Where Is the Next Path? A Deep Learning Approach to Path Prediction Without Prior Road Networks

Guiling Wang[1,2(✉)], Mengmeng Zhang[1,2], Jing Gao[1,2], and Yanbo Han[1,2]

[1] Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, North China University of Technology, No.5 Jinyuanzhuang Road, Shijingshan District, Beijing 100144, China
wangguiling@ict.ac.cn

[2] School of Information Science and Technology, North China University of Technology, No.5 Jinyuanzhuang Road, Shijingshan District, Beijing 100144, China

Abstract. Trajectory prediction plays an important role in many urban and marine transportation applications, such as path planning, logistics and traffic management. The existing prediction methods of moving objects mainly focus on trajectory mining in Euclidean space. However, moving objects generally move under road network constraints in the real world. It provides an opportunity to take use of road network constraints or de-facto regular paths for trajectory prediction. As yet, there is little research work on trajectory prediction under road network constraints. And these existing work assumes prior road network information is given in advance. However in some application scenarios, it is very difficult to get road network information, for example the maritime traffic scenario on the wide open ocean. To this end, we propose an approach to trajectory prediction that can make good use of road network constrains without depending on prior road network information. More specifically, our approach extracts road segment polygons from large scale crowdsourcing trajectory data (e.g. AIS positions of ships, GPS positions of vehicles etc.) and translates trajectories into road segment sequences. Useful features such as movement direction and vehicle type are extracted. After that, a LSTM neural network is used to infer the next road segment of a moving object. Experiments on real-world AIS datasets confirm that our approach outperforms the state-of-the-art methods.

Keywords: Trajectory prediction · LSTM · Crowdsourcing trajectories · AIS data

1 Introduction

In recent years, with the development of Cloud Computing, IoT, Big Data, 5G, machine learning and location-based services (LBS) technologies, it is possible for

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 350, pp. 219–235, 2021.
https://doi.org/10.1007/978-3-030-67540-0_13

collecting massive spatio-temporal trajectory data of large scale moving objects and mining valuable hidden information from such crowdsourcing trajectory data. Among various trajectory mining research topics, trajectory prediction is one of the most interesting and significant topic. The trajectory prediction plays an important role to solve problems such as path planning and traffic management.

There is some research work on trajectory prediction for land transportation based on GPS trajectory data [4]. Most of the trajectory prediction approaches are based on Euclidean space. They simulate the moving trajectory through mathematical formulas or mine the moving pattern from historical trajectories. However, these approaches have obvious limitations. Most moving objects move on a restricted road network and are necessarily restricted by the network and cannot move freely in space. This is why some of the research work tries to take use of road network information for trajectory prediction. We all know that there exists relatively stable road networks for land transportation. For maritime traffic, however, it is very difficult to get stable and accurate road network information for maritime traffic. In fact, although there is no static road network for maritime traffic, the moving trajectories of a large number of ships show certain de-facto paths. This is due to several reasons such as people always select routes with more fuel-efficiency, ships would not pass through existing protected sea areas where maritime traffic is prohibited and ships always bypass areas with well known security threats.

Our previous research work [6,16] proposed a method for extracting channels as polygons from large scale crowdsourcing ship trajectories. And our method in [12] further extracts road centerline and construct road network from the crowdsourcing ship trajectories. Based on these previous work, we aim to solve the problem of trajectory prediction that can make good use of road network constrains without depending on prior road network information based on large scale crowdsourcing trajectory data. Throughout this paper, we take ship trajectory prediction on the open ocean as an example. And the ship trajectories we use are Automatic Identification System (AIS) data.

The contributions of this paper are:

1) We propose a deep learning approach to trajectory prediction based on LSTM (Long Short-Term Memory) model from large scale crowdsourcing trajectory data. The method can predict the next road segment that moving objects would reach or pass without any prior knowledge about the road network.
2) We label each trajectory with road segments and transform the trajectories into sequences of road segments. We also analyze the features of ships that affect the trajectory prediction and select type of ships and ship's direction of travel as features and joint with sequences of road segments as input of LSTM model.
3) We compare with the trajectory prediction method based on frequent sequence mining algorithm. Experiments show that our method is more effective.

The rest of this paper is organized as follows. Section 2 reviews related work. In Sect. 3, the basic concepts throughout the paper are introduced and the problems to be solved are described. Section 4 describes trajectory serialization feature extraction. Section 5 introduces the LSTM methods for prediction. Section 6 introduces the experiments and compares it with the trajectory prediction method based on frequent sequence mining. Section 7 concludes the paper and describes future work.

2 Related Work

At present, a series of research achievements have been made on trajectory prediction of moving objects. According to the prediction duration, the existing trajectory prediction methods can be divided into long-term prediction and short-term prediction. Long-term prediction generally focuses on the motion of the moving object after a few hours, while short-term prediction only focuses on the motion of the moving object within a few seconds.

According to different methods, long-term prediction can be divided into the following categories: (1) Deep learning based long-term trajectory prediction methods. Some methods obtain the destination set through clustering, and use the RNN model to predict the destination [3]. There are also some other methods that use LSTM prediction [20] model based on user similarity. (2) Long-term trajectory prediction methods based on frequent pattern mining. The idea is to mine frequent patterns of moving objects through analysis of historical data, and predict the future of moving targets by matching the sequence to be predicted and the frequent sequence movement track. This type of method mainly focuses on the selection of frequent pattern mining algorithms and the selection of matching strategies. Some researchers have proposed a frequent pattern mining algorithm based on prefix projection [5,8,9], and proposed three matching strategies: overall match, tail match, longest tail match. There are also some methods to sort the matching results according to confidence and output the top-k prediction results [17]. (3) Clustering based trajectory prediction methods. Some methods [19] use changes in simplified trajectory directions to identify inflection points, and use the DBSCAN algorithm to cluster inflection points to obtain turning nodes. Then the ant colony algorithm is used to find the optimal path from the starting turning node to the ending turning node to achieve the purpose of prediction. There are also some trajectory prediction methods of clustering trajectories based on trajectory similarity firstly, and then predicting through trajectory matching [15].

Short-term prediction methods can also be divided into multiple categories: (1) Deep learning based short-term trajectory prediction methods. Some researchers have proposed a method based on LSTM model to predict the trajectory of vehicles driving on the highway [1], this method takes into account the characteristics of the target vehicle and other vehicles in the vicinity, and can predict the trajectory in the next 10 seconds. Other methods adopt RNN [7] and ANN [21] models. (2) Short-term trajectory prediction methods based on

Gaussian regression model. These methods focus on the current motion state of moving objects, and achieve the purpose of prediction by establishing a moving model of moving objects. Some researchers have proposed a mixed model based probabilistic trajectory prediction method, which uses previously observed motion patterns to infer the probability distribution as a motion model, which can effectively predict the position of moving objects within 2 seconds [2,14]. Some researchers use Kalman filter [10] to estimate the state of the dynamic behavior of moving objects, update the estimation of state variables using the previous value and the observation value of the current time, and then predict the trajectory position at the next time. (3) Short-term trajectory prediction method based on Markov model [11,18]. The idea is to mine the hidden state and determine the parameters of Markov model for trajectory prediction from historical trajectories.

Only a few of the above prediction methods are based on road network constraints [5]. However, in addition to trajectory data, this approach relies on a priori knowledge of the road network. Different from these related work, this paper realizes the trajectory prediction only based on massive crowdsourcing trajectory data without relying on any prior knowledge of the road network. And different from [5], our approach is based on LSTM model instead of frequent pattern mining algorithm.

3 Definitions and Problem Description

3.1 Definitions

Several definitions involved in this article are as follows:

Definition 1 (*Trajectory*)**.** A trajectory of a moving object (a ship or vehicle) is a set of trajectory points $T = \{t_1, t_2, t_3, ..., t_n\}$. Each ship or vehicle has its unique identification (i.e. Maritime Mobile Service Identify (MMSI) for ships), each trajectory is composed of multiple trajectory points, a trajectory point t_i is composed of a triple (utc_i, lon_i, lat_i), where utc_i is the time stamp of the occurrence of trajectory point t_i , lon_i is the longitude of track point t_i, lat_i is the latitude of track point t_i.

Definition 2 (*Road Boundary*)**.** A road boundary is a plane polygon set $C = \{c_1, c_2, c_3, ..., c_n\}$. A road polygon represents the boundary of the ocean road, which is composed of multiple vertices, arranged in a clockwise sequence, which can be expressed as $c_i = \{p_0, p_1, ..., p_n\}$, $p_i = \{lon_i, lat_i\}$ represents the vertices of the road polygon. Sometimes there exists cavity polygons located inside the road polygon and represents an area (such as islands and reefs) that cannot be navigated.

Definition 3 (*Road Segment*)**.** A road network is composed of multiple road segments. Each road segment is also a polygon. It is also composed of multiple triangles, which can be expressed as $S = \{tri_1, tri_2, tri_3, ...tri_n\}$. Each triangle consists of three points, which can be expressed as $tri_i = \{p_x, p_y, p_z\}$. We will introduce why to represent road segments as a set of triangles in Sect. 4.1.

Base on the above definitions, given a trajectory point, we are able to determine if it is inside a road segment or not. Generally speaking, most of the trajectory points in a trajectory locate in some road segments. Thus a trajectory can be represented as a sequence of road segments: $t_i = \{S_1, S_2, ..., S_m\}$.

3.2 Problem Description

Problem Statement. Given a set of trajectories $\mathcal{T} = \{t_1, t_2, ..., t_n\}$, for any trajectory of a moving object $t_i = \{S_1, S_2, ..., S_m\}$, infer the next road segment S_{m+1} this moving object will reach.

4 Geometry Translation

4.1 Trajectory Serialization Under Network Constrains

The aim of trajectory serialization under network constrains is annotating each trajectory with a sequence of road segments. We firstly extract road boundaries with road segments annotation from large scale crowdsourcing trajectory data.

As shown in Fig. 1(a), since the original trajectory data is collected from a large number of remote sensors, there will be a lot of quality problems such as errors, different sampling rates and missing positions. In order to improve the data quality, the MapReduce parallel computing framework is used in our previous work for trajectory sampling, interpolation, denoising, and segmentation [6,16]. Figure 1(b) shows an intermediate result of the pre-processing process.

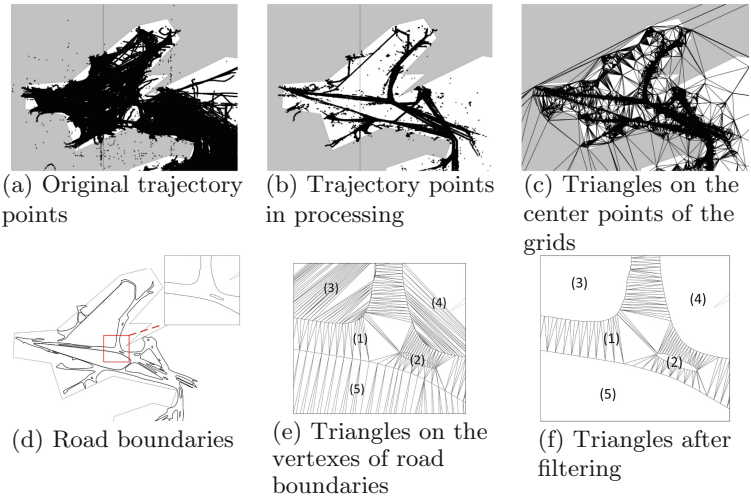
In order to extract the road boundaries, in our previous work [6,13], a parallel grid merging and filtering algorithm was proposed. Firstly we split the region of interest into grids with a parallel GeoHash encoding algorithm, and get the density values of the grids. After that, we perform grid merging according to the density values of the grids, then filters the merged grid data based on a local sliding window mechanism to get the grids within road boundaries. Then we apply the Delaunay Triangulation on the center points of the grids which are within road boundaries (as shown in Fig. 1(c)), filter the triangles within road and generate the polygons of road boundaries (for more details please refer to the proposed CirShape algorithm in our previous work [13]).

The extracted road boundary polygons have a lot of jagged edges. We need to smooth the boundaries before further processing. For the details of the smoothing method please refer to our previous work [12] An example result after smoothing is shown as Fig. 1(d).

The detailed network constrained trajectory serialization method could be divided into the following two steps:

(1) Extract polygons of road segments

We apply the Delaunay Triangulation on the vertexes of the smoothed road boundary polygons as shown in Fig. 1(e). Then the triangles outside the road



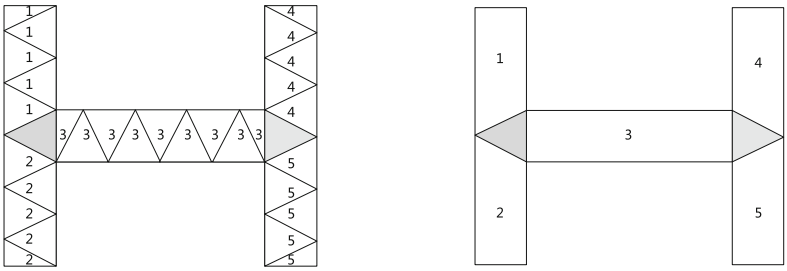
(a) Original trajectory points (b) Trajectory points in processing (c) Triangles on the center points of the grids

(d) Road boundaries (e) Triangles on the vertexes of road boundaries (f) Triangles after filtering

Fig. 1. Extraction of road boundaries and road segments

and inside the cavity (Area 2–5) are filtered. Among the left triangles (Area 1), those with three common edges are at the road intersection. The centers of gravity of these triangles are the nodes of road network and those triangles between two nodes are triangles on the road segments. The adjacent triangles are all traversed and all triangles on the road segments could be determined (For more details please refer to [12]). Then we can delete the inner edges of triangles on the road segments and only keep the edges forming the road segments. In another word, now we have polygons of road segments.

As shown in Fig. 2(a) below, there are a total of 5 road segments, and each road segment is composed of several triangles. Delete the inner edges of the road segment to get the road segment polygon, as shown in Fig. 2(b).



(a) (b)

Fig. 2. Extract polygons of road segments: an example

(2) Trajectory annotation with road segments

Given a trajectory, for each point of this trajectory, we search the road segment polygon which the point locates inside respectively. Thus a trajectory is annotated with road segments. Then we merge the repeated segments and get the final serialization results.

As shown in Fig. 3, there are 5 road segments in the road network. The black dots represent the trajectory points as $T = <P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}>$. We firstly annotate it with road segments as $T_s = <1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5>$, then get the final serialization result as $T_s = <1, 3, 5>$ after merging.

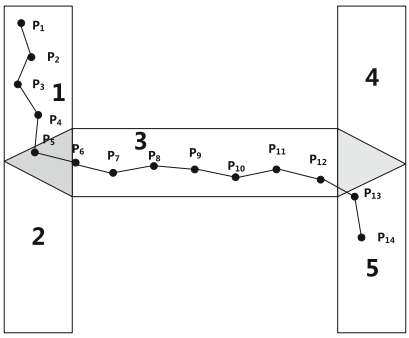


Fig. 3. An example trajectory serialization

4.2 Motion Feature Extraction

Data and features determine the upper limit of machine learning, and models and algorithms are only approaching that limit. In addition to the trajectory serialization, we extract some motion features from the trajectories to improve the accuracy of the prediction model.

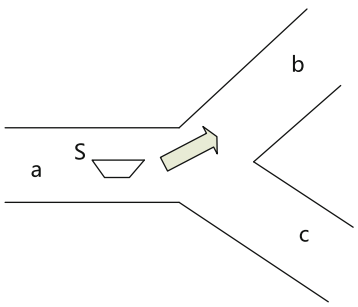


Fig. 4. Relationship between current movement direction and next road segment

Based on general observations of traffic, we believe that a vehicle's next segment destination is very related with its current moving direction when it travels across intersections. As shown in Fig. 4, if a ship S traveling on road segment a leaves the road segment in a direction shown by the arrow, the next road segment of S is likely to select the road segment b.

Here we use the Azimuth angle (Az) to measure the direction angel of movement. The calculation method of Az of two points is as follows: as shown in Fig. 5, each point is represented as longitude and latitude, and O is the earth's center. We calculate Az of point B relative to point A according formulas 1, 2, 3, 4 and 5.

$$\cos c = \cos(90 - B_{lat}) \cdot \cos(90 - A_{lat}) + \sin(90 - B_{lat}) \cdot \sin(90 - A_{lat}) \cdot \cos(B_{lon} - A_{lon}) \tag{1}$$

$$\sin c = \sqrt{1 - \cos^2(c)} \tag{2}$$

$$\sin A = \sin \frac{\sin(90 - B_{lat}) \times \sin(B_{lon} - A_{lon})}{\sin(c)} \tag{3}$$

$$A = \arcsin \frac{\sin(90 - B_{lat}) \times \sin(B_{lon} - A_{lon})}{\sin(c)} \tag{4}$$

$$A_z = \begin{cases} A, & Point\ B\ is\ in\ the\ first\ quadrant \\ 360 + A, & Point\ B\ is\ in\ the\ second\ quadrant \\ 180 - A, & Point\ B\ is\ in\ the\ third\ or\ fourth\ quadrant \end{cases} \tag{5}$$

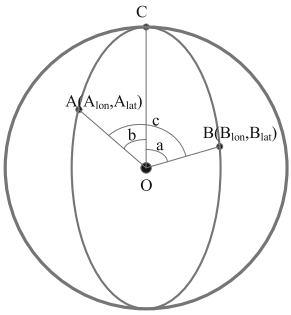


Fig. 5. Calculate the azimuth between two points A and B

The relationship between the direction angle and trajectories can be explained by comparing the direction of movement from the same segment to different segments. As shown in the Table 1, the distribution of direction always concentrates in a particular range.

Table 1. Statistics of direction distribution between different segments

Start-segment-ID	Arrival-segment-ID	Number-of-samples	Sailing-direction
30	31	157	5.4–6
30	59	356	2.5–3
60	61	421	5–6
30	105	477	2.5–3

For many reasons, such as climate, geographic environment and military, some road segments may only allow certain types of ships to travel. This article divides ships into six types, namely fishing vessels, cargo ships, oil tankers, passenger ships, container ships and other types. As shown in Table 2, we count the number of passes by different types of ships in different road segments. From the table, we can see that the number of different types of ships passing by the same segment is different, and different types of ships pass by the same segment. The proportion is different apparently. For some segments (e.g. segment 8), only certain types of ships will pass through. Therefore, road segments and types of ship are closely related. And the type of ship could be taken into considered as an important feature.

Table 2. Statistics of the number of different types of ships passing through road segments

Segment ID	Fishing boat	Cargo ship	Tanker	Passenger ship	Container Ship	Other types	Total number of passes
0	3	12	35	0	0	16	66
1	1	392	89	19	186	552	1239
2	0	38	4	0	9	6	57
3	1	2	2	0	0	4	9
4	3	22	6	0	0	15	46
5	1	19	3	0	1	5	29
6	0	37	12	0	9	55	113
7	1	34	1	0	0	5	41
8	0	3	0	0	0	3	6
9	2	101	13	0	9	52	177

5 Trajectory Prediction Based on LSTM Model

The single-layer LSTM architecture in this paper is shown in Fig. 6. First, the input layer receives tensors of fixed dimensions and length, then the LSTM layer consists of 256 neurons, and then one consists of 128 neurons. The fully connected layer uses Softmax as the activation function, and finally the output layer contains neurons with the same length as the output.

According to the characteristics of the input layer of the neural network, the input vectors needs to be divided into vectors of the same length. In this paper, the road segment sequences and features (direction and ship type) are selected as three-dimensional input vector units. We introduce the step size *step*, let $step = k$ (such as $k = 5$), then get a normalized $k * 3$ two-dimensional input tensor.

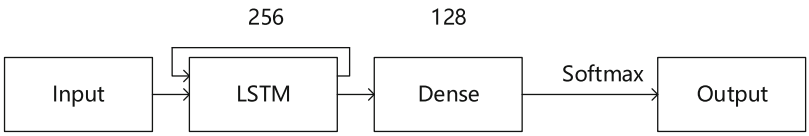


Fig. 6. LSTM model

6 Experiment and Analysis

6.1 Experimental Data

The dataset used in the experiment is the real AIS data from January 2017 to February 2017 in the Bohai Sea area. There are totally 15,725 ships in the dataset. After the sampling, denoising, interpolation and segmentation, a total of 10,668 trip records of 3574 ships were extracted. Table 3 gives the statistics for different types of ships:

Table 3. Dataset statistics for different types of ships

Ship type	Number of trips
Other	3644
Fishing boat	2197
Cargo ship	3295
Tanker	618
Passenger ship	376
Container Ship	538

Since there are some trajectory segments that cannot be matched with the road network, the trajectory segments should be filtered before serialization. We count the number of coordinate points in a trajectory segment located inside the road boundaries. If there are more than 80% of the coordinate points in a trajectory segment located inside the road boundaries, this segment can be serialized.

After preprocessing and serialization, there are a total of 3300 trips.

After serialization, the sequence of road segments is of varying lengths and need to be partitioned into vectors of the same length according to the requirement of the LSTM neural network input layer. As shown in Table 4, there is a sequence of road segments (a, b, c, d, e, f, g, h, i, j, k) that the ship passes through in a single trip. Each road segment passed is a three-dimensional vector, including the segment ID, ship type and travel direction. When we take $step = 5$, we get a $5 * 3$ tensor. So if we partition the complete sequence into several subsequences with step size of 5, we can get a three-dimensional normalized input tensor for any trajectory segment. The reason we take *step* as 5 is analyzed in Sect. 6.3.

Table 4. Examples of road segment sequences normalization

Road segment sequence	Input data samples
(a, b, c, d, e, f, g, h)	(a, b, c, d, e) (b, c, d, e, f) (c, d, e, f, g) (d, e, f, g, h)

After the above processing, we get over 33,000 data samples. Then the dataset is partitioned into training set, validation set and test set according to the ratio of 6:2:2 using the hold-out method for model training, hyper parameters determination and model evaluation respectively. The division of the dataset generated according to the step size of 5 is shown in Table 5:

Table 5. Partitioning of the dataset

Total number of samples	Training set	Validation set	Test set
33037	19822	6607	6608

6.2 Experimental Settings

This paper uses three metrics: average precision (*Macro-Precision*), average recall (*Macro-Recall*) and average F_1 value (*Macro-*F_1) to evaluate the performance of the model. For each prediction result, find its Precision, Recall, and F_1 values, and then take the average as the average accuracy, average recall, and average F_1 value.

For each prediction result, it can be divided into true examples (TP, the prediction results and the true values are positive examples), false positive examples (FP, the prediction results are positive examples, the true values are negative examples), and false negative examples (FN, the prediction result is a negative example and the true value is a positive example) and the true negative example

(TN, the prediction result and the true value are both negative examples). The calculation methods of its precision rate, recall rate and F_1 value are as follows:

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{8}$$

The calculation methods of average precision (*Macro-Precision*), average recall (*Macro-Recall*) and average F_1 value ($Macro\text{-}F_1$) are as follows:

$$Macro\text{-}Precision = \frac{1}{m} \cdot \sum_{i=1}^{m} Precision_i \tag{9}$$

$$Macro\text{-}Recall = \frac{1}{m} \cdot \sum_{i=1}^{m} Recall_i \tag{10}$$

$$Macro\text{-}F_1 = \frac{1}{m} \cdot \sum_{i=1}^{m} F_{1i} \tag{11}$$

The experimental environment of this article is shown in Table 6 below:

Table 6. Experimental environment

Operating system	Windows 7, 64-bit
CPU	Intel (R) Core (TM) i5-4200HQ CPU @ 1.60 GHz, 4 cores
RAM	4G
Tensorflow	Tensorflow-1.14.0
Keras	Keras-2.2.4
Java	1.8.0_241
Python	3.6.10

6.3 Results and Discussion

We perform the model tuning experiment firstly. Results of the experiment are shown in Table 7. The parameters of the model include loss function(Loss), activation function(Activation), optimizer(Optimizer), number of samples per batch (batch_size) and training rounds (Epochs). The last three columns in the table represent *Macro-Precision*, *Macro-Recall*, and $Macro\text{-}F_1$. We select *categorical crossentropy* as the Loss function.

Table 7. LSTM model tuning

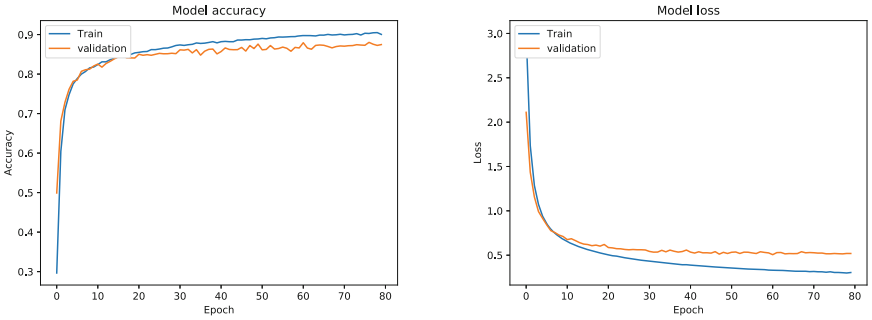
Test number	Activation	Optimizer	Batch size	Epochs	Training time	M-P	M-R	M-F_1
1	softmax	adam	32	20	139.148	0.847	0.858	0.844
2	sigmoid	adam	32	20	137.008	0.847	0.849	0.836
3	relu	adam	32	20	142.427	0.342	0.420	0.303
4	softmax	sgd	32	20	139.906	0.430	0.622	0.462
5	softmax	rmsprop	32	20	139.964	0.837	0.845	0.830
6	softmax	adam	50	20	135.195	0.840	0.852	0.835
7	softmax	adam	100	20	118.606	0.830	0.836	0.820
8	softmax	adam	200	20	112.956	0.817	0.824	0.803
9	softmax	adam	400	20	117.774	0.796	0.823	0.797
10	softmax	adam	20	20	160.907	0.848	0.856	0.842
11	softmax	adam	10	20	211.774	0.847	0.851	0.840
12	softmax	adam	32	40	173.147	0.874	0.873	0.866
13	softmax	adam	32	60	350.734	0.876	0.878	0.870
14	softmax	adam	32	80	308.795	0.884	0.885	0.879
15	softmax	adam	32	100	351.054	0.876	0.879	0.871

For experiments 1–3, we choose the same Optimizer, Batch_size, and Epochs, and softmax, sigmoid, and relu as the Activation functions respectively. The experimental results show that under the same other parameters, the time used to train the model is not much different, but the model trained using softmax as the activation function has a better effect on the test set.

For experiment 2, 4 and 5, we choose different Optimizers, and other parameters are the same. The experimental results show that we can find the direction of gradient descent faster with adam as the Optimizer when other parameters are the same. The model using adam as the optimizer has a better effect on the test set, and the time for training the model is not much different.

Experiment 2, 6–11 choose different Batch_size, other parameters are the same. The experimental results show that the average F_1 value of the test set will increase with the increase of batch_size within a certain range when other parameters are the same, but if this range exceeds the average F1 value will decrease with the increase of batch_size. The model trained with batch_size 32 is better on the test set. The model training time will decrease as the batch_size increases.

Experiment 1, 12–15 choose different Epochs, other parameters are the same. The experimental results show that the average F_1 value of the test set will increase with the increase of training times within a certain range, but when the training times increase to a certain number, overfitting will occur. Although the accuracy of the training set and the verification set has increased, the average F_1 value of the test set has decreased. The duration of model training increases with the number of training sessions.



(a) (b)

Fig. 7. Training curve of the 14th group

The results show that the 14th group of experiments is the best. The average accuracy rate is 0.884, the average recall rate is 0.885, and the average F_1 value is 0.879, with softmax as Activation function, adam as the Optimizer, 32 as the batch_size, and 80 as the Epochs times. The training curve is shown in Fig. 7. Figure 7-a is the model's Accuracy curve on the training set and validation set, and Fig. 7-b is the model on the training set and validation Loss curve on the set. We also use the controlled variable method to experiment with $step = \{3, 4, 5, 6, 7\}$. From the experimental results shown in Fig. 8 we can see that the effect is best when $step = 5$.

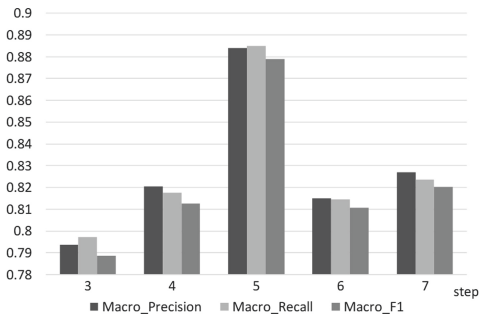


Fig. 8. Comparison of prediction effects for different step values

As analyzed in Sect. 2, only a few of the trajectory prediction methods are based on road network constraints. And the existing network constrained trajectory prediction approaches are based on frequent pattern mining algorithm, which is different with our work. So in the experiment, we compare our method with the trajectory prediction method based on frequent sequence pattern mining algorithm. We select 1,000 samples randomly as the test set, and other samples are used as the training set for frequent sequence pattern mining. Each

sample contains only the road segment sequence, and does not include other features such as direction and ship type. In the experiment, by setting the support degree to 20, 2153 frequent sequences can be obtained. The number of frequent sequences with different lengths (n-th order) is shown in Table 8:

Table 8. Statistics of frequent sequences for different lengths.

Frequent sequence length	2	3	4	5	6	7	8	9	10
Quantity	216	211	207	193	187	177	168	154	137
Frequent sequence length	11	12	13	14	15	16	17	18	19
Quantity	123	107	88	67	51	35	20	9	3

The sequences to be predicted are matched against frequent sequences, and the frequent sequence with maximum support value is taken as the prediction result. The average accuracy rate of the prediction is 0.812, the average recall rate is 0.824, and the average F_1 value is 0.815. Compared with our work, the prediction results based on LSTM model are better than those based on frequent sequence (PF). We believe that this is because the inputs of LSTM model include more features, including road segment sequences, direction and ship type, while the PF method can only adopt road segment sequences as inputs.

7 Conclusion and Future Work

In this paper, we study the problem of long term trajectory prediction under road network constraints only based on massive crowdsourcing trajectory data and without any prior knowledge about the road network. We propose a deep learning-based trajectory prediction method based on LSTM model. We extract the road segment information from large scale crowdsouring trajectory data and annotate the position sequences of the original trajectories with road segment polygons. In this way we translate the original trajectory into a road segment sequences. We also analyze the useful features and select movement direction and ship type jointly union with the road segment sequences as the input of LSTM model. Experiments on the real-world datasets show our method outperforms the network constrained trajectory prediction approach based on frequent pattern mining algorithm.

In the future, we will try to use the deep learning method to solve other long-term trajectory problems without any prior knowledge about road network such as predicting the final destination except for next road segment in this paper. We expect the idea of extracting and annotating trajectories with road segment polygons can also perform well in these problems.

Acknowledgments. This work is supported by National Natural Science Foundation of China under Grant 61832004 and Grant 61672042. We thank the Ocean Information Technology Company, China Electronics Technology Group Corporation (CETC Ocean Corp.), for providing the underlying dataset for this research.

References

1. Altche, F., de La Fortelle, A.: An LSTM network for highway trajectory prediction. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 353–359 (2017). https://doi.org/10.1109/ITSC.2017.8317913
2. Dalsnes, B.R., Hexeberg, S., Flåten, A.L., Eriksen, B.O.H., Brekke, E.F.: The neighbor course distribution method with Gaussian mixture models for AIS-based vessel trajectory prediction, pp. 580–5877 (2018)
3. Endo, Y., Nishida, K., Toda, H., Sawada, H.: Predicting destinations from partial trajectories using recurrent neural network. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10234, pp. 160–172. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57454-7_13
4. Georgiou, H.V., et al.: Moving objects analytics: Survey on future location & trajectory prediction methods. CoRR abs/1807.04639 (2018). http://arxiv.org/abs/1807.04639
5. Guo, L., Ding, Z., Hu, Z., Chen, C.: Uncertain path prediction of moving objects on road networks. J. Comput. Res. Dev. **47**, 104–112 (2010). (in Chinese)
6. Li, Z., Wang, G., Meng, J., Xu, Y.: The parallel and precision adaptive method of marine lane extraction based on QuadTree. In: Gao, H., Wang, X., Yin, Y., Iqbal, M. (eds.) CollaborateCom 2018. LNICST, vol. 268, pp. 170–188. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12981-1_12
7. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: a recurrent model with spatial and temporal contexts. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 194–200, AAAI Press (2016)
8. Qiao, S., et al.: Large-scale trajectory prediction model based on prefix projection technology. J. Softw. **28**, 3043–3057 (2017). (in Chinese)
9. Qiao, S., Han, N., Wang, J., Li, R.H., Gutierrez, L.A., Wu, X.: Predicting long-term trajectories of connected vehicles via the prefix-projection technique. IEEE Trans. Intell. Transp. Syst. **19**(7), 2305–2315 (2017)
10. Qiao, S., Han, N., Zhu, X.: Dynamic trajectory prediction algorithm based on Kalman filtering. Chin. J. Electron. **46**(2), 418–423 (2018). (in Chinese)
11. Qiao, S., Shen, D., Wang, X., Han, N., Zhu, W.: A self-adaptive parameter selection trajectory prediction approach via hidden Markov models, vol. 16, pp. 284–296. IEEE (2014)
12. Wang, G., Meng, J., Han, Y.: Extraction of maritime road networks from large-scale AIS data. IEEE Access **7**, 123035–123048 (2019)
13. Wang, G., Meng, J., Li, Z., Hesenius, M., Ding, W., Han, Y., Gruhn, V.: Adaptive extraction and refinement of marine lanes from crowdsourced trajectory data. Mob. Netw. Appl. 1392–1404 (2020). https://doi.org/10.1007/s11036-019-01454-w
14. Wiest, J., Höffken, M., Kreßel, U., Dietmayer, K.: Probabilistic trajectory prediction with Gaussian mixture models. In: 2012 IEEE Intelligent Vehicles Symposium, pp. 141–146. IEEE (2012)
15. Xie, B., Zhang, K., Zhang, Y., Cai, Y., Jiang, T.: Moving target trajectory prediction algorithm based on trajectory similarity. Comput. Eng. **44**(9), 177–183 (2018). (in Chinese)
16. Xu, Y., Li, Z., Meng, J., Zhao, L., Jianxin, W., Wang, G.: Extraction method of marine lane boundary from exploiting trajectory big data. Comput. Appl. **39**(1), 105–112 (2019). (in Chinese)
17. Yava, G., Katsaros, D., Ulusoy, Ö., Manolopoulos, Y.: A data mining approach for location prediction in mobile environments. Data Knowl. Eng. **54**, 121–146 (2005). https://doi.org/10.1016/j.datak.2004.09.004

18. Ye, N., Zhang, Y., Wang, R., Malekian, R.: Vehicle trajectory prediction based on hidden Markov model. Ph.D. thesis (2016)
19. Zhang, S., Shi, G., Liu, Z., Zhao, Z., Wu, Z.: Data-driven based automatic maritime routing from massive AIS trajectories in the face of disparity. Ocean Eng. **155**, 240–250 (2018)
20. Zhang, Z., Ni, G., Xu, Y.: Ship trajectory prediction based on LSTM neural network, pp. 1356–1364 (2020)
21. Zissis, D., Xidias, E.K., Lekkas, D.: Real-time vessel behavior prediction. Evolv. Syst. **7**(1), 29–40 (2015). https://doi.org/10.1007/s12530-015-9133-5