks. The two networks are then fused, and a single output node is added to the newly integrated model. From this, the configuration is like above, with a binary log cosh loss and the Nadam optimizer.
To improve the accuracy of representing probability distributions, many variants have been proposed. An alternative approach for improvement is through the reconstruction loss:

$$f(t; a) = \frac{1}{a} \log(\cosh(at)) = \frac{1}{a} \log \frac{e^{at} + e^{-at}}{2} \quad (15)$$

Nadam optimizer can be written as:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\tau_t} + \epsilon} \left( \beta_1 m_{t-1} + \frac{1 - \beta_1}{1 - \beta_1^T} \frac{\partial L}{\partial w_t} \right) \quad (16)$$

Nesterov is used by Nadam optimizer to modify the gradients 1 step forward by substituting the prior $m_{vec}(m_{t-1})$ in (16) with the current $m_{vec}(m_t)$:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\tau_t} + \epsilon} \left( \beta_1 m_t + \frac{1 - \beta_1}{1 - \beta_1^T} \frac{\partial L}{\partial w_t} \right) \quad (17)$$

Algorithm 1: RLSD (Recommendation based location services using the Deep Learning Algorithm).

**Required:** user_service request matrix ($Rm$), density of matrix ($Dm$), DNN topology design ($t$), learning rates ($lr$), decayed ratios ($Dr$), no. of iterations ($Ni$).

**Output:** Weighted matrix and the bias $P_1, Q_1, W_1, ..., W_n, b_1, b_2, ..., b_n$.

**Strategy:**

1. **step 1.** Scarce $Rm$ based on $Dm$.
2. **step 2.** Make training inputs entries for $R_{train}$ and test input entries $R_{test}$.
3. **step 3.** Create input attributes/features $i_u, g_u, i_s, g_s$.
4. **step 4.** Construct the DNN using $t$ and (10)-(12).
5. **step 5.** Set $P_1, Q_1, W_1, ..., W_n$ according to the Gaussian distribution.
6. **step 6.** Set $b_1, b_2, ..., b_n$ to 0.
7. **step 7.** For each epoch = 1 to $Ni$ do.
8. **step 8.** For each user and service in $R_{train}$ do.
9. **step 9.** Create embedding vectors sequence using (1)-(4).
10. **step 10.** Create input vector sequence using (5)-(7).
11. **step 11.** Create the AC output from either (8) or (9).
12. **step 12.** Make prediction, $\hat{Q}_{u,s}$ using (13)-(14).
13. **step 13.** end for.
14. **step 14.** Give $lr$ and $Dr$ to Nadam optimizer.
15. **step 15.** Modify parameters of the model using the Nadam minimization (15).
16. **step 16.** For each user and service in $R_{test}$ do.
17. **step 17.** Use (20)-(21) to measure the performance of the model.
18. **step 18.** end for.
19. **step 19.** end for.

**IV. RESULTS AND DISCUSSION**

This section discusses the experimentation results for location-based services on the WS-Dream dataset using Python programming. To assess the effectiveness of the proposed method, two evaluation measures are used.

**A. Dataset**

We ran tests upon the WS-Dream dataset which is a substantial Web services dataset gathered and managed in [22], which consists of 19,74,675 QoS values from 339 users on 5,825 services. This dataset includes information on users’ and services’ locations. QoS data are presented in this work as a user-service matrix, with the row indicator representing the user identifier and the column indicator representing the services identifier. This work utilized response time and throughput as inputs to the RLSD.

**B. Pre-processing**

This research makes use of two spatially relevant dataset features/attributes: Country Name (CN) and Autonomous System Number (ASN). Users were spread throughout 30 nations and 136 ASNs, whereas services were spread across 990 ASNs across 73 countries, according to database figures. For CN, we utilized Sklearn’s category encoding to convert the classified characteristics into numerical embeddings, such that each classified attribute is expressed by a nation code. The data can be viewed like after pre-processing:

$$U = (U_ID, UASN, UCN) \quad (18)$$

$$S = (S_ID, SASN, SCN) \quad (19)$$

where $U$ and $S$ are the user input embedding vector and service input embedding vector correspondingly, $U_ID$ is the user identifier, $UASN$ is an autonomous system numeric user code, $UCN$ is the nationwide user code, and $S_ID$, $SASN$, and $SCN$ are the same as above.

**C. Parameter Setting**

For CP methods (UPCC, LACF, IPPC, RegionKNN, etc.), the best-$k$ value is set to 10, the learning rate is fixed at $1e^{-4}$, the total of implicit feature factors (size) is fixed to 10, the number of total iterations is fixed to 300, the regularization variables are fixed to 0.1, and the randomized factor for the rarefy matrix is fixed to 7. Deep learning techniques (NCF, RLSD), can be developed using the Keras API, wherein the Gaussian distribution function is utilized (average is 0, stdev is 0.01) to set the model hyper parameters and (15) to modify the hyper parameters. Then we configured the batch size to 256, learning rate to $1e^{-4}$, 4 MLPs, and Nadam optimizer.

**D. Evaluation Metrics**

The recommendation success of the specified approach is measured using two fundamental statistical evaluation metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE):

$$MAE = \frac{\sum_u |Q_{u,s} - \hat{Q}_{u,s}|}{N} \quad (20)$$

$$RMSE = \frac{1}{N} \sum_u (Q_{u,s} - \hat{Q}_{u,s})^2 \quad (21)$$

where $Q_{u,s}$ is the QoS initial value for the user $u$ raising the service $s$, $\hat{Q}_{u,s}$ is the QoS prediction value $u$ raising the service $s$, and $N$ is the overall QoS value.

Tables I and II show the obtained results by the algorithms performed in [21], with which the proposed model of this paper is compared to. The values in the Tables show that the proposed model is better in MAE and RMSE for response time (RT) and TP (throughput).
TABLE I. RESPONSE TIME EXPERIMENTAL RESULTS

<table>
<thead>
<tr>
<th>Approach</th>
<th>Density 0.5</th>
<th>MAE 0.10</th>
<th>RMSE 0.10</th>
<th>MAE 0.15</th>
<th>RMSE 0.15</th>
<th>MAE 0.20</th>
<th>RMSE 0.20</th>
<th>MAE 0.25</th>
<th>RMSE 0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMEN</td>
<td>0.876</td>
<td>0.853</td>
<td>0.873</td>
<td>1.857</td>
<td>0.874</td>
<td>1.857</td>
<td>0.873</td>
<td>1.858</td>
<td>0.874</td>
</tr>
<tr>
<td>IMEN</td>
<td>0.703</td>
<td>0.567</td>
<td>0.686</td>
<td>1.542</td>
<td>0.684</td>
<td>1.533</td>
<td>0.681</td>
<td>1.529</td>
<td>0.680</td>
</tr>
<tr>
<td>UPCC</td>
<td>0.634</td>
<td>0.377</td>
<td>0.553</td>
<td>1.311</td>
<td>0.511</td>
<td>1.258</td>
<td>0.483</td>
<td>1.220</td>
<td>0.467</td>
</tr>
<tr>
<td>IPCC</td>
<td>0.633</td>
<td>1.397</td>
<td>0.591</td>
<td>1.341</td>
<td>0.507</td>
<td>1.258</td>
<td>0.454</td>
<td>1.208</td>
<td>0.431</td>
</tr>
<tr>
<td>UPCC</td>
<td>0.624</td>
<td>1.386</td>
<td>0.579</td>
<td>1.328</td>
<td>0.498</td>
<td>1.247</td>
<td>0.448</td>
<td>1.197</td>
<td>0.425</td>
</tr>
<tr>
<td>RegionKNN</td>
<td>0.594</td>
<td>1.641</td>
<td>0.577</td>
<td>1.637</td>
<td>0.557</td>
<td>1.627</td>
<td>0.560</td>
<td>1.617</td>
<td>0.562</td>
</tr>
<tr>
<td>LACF</td>
<td>0.682</td>
<td>1.500</td>
<td>0.650</td>
<td>1.468</td>
<td>0.610</td>
<td>1.416</td>
<td>0.582</td>
<td>1.381</td>
<td>0.562</td>
</tr>
<tr>
<td>PMF</td>
<td>0.568</td>
<td>1.537</td>
<td>0.487</td>
<td>1.321</td>
<td>0.451</td>
<td>1.221</td>
<td>0.430</td>
<td>1.171</td>
<td>0.416</td>
</tr>
<tr>
<td>NCF</td>
<td>0.440</td>
<td>1.325</td>
<td>0.385</td>
<td>1.283</td>
<td>0.372</td>
<td>1.253</td>
<td>0.362</td>
<td>1.205</td>
<td>0.349</td>
</tr>
<tr>
<td>LDCF</td>
<td>0.246</td>
<td>0.739</td>
<td>0.278</td>
<td>0.895</td>
<td>0.229</td>
<td>0.757</td>
<td>0.237</td>
<td>0.800</td>
<td>0.258</td>
</tr>
<tr>
<td>RLSD (proposed)</td>
<td>0.176</td>
<td>0.497</td>
<td>0.169</td>
<td>0.530</td>
<td>0.189</td>
<td>0.615</td>
<td>0.183</td>
<td>0.597</td>
<td>0.171</td>
</tr>
</tbody>
</table>

TABLE II. THROUGHPUT EXPERIMENTAL RESULTS

<table>
<thead>
<tr>
<th>Approach</th>
<th>Density 0.5</th>
<th>MAE 0.10</th>
<th>RMSE 0.10</th>
<th>MAE 0.15</th>
<th>RMSE 0.15</th>
<th>MAE 0.20</th>
<th>RMSE 0.20</th>
<th>MAE 0.25</th>
<th>RMSE 0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMEN</td>
<td>54,333</td>
<td>110,296</td>
<td>53,947</td>
<td>110,345</td>
<td>53,971</td>
<td>110,201</td>
<td>53,906</td>
<td>110,190</td>
<td>53,862</td>
</tr>
<tr>
<td>IMEN</td>
<td>27,342</td>
<td>65,844</td>
<td>26,962</td>
<td>64,843</td>
<td>26,757</td>
<td>64,266</td>
<td>26,669</td>
<td>64,069</td>
<td>26,595</td>
</tr>
<tr>
<td>UPCC</td>
<td>27,559</td>
<td>60,757</td>
<td>22,687</td>
<td>54,598</td>
<td>20,525</td>
<td>50,906</td>
<td>19,243</td>
<td>48,834</td>
<td>18,253</td>
</tr>
<tr>
<td>IPCC</td>
<td>27,102</td>
<td>62,665</td>
<td>26,270</td>
<td>60,479</td>
<td>25,487</td>
<td>57,561</td>
<td>23,726</td>
<td>54,564</td>
<td>22,286</td>
</tr>
<tr>
<td>UPCC</td>
<td>27,070</td>
<td>60,510</td>
<td>22,440</td>
<td>54,506</td>
<td>20,256</td>
<td>50,585</td>
<td>18,888</td>
<td>48,238</td>
<td>17,863</td>
</tr>
<tr>
<td>RegionKNN</td>
<td>26,857</td>
<td>69,614</td>
<td>25,352</td>
<td>68,015</td>
<td>24,947</td>
<td>67,365</td>
<td>24,687</td>
<td>66,923</td>
<td>24,746</td>
</tr>
<tr>
<td>LACF</td>
<td>27,419</td>
<td>65,770</td>
<td>24,847</td>
<td>62,057</td>
<td>22,943</td>
<td>58,816</td>
<td>21,562</td>
<td>56,507</td>
<td>20,587</td>
</tr>
<tr>
<td>PMF</td>
<td>18,943</td>
<td>57,020</td>
<td>16,004</td>
<td>47,933</td>
<td>14,668</td>
<td>43,642</td>
<td>13,988</td>
<td>41,652</td>
<td>13,398</td>
</tr>
<tr>
<td>NCF</td>
<td>15,468</td>
<td>49,703</td>
<td>13,616</td>
<td>46,034</td>
<td>12,284</td>
<td>42,317</td>
<td>11,833</td>
<td>41,263</td>
<td>11,312</td>
</tr>
<tr>
<td>LDCF</td>
<td>17,629</td>
<td>57,746</td>
<td>14,711</td>
<td>50,730</td>
<td>13,952</td>
<td>48,871</td>
<td>13,199</td>
<td>47,323</td>
<td>12,332</td>
</tr>
<tr>
<td>RLSD (proposed)</td>
<td>19,517</td>
<td>61,822</td>
<td>16,154</td>
<td>53,344</td>
<td>14,958</td>
<td>50,681</td>
<td>14,388</td>
<td>49,393</td>
<td>14,332</td>
</tr>
</tbody>
</table>

Figures 2 and 3 show the loss values obtained by both models in response time and throughput respectively. The lesser the loss value, the more accurate the simulation (unless the model was over fitted to the training set). The loss value can be computed during training and testing and is used to determine how good the model performs for the two different sets. Loss, like accuracy, is also not a proportion. It is the summarization of the errors incurred in training and test sets with each sample. It is clearly visible that the proposed RLSD model performed better than the existing methods by reducing the loss values in both response time and throughput.

V. CONCLUSION

A recommendation system is the backbone for retrieving and processing information that aims to forecast user interests. This study offers a novel deep learning-based approach called RLSD to provide recommendation services to predict QoS which incorporates a similarity AC to improve the prediction values. In the future, we plan to analyze and solve additional privacy concerns in location service recommendations, as well as improve the performance, using a parallel and distributed recommendations system to manage the required massive data. This model may also improve the results of [15, 23–29].

REFERENCES


