EP-LSTM: Novel prediction algorithm for moving object destination

Bingrong Li | Dechang Pi | Mengru Hou

Abstract
Predicting the destination of a moving object is a popular research subject in location-based services. By predicting destinations, suggestions can be offered to people regarding their trips. At present, there are problems such as data sparsity and long-term dependence based on historical trajectory prediction methods, which affect the accuracy of prediction. To solve data sparsity problem, this paper has devised an improved minimum description length method, which incorporates weighting parameters and optimizes the partitioning of trajectories with undirected complete graph. Long–short-term memory is a trajectory-prediction model that solves the problem of long-term dependence, but the model tends to have vanishing gradient issues when used to process longer sequences. This is because the hidden layers of long–short-term memory is largely affected by the lengths of sequences. Using embedded technology, the authors convert trajectory sequences into embedded vector sequences, and thus propose a deep-learning prediction model, EP-LSTM (Embedded Processing - Long Short Term Memory), which integrates embedded technology and long–short-term memory. The authors have conducted a great amount of testing with real data sets, comparing EP-LSTM with currently available predicting methods. The results have shown that EP-LSTM not only effectively solves data sparsity and long-term dependence but also achieves a high degree of prediction accuracy.

1 | INTRODUCTION

With the development of location-based services (LBS), many moving positioning devices have emerged, which can provide users with precise moving trajectories using global positioning systems (GPS, Beidou etc.). More importantly, by studying the trajectory data, we find that human social activities have certain regularity and that people always arrive at certain places repeatedly [1]. For instance, everyone has their own behavior habits. They are used to going to work in a certain company, shopping in a certain mall, eating in a certain restaurant, and even choosing the transportation way when they travel to and from here is nearly the same. Everyone has their own ‘circle’ and they visit friends and relatives from time to time. The purpose of destination prediction of the trajectory is to predict the destination before the moving object sets off, namely, the location information of the moving object at a certain time in the future. Now the coordinate information is no longer a simple longitude, latitude or plane horizontal and vertical coordinate values. Among them, the trajectory implies the action rules and hobbies of massive moving objects. If we get the location information of the moving objects, we can push relevant services to them timely and accurately, such as pushing the information of nearby delicious food and discount in the shopping mall to the user. Destination prediction is an essential task for many emerging location-based applications, including intelligent navigation [2] and detecting road congestion to conduct traffic planning. In terms of people’s lives [3], the urban planning manager can understand the pattern of residents’ activities for urban planning. In terms of disease transmission, the destination prediction based on moving objects can control the movement of residents. In terms of mining hot spots for business analysis [4], we can make advertising more accurate and efficient.

The destination prediction of moving objects is an important part in the field of trajectory prediction. In the recent years, many scientists have studied the trajectory prediction of...
moving objects, and made some achievements in this field. Trajectory prediction models can be mainly divided into the following three types.

The first type is prediction methods based on sequence analysis. In [5], Morzy adopted the improved algorithm to generate association rules. In [6], they constructed a T-pattern decision tree through the movement pattern in trajectories, and found the best matching path from the tree to predict the next position of the trajectory. However, the method has a significant drawback, as it tends to ignore the data sparsity. The obtained trajectories of the moving object are far from covering all the queries of trajectories in the data set, which is called the data sparsity problem. Hence, the model will lose the predictive ability. At present, the methods to solve the data sparsity problem are gridding [7], clustering [8], and similarity matching [9, 10]. But they all have some shortcomings. For example, the gridding method will lose some important trajectory information, while the similarity matching method will reduce the matching rate in the case of less historical trajectory information, resulting in inaccurate prediction results.

The second type is prediction based on Markov statistical learning. In [11], a simple Markov model is used for position prediction. It only considers the influence of current position on the future position, resulting in low prediction accuracy. In [12], the hybrid Markov model is used to predict the moving path of pedestrians, which reduces the space cost of the second-order Markov model and effectively predicts the future position of pedestrians. In [13], Yang et al. proposed an algorithm based on DBSCAN to obtain the residence point, and reached the prediction of the next position of the moving object by using the variable order Markov model. However, the trajectory destination prediction algorithm based on Markov only considers two to three GPS points in front of the sequence, which is not suitable for the trajectory with long dependence.

The third type is prediction based on deep neural network (DNN). In [14], Lv et al. designed T-CONV model to predict the destination of moving objects by using multi-level convolution neural network to extract multi-scale two-dimensional trajectory features. In [15], Rossi et al. used geographic information in a location-based social network and encoded the location semantics of the visit, and then established a recursive neural network model to predict the next destination. However, these neural network models are not good at memorizing long trajectory sequences, resulting in unsatisfactory prediction. In particular, the recurrent neural network (RNN) [16], one of the deep learning networks, which has the problems of gradient disappearance and gradient explosion, is always facing the challenge of long-term dependence problem. When the number of trajectory points that the trajectory prediction destination depends on increases, and the relevant point of trajectory prediction is too far from the output time, the occasion is called the long-term dependence problem. Long–short-term memory (LSTM) [17] is very popular for trajectory prediction to solve the long-term dependence. However, the LSTM model is not effective in dealing with long sequences. Its hidden layer is greatly affected by the length of trajectory data sequence, which requires the input of low-dimensional vector, otherwise it will lead to serious dimension disaster and increase time overhead, so as to reduce the learning efficiency of the model. Therefore, it is still facing great challenge to effectively predict the destination just based on historical trajectories.

To fully address the abovementioned issues, this paper proposes a deep learning prediction model named EP-LSTM (Embedded Processing - Long Short Term Memory), which is validated on real data sets. It can not only effectively solve the data sparsity and long-term dependence, but also has better prediction result. Note that, this paper focuses on the destination prediction of moving object, but the present work can also be applied to location prediction in other fields, such as prediction of typhoon landing point and prediction of epidemic virus transmission.

In summary, we make the following contributions in this paper.

1. We put forward an improved minimum description length (IMDL), which is used to transform the trajectory segmentation into the shortest path for solving the undirected complete graph, so as to obtain the best segmentation of the trajectory. As a result, this method effectively overcomes the data sparsity problem in destination prediction.
2. We propose a deep learning prediction model named EP-LSTM. First, in order to reduce the dimension of the input sequence, the input sequence is transformed into embedded vectors using embedded technology. Furthermore, the feature vectors are extracted and used as the input for LSTM, which can effectively alleviate the long-term dependence problem. Consequently, the EP-LSTM algorithm can effectively solve the long-term dependence without neglecting the movement law of moving objects.
3. We conduct extensive experiments with the real trajectory data set of Kaggle-ECML/PKDD competition [18]. The prediction accuracy and distance error are used to evaluate the effectiveness and efficiency of the algorithm. A large number of experimental results with the real trajectory data show that the accuracy of our algorithm is 82.4%, far beyond the current existing algorithms.

The remainder of this paper is organized as follows: Section 2 offers the problem statement and the methods analysis. Section 3 presents the EP-LSTM algorithm in detail. Section 4 reports experimental results. Finally, Section 5 concludes the paper.

## 2 | DESTINATION PREDICTION ALGORITHM EP-LSTM

For destination prediction of moving objects, the common method is to build a model based on historical trajectories. If there exists a history trajectory that includes the part of current trajectory, the terminal point of the history trajectory is regarded as the destination of current trajectory. However, this method does not take into account the limited number of trajectories in the data set and the limited amount of information, and
destination prediction is also affected by other objective factors, such as data sparsity problem. Moreover, the existing prediction methods based on Markov chain and RNN’s DNN cannot solve the long-term dependence problem. Markov chain only depends on the few GPS points, and historical data has a certain timeliness. RNN has the problem of gradient disappearance, which does not solve the long-term dependence.

In order to address the problems of data sparsity and long-term dependence, the trajectory destination is predicted accurately while maintaining trajectory information. A novel method named EP-LSTM algorithm is proposed to predict the destination. Here, Figure 1 shows the flow chart of the algorithm training, and Figure 2 shows the flow chart of applying the algorithm to prediction. First, the historical trajectory data are segmented, and the IMDL method is used to process the original trajectory data. While retaining the original trajectory characteristics, each trajectory data is segmented to solve the data sparsity, as detailed in Section 2.2. Then, an embedded technology is presented, which extracts the feature vector and converts the trajectory sequence into the embedded vector sequence as the input of the model. LSTM model was trained on data set. The problem of long-term data dependence has been effectively solved, which largely makes up for the deficiency of the RNN model. Finally, we put the processed training data into the LSTM model for training and select the best training output prediction model.

In Figure 2, the test trajectory data is preprocessed first, and the trajectory sequence is converted into an embedded vector by embedded technology. Finally, the trajectory destination is predicted by inputting the sequence vectors into EP-LSTM algorithms as detailed in Section 2.3.

2.1 Basic definitions and problem statements

Definition 1. (Trajectory sequence):
A trajectory is a collection of GPS trajectory points generated by a moving object within a certain period of time. $O$ is a collection of moving objects, represented by $\{O_1, O_2, \ldots, O_i\}.$
trajectory of moving object is represented by \( T_j = (p_1, p_2, \ldots, p_t) \), where \( \lambda_i \) represents that the location of moving object \( O_i \) at time \( t_i \), \( \lambda_i \), and \( \varphi_i \) correspond to the latitude and longitude of \( p_i \), respectively. \( T_{ij} = (p_1, p_2, \ldots, p_t) \) is defined as a part of the moving object trajectory from the beginning to the current point, \( i \leq n \). \( V_i = \{V_1, V_2, \ldots, V_{i-1}\} \) represents the vector mapped by the trajectory.

**Definition 2.** (Destination prediction of moving objects): Assume that \( T \) is a historical trajectory composed of \( \lambda \) historical position sequences, where \( T \) can be shown as \( (p_1, p_2, \ldots, p_k) \). The current trajectory is \( T_i = \{\lambda_i, \varphi_i, t_i\}, 0 \leq i \leq t \). When \( i \) is equal to \( s \), the position of the moving object at time \( i \) is expressed as \( T_i \).

### 2.2 Trajectory data pre-processing

Before inferring the trajectories of moving objects, we have to preprocess trajectory data. In this section, we introduce a well-designed pretreatment method to achieve accurate prediction of destination. In order to make the trajectory more concise and convenient, it is necessary to segment the trajectory. Minimum description length (MDL) has been proved to be one of the effective trajectory segmentation methods. For trajectory segmentation, first, the feature points are found in a trajectory, and the adjacent feature points are connected to form a new trajectory. This paper presents an IMDL approach to simplify the trajectory.

According to the definition of reference [19], first, the distance between different trajectory segments is defined, and then the trajectory segmentation problem is formally described. The distance between lines consists of three parts: vertical distance, parallel distance, and angular distance. The trajectory can be represented as a line segment in three-dimensional space, \( T \) is a trajectory sequence, and \( T = (p_1, p_2, \ldots, p_k) \), where \( p_i \) representing the position of the trajectory \( T \) at time \( t_i \) is \( (\lambda_i, \varphi_i) \), \( 1 \leq i \leq n \). Figure 3 shows the definitions of three line segment distances.

In Figure 3, any two trajectory segments are represented by line segments \( L_i \) and \( L_j \). The vertical distance, parallel distance, and angular distance between \( L_i \) and \( L_j \) are shown in Equation (1). \( L_{1j} \) denotes the Euclidean distance between point \( S_j \) and its vertical mapping point \( p_j \) on line segment \( L_i \), and \( L_{12} \) denotes the Euclidean distance between \( \varepsilon_j \) and \( p_i \). 

\[
\begin{align*}
\text{denotes the length of line segment. } \\
\theta(0^\circ \leq \theta \leq 180^\circ) \text{ is the smaller angle between the line segments } L_i \text{ and } L_j.
\end{align*}
\]

declares the length of line segment. \( \theta(0^\circ \leq \theta \leq 180^\circ) \) is the smaller angle between the line segments \( L_i \) and \( L_j \).

\[
d_1 = \frac{l_{1j}^2 + l_{1j}^2}{l_{1j}^2 + l_{1j}^2}
\]

\[
d_j = \text{MIN}(\|l_1\|, \|l_2\|)
\]

\[
d_\theta = \|L_j\| \times \sin(\theta)
\]

According to the above definition, the distance between the lines is expressed in Equation (2).

\[
dist(L_j, L_j) = \omega_1 \cdot d_1(L_j, L_j) + \omega_2 \cdot d_j(L_j, L_j) + \omega_3 \cdot d_\theta(L_j, L_j)
\]

where \( \omega_1, \omega_2, \) and \( \omega_3 \) are the weights corresponding to the vertical distance, the parallel distance, and the angular distance, and they are usually set as 1.

The MDL consists of two parts: \( L(H) \) and \( L(D(H)) \). Here, \( H \) means hypothesis, and \( D \) means data. \( L(H) \) is used to describe the hypothetical length, and \( L(D(H)) \) is the code length obtained by encoding data \( D \) with the help of the hypothesis of \( H \). When the sum of the two is minimal, \( H \) is the best hypothesis to explain data \( D \).

There is a disadvantage in this method. It can only find the approximate trajectory segmentation that makes \( A \) smaller. However, in actual applications, there are certain requirements for the accuracy and simplicity of the trajectory. In this paper, first, an IMDL method is presented, which makes the accuracy and simplicity of the trajectory meet the requirements [20]. We find a segment to minimize \( mL(H) + nL(D(H)) \) with \( mL(H) + nL(D(H)) \) instead of \( L(H) + L(D(H)) \). Here, \( m, n \geq 0 \). At the same time, the size of \( m \) and \( n \) is adjusted according to the specific requirement to get the best segment. \( L(H) \) and \( L(D(H)) \) are shown in Equation (3):

\[
L(H) = \sum_{j=1}^{1\leq j \leq 1} \log_2\left(\text{len}(p_j, p_{j+1}) + 1\right)
\]

\[
L(D(H)) = \sum_{j=1}^{1\leq j \leq 1} \sum_{k=1}^{\text{len}(p_k, p_{k+1}) + 1} \left\{\log_2\left(\|d_1(p_j, p_{j+1}, p_k, p_{k+1}) + 1\right)\right\}
\]

Hence, \( L(D(H)) \) denotes the sum of the length of all trajectory partitions, and \( L(D(H)) \) denotes the sum of the difference between a trajectory and a set of its trajectory partitions. Then, all the points in a trajectory are converted into an undirected complete graph, namely, the trajectory segmentation problem is transformed into the shortest path problem of the undirected complete graph, where the weight \( \omega(p, p_j) \) of edge \( p, p_j \) is calculated by \( mL(H) + nL(D(H)) \). Finally, the best segmentation can be obtained using the calculation weight method between any two nodes in the undirected complete graph.
Feature vector extraction and embedded processing

In Section 2.1, through the trajectory segmentation algorithm of IMDL, a collection of segmented trajectory data can be obtained. At the same time, based on this trajectory segmentation method, the data sparsity problem can be effectively overcome. In this section, we will process the sequence after the trajectory segmentation. In order to extract the feature vector, we need to convert the position coordinate points in the segmented trajectory sequence into a grid chain, modify the time-stamp data format, and, finally, divide the data set.
One-hot encoding is an approach to extract text features in natural language processing. But the one-hot encoding algorithm has the following two defects.

1. When the amount of data is large, it will face the problem of too long vector length and too little effective information.
2. It cannot show the movement pattern of moving objects, and the relationship between geographical location cannot be reflected.

Although we have simplified the trajectory, the dimension of the encoded trajectory sequence vector is still relatively high for the neural network model. In view of the defects existing in one-hot encoding, we use embedded technology to solve the problem of discrete input values. Namely, the input trajectory sequence is mapped to an embedded vector. Based on this idea, we propose a novel prediction algorithm named EP-LSTM, which is a deep learning prediction model based on embedded technology combined with LSTM.

Before introducing the EP-LSTM algorithm, first, we introduce the embedded technology. It is called word2vec technology which is often used in natural language processing field. Word2vec can better express the similarity and analogy between different words. Essentially, it is a kind of shallow neural network. Word2vec is divided into two large frameworks in terms of realization: negative sampling and hierarchical software max.

In the present study, selected hierarchical software max is selected to realize word2vec, which consists mainly of two models: continuous bag-of-words model (skip-gram) and continuous skip-gram model (CBOW). CBOW model predicts context based on target words, Skip-gram model predicts target words based on context. Because the destination prediction of the trajectory is based on the historical trajectory data, CBOW is a suitable model to implement hierarchical software max.

The CBOW model consists of three layers: input layer, projection layer, and output layer. It is a three-layer neural network. Compared with DNNs, it is not necessary to calculate all the non-leaf nodes, and the node parameters on the path are updated by the partial derivative each time. The efficiency of word vector training is improved. The model is equivalent to the vector of a word bag model multiplied by an embedding matrix, resulting in a continuous embedding vector. Its framework is shown in Figure 5.

As shown in Figure 5, the mapping idea between layers in the CBOW model is as follows.

1. Input layer: the one-hot encoding method is used to form a low-dimensional feature vector \( X \), and then the feature vector is updated by linear transformation. Here, \( W \) is a dictionary composed of all words. The linear transformation is as shown in Equation (4).

\[
\mathbf{y}^{\langle k \rangle} = W X^{\langle k \rangle}
\]

2. Projection layer: the input matrix is mapped to a vector by summing and averaging the corresponding dimensions, as shown in Equation (5).

\[
\mathbf{y}^{\prime} = \frac{\mathbf{y}^{\langle k \rangle} + \mathbf{y}^{\langle k+1 \rangle} + \cdots + \mathbf{y}^{\langle m \rangle}}{2^m}
\]

3. Output layer: the occurrence frequency of each word is taken as the weight, and then the Huffman tree is constructed and output according to the weight and model parameters of each category, as shown in Equation (6).

\[
\zeta = U y^{\prime} \\
\gamma = \text{softmax}(\zeta)
\]

With regard to the calculation of gradient, the CBOW model uses the random gradient escalation method to update each parameter. Therefore, the maximum likelihood function is used to solve the parameter on the binary tree, namely, the vector on the non-leaf node. Hierarchical softmax constructs a classified binary tree using Huffman encoding. It reduces the computational complexity. When \( N \) training samples are trained, the number of occurrences of each category corresponds to the weight of the Huffman tree. For samples with more occurrences, the path is shorter, and for samples with fewer occurrences, the path is longer.

The main idea of CBOW is to maximize the probability of \( p(w|\text{Context}(w)) \). With the training of the neural network, the parameters are constantly updated until a suitable weight vector is found, so that the conditional probability \( p(w|\text{Context}(w)) \) is the largest. The result of multiplying the weight vector by the input matrix is the word vector corresponding to the input value, which prepares for building the LSTM model.

The objective function based on the neural network model usually uses a logarithmic likelihood method, as shown in Equation (7).

\[
L = \sum_{w \in \text{occ}} \log p(w|\text{Context}(w))
\]

The key is to construct the conditional probability \( p(w|\text{Context}(w)) \). The essence of hierarchical softmax is to change the \( N \) classification problem into a binary classification problem. The idea of training for CBOW is to decompose the complex normalized probability into a series of conditional
The unit structure of LSTM is shown in Figure 6. \(X_t\) denotes the input value at time \(t\); \(h_{t-1}\) denotes the output data at time \(t\); \(b_{t-1}\) denotes the output value of the previous cell state; \(i\) and \(O\) are the input layer, the forgetting layer, and the output layer; and \(C_{t-1}\) is the state value of LSTM unit at time \(t - 1\); \(C_t\) is the state value of LSTM unit at time \(t\); \(\sigma\) represents the sigmoid activation function, which converts digital signals to a value between 0 and 1. Each tanh layer generates a vector for updating the input content. \(\oplus\) denotes the addition of the matrix. \(\otimes\) means to multiply the corresponding elements in the operational matrix. The unit structure of LSTM is shown in Figure 6.

There are three main stages within LSTM:

1. Forget the stage: forget gate \(f_t\). The function of the forget gate is to forget information. In the use of LSTM, this stage is made up of sigmoid neural network layer and multiplication by bit. \(C_{t-1}\) point multiplication with \(f_t\) means discarding the information to be forgotten, and \(i_t\) point multiplication with \(C_{t-1}\) means picking out the information to be updated from the candidate value vector (as shown in Equation (9)).

2. Selective memory stage: input gate \(i_t\). The function of the input gate is to selectively memorize. The stage consists of input gate with neural network layer and multiplication by bit. First, the input gate is used to calculate the state to be input, represented by \(i_t\), where the sigmoid function determines the part that needs to be updated. And a new candidate variable \(C'_t\) is created by tanh layer, and the value range of the function is between \((-1, 1)\). As shown in Equation (10), it can select the information to update by the input gate point multiplication with the candidate value variable \(C'_t\).

3. Output stage: output gate \(O_t\). The function of the output gate is to decide which will be used as the output of the current state. At this stage, the output gate, tanh, neural network layer, and multiplication by bit operate together to transfer the cell state and input signal to the output. Namely, input \(C_t\) into the tanh function and compress all parameter values between \(-1\) and 1, as shown in Equation (11).

Here, let \(X_t\) be the input time sequence at the current time \(t\), through the above gating structure, the cell state \(c\) is given by the following equations:

\[
f_t = \sigma(\mathbf{W}_f \cdot [h_{t-1}, x_t] + b_f) \tag{9}
\]

\[
i_t = \sigma(\mathbf{W}_i \cdot [h_{t-1}, x_t] + b_i) \tag{10}
\]

\[
\bar{C}_t = \tanh\left(\mathbf{W}_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{11}
\]

\[
f_t \cdot C_{t-1} + i_t \cdot \bar{C}_t
\]

\[
O_t = \sigma(\mathbf{W}_o \cdot [h_{t-1}, x_t] + b_o) \tag{9}
\]

\[
b_t = O_t \cdot \tanh(C_t)
\]

LSTM is a special RNN whose core idea is to control the transmission state through gated state, remember the information that needs to be remembered for a long time, and forget the unimportant information, mainly to solve the problem of gradient disappearance and gradient explosion during long sequence training. Next, starting from the core part of LSTM, the training process of destination prediction algorithm is described in detail, as shown in Figure 7.

In order to simplify the model, only one hidden layer was drawn, and the original trajectory data was preprocessed in IMDL method to obtain the collection of historical trajectory sequences. A trajectory sequence composed of multiple positions can be obtained by adopting embedded technology and represented by \(X = \{x_1, x_2, x_3, \ldots, x_n\}\). By randomly selecting a certain number of sample data from \(X\) and setting \(p\) positions after the number of samples as label \(y\), the \(X\) and \(y\) are used as the input of the model. It is transmitted from the input layer to the LSTM neural network linearly connected by \(N\) hidden
layers, and the output vector sequence \( b = \{b_1, b_2, \ldots, b_n\} \) of \( n \) hidden layers is obtained through continuous iterative calculation. Then, the output vector is transported to the output layer. If the number of samples is set as \( k \), the trained vector sequence \( y = \{y_{k+1}, y_{k+2}, \ldots, y_{k+p}\} \) can be obtained through the calculation of the activation function and the backpropagation mechanism. Among them, the activation function of the hidden layer from time \( t = 1 \) to time \( t = T \) is continuously iterated by Equation (12):

\[
b_y = \tanh(C_y) \ast \sigma \left( W_h [b_{y-1}, x_t] + b_h \right)
\]

where \( C_y \) denotes the state value of the memory unit at the current moment, \( W \) denotes the weight between the input layer and the hidden layer, \( b \) denotes the bias vector, and \( \sigma \) denotes the activation function of the hidden layer.

When the vector sequence passes to the output layer, the output result is the position corresponding to each position of embedded vector sequence. Therefore, the embedded position vector of the output can be obtained by putting the trajectory sequence to be predicted into the trained model. Finally, the Euclidean distance is calculated between the output sequence and all the vectors in the input sequence, and the embedded position vector with the smallest distance is the prediction result.

In the field of trajectory prediction of moving objects, LSTM is one of the important models for solving long-term dependence problems. However, the hidden layer of LSTM is greatly affected by the length of data sequence. We use the embedded technology introduced in Section 2.3 to transform high-dimensional input vectors into word vectors, which also transforms trajectory sequences into distributed vector sequences to reconstruct LSTM model. In this paper, the prediction algorithm EP-LSTM is proposed, which combines the embedded technology with LSTM to overcome the gradient disappearance of the basic LSTM model to deal with the long trajectory sequences and improve the accuracy of destination prediction, as depicted in Algorithm 3.

The basic idea of the above algorithm EP-LSTM is as follows. First, the training parameters are initialized (Line 01), and then the graph structure of the neural network is constructed (Lines 02–11). Here, a multi-layer LSTM structure is constructed to process the embedded position vector (Lines 02–06). In addition, we use embedded technology based on word2vec principle to convert trajectory sequence to word vectors, and the corresponding elements of the index in the trajectory sequence are selected as input of the model. Then, the preprocessed data are trained in the model and the loss function is calculated. Through a certain number of iterations, the errors corresponding to different iterations can be calculated (Lines 07–11). Finally, the best training model is selected and output (Lines 12–13).

The destination prediction algorithm based on EP-LSTM is given below, as depicted in Algorithm 4.
model of Algorithm 3 to input the embedding position vector to be predicted into the model for prediction (Lines 04–05). Next, by traversing each trajectory sequence, calculating the Euclidean distance, the embedded vector closest to the real destination is found within the limited threshold range. The position of the vector after mapping is the most likely location in the destination prediction (Lines 06–09). Finally, the best training model is selected and output (Line 10).

4 | EXPERIMENTS AND ANALYSIS

4.1 | Data set and test environment

In our study, we focus on the Porto data set; it was released in the context of the ECML/PKDD 2015 challenge and hosted as a Kaggle competition [18], which allows us to easily compare our model with the winner’s approach. It provides data of 442 taxis in Porto for one year. Sample data from the experimental data set is shown in Table 1. Table A.1 gives a detailed description of each feature name, and Table A.1 is given in Appendix 1.

All algorithms were implemented on Anaconda3 IDE, using the Python programming language. The hardware environment included an Intel(R) Xeon(R) E5-2620 v3 2.40 GHz CPU, with 64GB RAM.

4.2 | Performance evaluation

Definition 1. (Distance error calculation):

This paper used Haversine distance to calculate the distance deviation between the real destination and the predicted destination, and the deviation is used as the standard to measure the prediction model. It measures the distance between two points on the sphere according to longitude and latitude, as shown in Equation (13).

\[
\alpha = \sin^2 \left( \frac{\lambda_1 - \lambda_2}{2} \right) + \cos(\lambda_1) \cos(\lambda_2) \sin^2 \left( \frac{\varphi_1 - \varphi_2}{2} \right)
\]

\[
d_{\text{haversine}} = 2R \arctan \left( \sqrt{\frac{\alpha}{1 - \alpha}} \right)
\]

where \( \lambda_1 \) and \( \lambda_2 \) are the latitudes of the two coordinate points, and \( \varphi_1 \) and \( \varphi_2 \) are the longitudes of the two coordinate points.

d_{\text{haversine}} denotes the Haversine distance, and \( R \) is the radius of the earth, set as 6371 km.

Definition 2. (Prediction accuracy):

Trajectory sequence is \( T = (\varphi_1, \varphi_2, \ldots, \varphi_n) \), and the predicted trajectory sequence can be expressed as \( T_p = \{ T_{p_1}, T_{p_2}, \ldots, T_{p_n} \} \).

The definition of prediction accuracy is shown in Equation (14).

\[
\text{Accuract} = \frac{\sum_{i=1}^{n} H(T_i, T_{p_i})}{|T|}
\]

\[
H(T_i, T_{p_i}) = \begin{cases} 
1, & (d(T_i, T_{p_i}) < \sigma) \\
0, & (d(T_i, T_{p_i}) > \sigma)
\end{cases}
\]

where \( T_{p_i} \) represents the predicted result of the \( i \) sample, \( T_i \) is the real result, \( d(p, q) \) represents the Euclidean distance between \( p \) and \( q \) in space, and \( \sigma \) represents the distance threshold, such as setting it to 1 km. When \( d(T_i, T_{p_i}) < \sigma \), it means that the prediction is relatively accurate within the threshold range, and the predicted destination is similar to the real destination.

4.3 | Model training and test results analysis

In order to evaluate the effectiveness of the proposed algorithm, this section uses the distance error criterion in Section 4.2 to quantify the performance of the algorithm. Based on the observation that the local portions near the start and the end of a trajectory play more significant roles to the destination prediction, we do not take the complete trajectory length, but intercept the trajectory information near the starting point and the end point according to the general partition ratio (such as 30%) as input. In the training phase, with the increase of the number of iterations, the error decreases gradually, and after a certain number of iterations, the training error tends to be stable, and the model training is over. The results is given in Table 2.

As shown in Table 2, we mainly compare the EP-LSTM with the neural networks-based models [21], which win the champion of the competition. This table shows the sum of the errors of all training set trajectory data. It is calculated based on the distance measurement Equation (13). Model 1 is the champion model of the ECML-PKDD competition, which
TABLE 2  Comparison of prediction errors with different models

| Model                                      | Test error |
|--------------------------------------------|------------|
| 1. MLP, clustering (winning model)         | 2.81       |
| 2. MLP, direct output                      | 2.97       |
| 3. MLP, clustering, no embeddings          | 2.93       |
| 4. MLP, clustering, embeddings, only       | 4.29       |
| 5. RNN                                     | 3.14       |
| 6. Bidirectional RNN                       | 3.01       |
| 7. Bidirectional RNN with window            | 2.60       |
| 8. EP-LSTM                                 | 2.36       |

TABLE 3  Display table of model parameters

| Model parameters   | Value |
|--------------------|-------|
| learning_rate_base | 0.001 |
| learning_rate_decay| 0.99  |
| batch_size         | 200   |
| vocabulary_size    | 5000  |
| embedding_size     | 128   |
| num_hidden         | 128   |
| context_window     | 6     |
| input_size         | 10    |
| num_sampled        | 64    |

predicts the travel destination of moving objects through multi-layer perceptron (MLP) and meanshift clustering. Models 2, 3, and 4 are improved models based on MLP, which predict destinations by embedding other attributes or performing internal conversion processing. Model 5 reads the GPS points in the trajectory by using RNN, and predicts the trajectory destination of the moving object directly through the last internal state. Models 6 and 7 are predicted using a bidirectional RNN model, which is a variant of RNN. It is composed of two RNN superimposed together. The output is determined by the state of the two RNN. From the test error results, we can see that the EP-LSTM algorithm proposed in the present study has a great improvement compared with the above seven models.

In the task of training model, we used the deep learning library tensor-flow to build the LSTM model, and choose the optional parameters for the EP-LSTM algorithm after many experiments. To train the network, we used the adagrad optimize to obtain the final prediction model. The adagrad optimize is an adaptive gradient algorithm, which can automatically adjust the learning rate during training. The network is trained using the mean squared error (MSE) as the loss function and implementing the early stopping. During the test phase, we use the parameters of the LSTM that produced the best MSE score on the validation set. The parameters of the EP-LSTM algorithm are given in Table 3.

As shown in Table 3, learning_rate_base denotes the basic learning rate, learning_rate_decay denotes the attenuation rate, vocabulary_size represents the dimension of input data converted into word vector, embedding_size denotes the dimension of embedding vectors, and num_hidden represents the number of neurons in the hidden layer. In addition, we choose the CBOW model with a window size of 6 and without filtering words on frequency context_window, input_size denotes the length of the input data, and num_sampled denotes the size of the data input into the loss function.

Next, we conduct experiments to train EP-LSTM models. In order to test the prediction accuracy of the EP-LSTM algorithm, for this set of experiments, we use the same division way as in reference [21], by randomly selecting 19,427 trajectories from the original set as the verification set, 19,770 trajectories as the test set, and the rest as the training set. After the training of the model is completed, the validated data set is used to verify the model. The result is shown in Figure 8.

As shown in Figure 8, the vertical axis represents the error value, and the horizontal axis represents the number of iterations. The solid and dashed lines, respectively, represent the training error and verification error. The smaller the error value, the better the training model. As can be seen from Figure 8, as the number of iterations increases, the error value in the validation data set is always smaller than that in the training data set, namely, the prediction effect is better, which avoids the over-fitting in the training process. We can also find that when the number of iterations reaches 2000, the difference in the error value between the validation data set and the training data set is the largest. Moreover, when the number of iterations is 12,000, the error value tends to be stable, reaching about 0.31. As a consequence, the trajectory segmentation algorithm of IMDL effectively solves the data sparsity problem.
4.4 Comparison and analysis

In this section, we conduct experiments to evaluate the effectiveness of the EP-LSTM algorithm, and compare it with the existing algorithm. At present, there are some related algorithms for trajectory destination prediction. The most common trajectory matching method is called Baseline algorithm [22], which matches the query trajectory with the historical trajectory. However, the algorithm has a great drawback. When there is no ‘matching’ trajectory in the historical trajectory, it is difficult to meet the need of trajectory prediction in real life. There is also Markov prediction algorithm [23], which established the transition matrix by calculating the conversion probability between user locations, and then infers the most likely location of the destination. In addition, Xue et al. proposed a SubSyn algorithm [7], which decomposes the historical trajectory into sub-trajectories consisting of two adjacent positions, and the sub-trajectories are spliced into a complete trajectory for trajectory prediction.

In order to prove that the EP-LSTM algorithm effectively solves the problem of low hit ratio, we use the prediction accuracy to quantify the performance of the model. Therefore, we report the prediction accuracy and prediction time between the EP-LSTM algorithm and the existing algorithms, including SubSyn, Baseline, and 2-Markov (second-order Markov chain). Comprehensive experiments based on real trajectory data show that EP-LSTM can achieve higher accuracy than the existing algorithms. The results are shown in Figures 9 and 10.

As shown in Figure 9, the vertical axis represents the prediction accuracy of four algorithms EP-LSTM, SubSyn, Baseline, and 2-Markov, and the horizontal axis represents the number of trajectories of moving objects. As can be seen from Figure 9, with the increase of the number of trajectories in the training set, the accuracy of all the algorithm has an apparent improvement. However, compared with the other three algorithms, the prediction accuracy of EP-LSTM algorithm has been significantly improved. Based on the experimental results, we find that the number of trajectories of moving objects reaches 12,000, and the prediction accuracy reaches 82.4%, which shows the effectiveness of the destination prediction algorithm proposed in the present study. At the same time, it also reflects that the methodology of introducing trajectory segmentation and embedded processing is desirable.

As we know, the time complexity of the algorithm can reflect the efficiency of the algorithm to a great extent. The lower the time complexity of the algorithm, the higher the efficiency of the algorithm. According to the definition of time complexity, it refers to the time spent in running the algorithm, excluding training time and verification time. Therefore, in order to evaluate the operation efficiency of EP-LSTM, we adopt prediction time as evaluation metric. The experimental data are 320 trajectory data of test.csv, and batch data is bach_size which is 5. The result is shown in Figure 10.

As shown in Figure 10, the vertical axis represents the prediction time of each batch_size, and the horizontal axis represents the four algorithms. According to the experimental results, we can see that the Baseline algorithm is the most time-consuming, because the entire experimental data set needs to be traversed when matching the query trajectory with the historical trajectory. 2-Markov and Subsyn have relatively little prediction time, because the time and space complexity of the sample data during processing is lower. The prediction time of EP-LSTM algorithm is lower than that of Baseline algorithm, and is higher than that of Subsyn and Markov. This is explained by the fact that the training of the neural network requires a lot of matrix operations and the time complexity is relatively high. From a comprehensive point of view, the prediction effect of EP-LSTM algorithm is better. Without neglecting the movement law of moving objects, the data sparsity problem of destination prediction is effectively addressed.

5 CONCLUSION AND FUTURE WORK

Destination prediction of moving objects is an important part of LBS, which has attracted more and more attention due to its broad applications, such as traffic navigation, virus transmission, hotel recommendation, and personalized advertising.
However, there are always trajectory data sparsity and long-term dependence in this area.

To address the above problems, we simplify the trajectory by introducing the method of trajectory segmentation IMDL. Then, the embedded vector is used to extract the feature vector. Finally, the data set is trained in the LSTM model to establish the EP-LSTM algorithm, which is used to reach destination prediction. While ensuring the availability of data, we improve the accuracy of destination prediction and effectively solve the adverse effects caused by data sparsity and long-term dependence. We report on a series of experiments with a real-world location history data set from the ECML/PKDD Discovery Challenge 2015 data set, showing that the EP-LSTM algorithm can achieve better prediction accuracy than the existing algorithms.

In terms of our future work, we plan to improve the time performance of the EP-LSTM algorithm. In addition, this algorithm does not consider the objective factors such as speed, road condition, and weather of moving objects. Therefore, we will analyze the factors mentioned above in the future work to improve the prediction accuracy of EP-LSTM.

CONFLICT OF INTEREST
The authors declare that there is no conflict of interest with any person(s) or organization(s).

REFERENCES
1. Li, C., Cheung, W.K.: Recovering human mobility flow models and daily routine patterns in a smart environment. In: IEEE International Conference on Data Mining Workshop (ICDMW), IEEE Computer Society, Shenzhen, China, 1, pp. 541–548 (2014)
2. Natarajan, S., et al.: Traffic sign recognition using weighted multi-convolutional neural network. IET Intel. Transport Syst. 12(10), 1396–1405 (2018)
3. Gonzalez, M.C., et al.: Understanding individual human mobility patterns. Nature 453(7196), 779–782 (2008)
4. Wang, C., et al.: Destination prediction based scheduling algorithms for message delivery in IoVs. IEEE Access 8, 14965–14976 (2020)
5. Motry, M.: Mining frequent trajectories of moving objects for location prediction. In: Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition, Leipzig, Jul 18–20, 2007. Berlin, Heidelberg: Springer, pp. 667–680 (2007)
6. Monreale, A., et al.: Wherenext: a location predictor on trajectory pattern mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, Paris, France, pp. 637–646 (2009)
7. Xue, A.Y., et al.: Solving the data sparsity problem in destination prediction. VLDB J, Springer Verlag, Germany, 24(2), 219–243 (2015)
8. Dai, C., et al.: Brain EEG time-series clustering using maximum-weight clique. IEEE Trans. Syst. Man Cybern. 99(online), 1–15 (2020)
9. Cai, L., et al.: Dual implicit mining-based latent friend recommendation. IEEE Trans. Syst. Man Cybern. 50(5), 1663–1678 (2020)
10. Yang, Z., et al.: An efficient destination prediction approach based on future trajectory prediction and transition matrix optimization. IEEE Trans. Knowl. Data Eng. 32(2), 203–217 (2020)
11. Chen, M., et al.: Nlpmm: a next location predictor with markov modeling. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer International Publishing, Charn, Switzerland, 8444(2), pp. 186–197 (2014)
12. Ashara, A., et al.: Pedestrian-movement prediction based on mixed Markov-chain model. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, Chicago, IL, United states, pp. 25–33 (2011)
13. Yang, J., et al.: Predicting next location using a variable order Markov model. In: Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoStreaming. ACM, Dallas, TX, United states, pp. 37–42 (2014)
14. Lv, J., et al.: T-CONV: a convolutional neural network for multi-scale taxi trajectory prediction. In: 2018 IEEE International Conference on Big Data and Smart Computing (bigcomp). IEEE, Shanghai, China, pp. 82–89 (2018)
15. Rossi, A., et al.: Modeling taxi drivers’ behaviour for the next destination prediction. IEEE Trans. Intell. Transp. Syst. 21(7), 2980–2989 (2020)
16. Mikolov, T., et al.: Rnnlm-recurrent neural network language modeling toolkit. In: Proceedings of the 2011 IEEE workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 196–201 (2011)
17. Pei, Z., et al.: Human trajectory prediction in crowded scene using social-affinity long short-term memory. Pattern Recognit. 93, 273–282 (2019)
18. Kaggle, Kaggle competition. https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i (2015)
19. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery (ACM), United states, pp. 593–604 (2007)
20. Zhang, H.F., Pi, D.C., Dong, Y.L.: IBTC: a clustering algorithm for moving object trajectories based on independent forests. Comput. Sci. 46(01), 251–259 (2019)
21. Brébisson, A.D., et al.: Artificial neural networks applied to taxi destination prediction. In: Proceedings of the 2015th International Conference on ECML/PKDD Discovery Challenges, 1526, pp. 40–51 (2015)
22. Besse, P.C., et al.: Destination prediction by trajectory distribution-based model. IEEE Trans. Intell. Transp. Syst. 19(8), 2470–2481 (2017)
23. Neto, F.D.N., de Souza Baptista, C., Campelo, C.E.C.: Combining Markov model and prediction by partial matching compression technique for route and destination prediction. Knowledge-Based Syst. 154, 81–92 (2018)

How to cite this article: Li B, Pi D, Hou M. EP-LSTM: Novel prediction algorithm for moving object destination. IET Intell Transp. Syst. 2021;15:235–247. https://doi.org/10.1049/itr2.12017
### APPENDIX A

**TABLE A.1**  
Information description of each figure name

| Figure name   | Description                                                                                                                                 |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| **TRIP_ID**   | It contains an unique identifier for each trip                                                                                               |
| **CALL_TYPE** | It identifies the way used to demand this service. It may contain one of three possible values:                                                  |
|               | ‘A’: if this trip was dispatched from the central;                                                                                           |
|               | ‘B’: if this trip was demanded directly to a taxi driver on a specific stand;                                                                  |
|               | ‘C’: otherwise (i.e. a trip demanded on a random street)                                                                                     |
| **ORIGIN_CALL** | if CALL_TYPE = ‘A’, ORIGIN_CALL will have a unique number for a unique customer's phone number. Otherwise, it assumes a NULL value             |
| **ORIGIN_STAND** | It contains an unique identifier for the taxi stand. If CALL_TYPE = ‘B’, it identifies the starting point of the trip. Otherwise, it assumes a NULL value |
| **TAXI_ID**   | It contains an unique identifier for the taxi driver that performed each trip                                                                  |
| **TIMESTAMP** | It identifies the trip's start                                                                                                               |
| **DAYTYPE**   | It identifies the daytype of the trip's start. It assumes one of three possible values:                                                      |
|               | ‘B’: if this trip started on a holiday or any other special day;                                                                            |
|               | ‘C’: if the trip started on a day before a type-B day;                                                                                     |
|               | ‘A’: otherwise (i.e. a normal day, workday, or weekend)                                                                                   |
| **MISSING_DATA** | It is FALSE when the GPS data stream is complete and TRUE whenever one (or more) locations are missing                                     |
| **POLYLINE**  | It contains a list of GPS coordinates (i.e. WGS84 format) mapped as a string. The beginning and the end of the string are identified with brackets (i.e. [ and ], respectively). Each pair of coordinates is also identified by the same brackets as [LONGITUDE, LATITUDE]. This list contains one pair of coordinates for each 15 s of trip. The last list item corresponds to the trip's destination while the first one represents its start |