# A Road-Aware Neural Network for Multi-step Vehicle Trajectory Prediction

Jingze Cui, Xian Zhou, Yanmin Zhu(✉), and Yanyan Shen

Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, China
{misaki1,zhouxian,yzhu,shenyy}@sjtu.edu.cn

**Abstract.** Multi-step vehicle trajectory prediction has been of great significance for location-based services, e.g., actionable advertising. Prior works focused on adopting pattern-matching techniques or HMM-based models, where the ability of accurate prediction is limited since patterns and features are mostly extracted from historical trajectories. However, these methods may become weak to multi-step trajectory prediction when new patterns appear or the previous trajectory is incomplete.

In this paper, we propose a neural network model combining road-aware features to solve multi-step vehicle trajectory prediction task. We introduce a novel way of extracting road-aware features for vehicle trajectory, which consist of intra-road feature and inter-road feature extracted from road networks. The utilization of road-aware features helps to draw the latent patterns more accurately and enhances the prediction performances. Then we leverage LSTM units to build temporal dependencies on previous trajectory path and generate future trajectory. We conducted extensive experiments on two real-world datasets and demonstrated that our model achieved higher prediction accuracy compared with competitive trajectory prediction methods.

**Keywords:** Multi-step trajectory prediction
Road-aware features · LSTM

## 1 Introduction

Recent years have witnessed the rapid developments of wireless communication techniques, which gives rise to the availability of a huge number of GPS-enabled devices such as locators on taxies. And lots of location-based services such as navigation have been well developed, which brings great conveniences in various aspects of people's life, especially in transportation systems.

Typically, predicting vehicle future locations is the fundamental basis of most existing location-based services, (e.g., navigation route planning, location-based advertising, traffic management and so on), which has raised great attention to trajectory prediction problem. Furthermore, instead of predicting only for the


J. Pei et al. (Eds.): DASFAA 2018, LNCS 10827, pp. 701–716, 2018.
https://doi.org/10.1007/978-3-319-91452-7_45

next step (e.g., next one minute), the multi-step trajectory prediction is more favorable to most location-based services. The reason is that, results of multi-step prediction offer a relatively long-term trend information [16], which is crucial for applications like traffic management.

In this paper, we focus on solving the multi-step vehicle trajectory prediction problem, i.e., predicting vehicle locations in next few steps. The key challenges lie to (1) accurately extract the latent patterns, and (2) effectively build temporal dependencies on previous trajectory path. Specifically, GPS data usually suffers from noise interference and partly missing problems. Meanwhile, the road networks in reality are mostly complicated, which makes it difficult to extract the real latent patterns accurately. Moreover, it is complex to capture the tendency of varying trajectories when applying traditional machine learning approaches due to the intricate road networks.

Previous methods mostly employed pattern-matching techniques [3,7,14] or HMM-based models [12,15]. In such methods, the ability of trajectory prediction is limited when new patterns appear, or previous trajectory information is incomplete. Besides, it is inferior to build a prediction model under assumptions like vehicles proceed following the shortest route [1] or frequent trajectory patterns [6]. Such assumptions are unreasonable in some scenarios, e.g., travellers may want to take another longer route with multiple visiting places.

As for these problems, we have several key observations about road network. Firstly, the intrinsic property of road itself plays an important role in vehicle trajectories. For example, there are a large amount of trajectories going through the primary road, and the path through one-way road is always directional. With the information of inner propensity, the intra-road features can be used to help predict future trajectory with higher accuracy and confidence. Secondly, the connection between roads presents regularity, which can be learned from large quantity of trajectory data. We find that the connections are more determined by extrinsic factors, e.g., urban district structure. Usually, the connection between roads in different districts changes when the functionality of these districts is changed. Failure to utilize the inter-road features may compromise overall prediction performance.

Based on the above observations, we propose a neural network model combining road-aware features to solve multi-step trajectory prediction problem. First of all, we partition the road network into uniform road segments and project trajectories into sequences of road segments. Secondly, we extract the road-aware features and encode them into intra-road feature and inter-road feature embeddings for each road segment. The intra-road feature involves main intrinsic properties (e.g., road type). The inter-road feature is extracted based on social interactions with neighboring road segments, which is embedded in a continuous vector space. And we learn the inter-road feature embedding for each road segment via network embedding techniques by constructing a graph which reflects the correlations between roads. Thirdly, we propose **R**oad-**A**ware-**LSTM** model (**RA-LSTM**) based on long short-term memory network, which was proposed by [2] to specifically learn long-term dependencies in a sequence. Given a query

trajectory, after projecting into sequence of road segments, RA-LSTM takes as inputting the road-aware feature vectors of corresponding road segments, which concatenate intra-road and inter-road feature embeddings. Then we leverage LSTM to effectively capture temporal dependencies in the input sequence (i.e., sequence of road-aware feature vectors), and generate future trajectory.

The main contributions of this paper are as follows:

- We introduce a novel way of extracting road-aware features for vehicle trajectories. The road-aware features consist of intra-road feature and inter-road feature, which are able to help extract latent patterns accurately from trajectories and lead to a substantial increase in prediction accuracy according to our experiments.
- We propose RA-LSTM to achieve multi-step trajectory prediction. The model leverages LSTM units to capture temporal features from previous trajectory path.
- We conduct extensive experiments to evaluate the performance of proposed model on two real-world trajectory data sets. The experimental results show that our model outperforms three baseline methods by 28%–65% on two datasets.

The rest of the paper is organized as follows. Section 2 gives problem definitions. In Sect. 3, we provide an overview of the proposed model and preliminaries. Section 4 describes the details of our prediction model. The experimental results are presented in Sect. 5 and related work are given in Sect. 6. Finally, we conclude this paper in Sect. 7.

## 2 Problem Definition

To exploit road network information, we use road records downloaded from OSM[1]. Originally, roads are represented as variable-length sequences. We partition these roads into segments of identical length for fine-grained prediction.

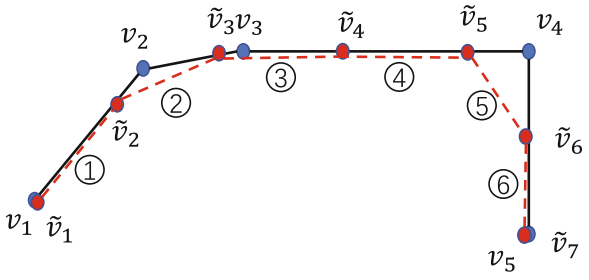



**Fig. 1.** Illustration of constructing road segments.

[1] http://www.openstreetmap.org/.

**Definition 1 (Road Segment Set).** *Let* $\mathcal{A} = \{r_i | i = 1, 2, \cdots, M\}$ *be road network records downloaded from OSM for a city (e.g., Shanghai) where* $r_i$ *indicates a road. We propose a road network partition algorithm(details in Sect. 3.2) to partition variable-length roads in* $\mathcal{A}$ *into road segments of equal length. And we denote the road segment set as* $\mathcal{RS} = \{s_i | i = 1, 2, \cdots, N\}$*. We use length* $g$ *to control the granularity of road segments. Intuitively, for a constant* $\mathcal{A}$*, smaller value of* $g$ *creates a larger number of road segments and fine-grained trajectory predictions.*

*For example, given a road* $r = v_1 v_2 \cdots v_5$ *as illustrated in Fig. 1 where* $v_i$ *means geographical vertex of road* $r$*. After partition, we get* $\tilde{r} = \tilde{v}_1 \tilde{v}_2 \cdots \tilde{v}_7$ *and create 6 segments of same length:* $\tilde{v}_1\tilde{v}_2, \tilde{v}_2\tilde{v}_3, \cdots, \tilde{v}_6\tilde{v}_7$ *marked as red dashed lines in Fig. 1.*

**Definition 2 (Trajectory Sequence).** *We consider a set of historical trajectories occurred in a city. Let* $\mathcal{T}$ *be a set of trajectories over one city. For some trajectory* $T = p_1 p_2 \cdots p_l$*,* $p_i = (lon_i, lat_i)$ *indicates longitude and latitude coordinates at timestamp* $i$*. Through map matching, each* $p_i$ *corresponds to a unique road segment* $s_{t_i}$*. And, we denote trajectory sequence as* $S = s_{t_1} s_{t_2} \cdots s_{t_l}$*, where each geographical point* $p_i$ *is contained in road segment* $s_{t_i}$*.*

**Definition 3 (Problem Definition).** *Consider a road network* $\mathcal{A}$ *in a city, given a query trajectory* $T = p_1 p_2 \cdots p_l$*, we focus on multi-step trajectory prediction problem which is to predict next* $k$ *trajectory points* $p_{l+1} p_{l+2} \cdots p_{l+k}$*.*

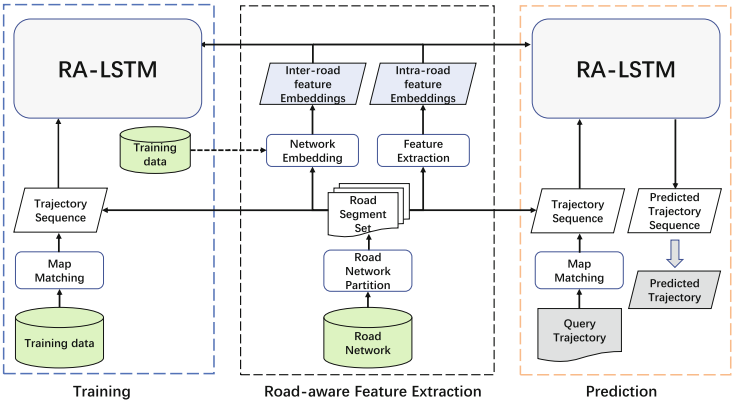



**Fig. 2.** Road-aware trajectory prediction framework

## 3 Overview and Preliminaries

### 3.1 Overview

Figure 2 provides an overview of our approach to the trajectory prediction task. The whole framework contains three parts: road-aware feature extraction, train-

ing and prediction. Road-aware feature extraction is applied to extract intrinsic and social features for each road segment, which are encoded as intra-road and inter-road feature embeddings. Training phase uses training data and road information to train a RA-LSTM. Given a query trajectory, we use the well-trained RA-LSTM and road feature embeddings to generate future trajectory. Road-aware feature extraction and training can be processed offline in advance.

*Road-Aware Feature Extraction:* Given a road network, road network partition is firstly applied to obtain road segment set. For each road segment, we extract road-aware features, which contain intra-road features and inter-road features. We learn inter-road feature embeddings via network embedding (details in Sect. 4.2), which is based on social connections between road segments. The intra-road features (details in Sect. 4.1) are mainly extracted from intrinsic properties (e.g., road types).

*Training:* Training data is created from historical vehicle trajectories occurred in a city. Through map matching, we obtain trajectory sequences based on road segment set and training data. RA-LSTM (details in Sect. 4.3) takes feature vectors of trajectory sequence as input and recognizes latent temporal features via LSTM network to generate future trajectory. Model parameters are updated using gradient descent during training phase. Training data is also useful for network embedding (details in Sect. 4.2).

*Prediction:* Given a query trajectory, we firstly convert query trajectory into trajectory sequence via map matching. Then the feature vectors of trajectory sequence are taken as input in RA-LSTM. Finally, the output of RA-LSTM is converted back to geographical points.

### 3.2 Preliminaries

**Road Network Partition.** Road partition is applied to convert road network $\mathcal{A}$ to road segment set $\mathcal{RS}$. Assume that a road $r$ in $\mathcal{A}$ is $v_1 v_2 \cdots v_m$ and after partition we get $\tilde{r} = \tilde{v}_1 \tilde{v}_2 \cdots \tilde{v}_{\tilde{m}}$. Define $D(v_1, v_2)$ as the distance between geographical vertices $v_1$ and $v_2$. We implement Algorithm 1 for each road $r$ in road network $\mathcal{A}$ and add corresponding segments to $\mathcal{RS}$. For each road $r$ in $\mathcal{A}$ (line 1), first we use an array to store accumulative distance from each geographical vertex to start vertex $v_1$ in road $r$ (lines 2–5), and calculate the number of vertices in $\tilde{r}$ (line 6). Then, we generate new vertices by solving linear and distance equations and add each segment into $\mathcal{RS}$ (lines 7–12). Finally, the last new vertex is the last origin vertex and we get the last segment of road $r$ (line 13).

**Trajectory Projection.** We use trajectory projection to convert original trajectory $T$ into trajectory sequence $S$. The widely used map matching approach based on HMM [12] is utilized to achieve trajectory projection. The core idea of the method is that a trajectory $T$ is modeled to move in the light of a Markov process between road segments. Road segments are considered as hidden states

**Algorithm 1.** Road Partition Algorithm

**Input:** road network $\mathcal{A}$, equal length $g$
**Output:** road segment set $\mathcal{RS}$

**for** each road r in network $\mathcal{A}$ **do**
  $cd$ =new array$[m]$, $cd[1] = 0$
  **for** $i = 2$ to $m$ **do**
    $cd[i] = cd[i-1] + D(v_{i-1}, v_i)$
  **end for**
  $\tilde{m} = \lceil cd[m]/g \rceil + 1$, $\tilde{v}_1 = v_1$
  **for** $i = 2$ to $\tilde{m} - 1$ **do**
    $seglen = (i-1) \cdot g$.
    find $j$ s.t. $cd[j] \leq seglen < cd[j+1]$
    find $\tilde{v}_i$ s.t.:
      (1) $\tilde{v}_i$ on the line defined by $v_j$ and $v_{j+1}$
      (2) $D(\tilde{v}_i, v_j) = seglen - cd[j]$
    $\mathcal{RS} \Leftarrow \tilde{v}_{i-1}\tilde{v}_i$
  **end for**
  $\tilde{v}_{\tilde{m}} = v_m$, $\mathcal{RS} \Leftarrow \tilde{v}_{\tilde{m}-1}\tilde{v}_{\tilde{m}}$
**end for**
**return** $\mathcal{RS}$

and geographical points are considered as observable states. Emission Probability describes how close the geographical point and road segment are. And state transition probability describes the condition probability from a road segment to another. Then Viterbi algorithm is applied to find the trajectory sequence $S$ with the highest probability corresponding to $T$.

In prediction phase, we convert predicted trajectory sequence $S$ back to trajectory geographical points. To simplify our model and calculation, we use midpoint of each segment as the corresponding geographical point. The reason is that we know the equal length of segments is $g$ and assume that the predicted points are following uniform distribution in a segment. Suppose a predicted point on the segment is in location $bg$ which $0 \leq b \leq 1$. Then we have the expected error $E(b)$ of the predicted point by Eq. 1:

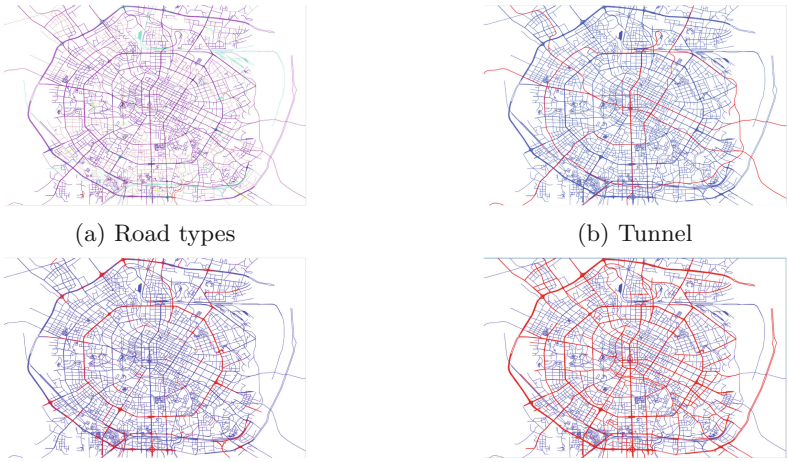
$$E(b) = \int_0^{bg} (bg - x)\frac{1}{g}dx + \int_{bg}^{g} (x - bg)\frac{1}{g}dx = (\frac{1}{2} - b + b^2)g \tag{1}$$

The minimum value of $E(b)$ is $g/4$ when $b$ =1/2. So we set the midpoint of the segment as the predicted point for prediction. Then we can get predicted trajectory $T$ from predicted trajectory sequence $S$.

## 4 Road-Aware Trajectory Prediction

### 4.1 Intra-road Feature Extraction

In this part, we present how to extract the intra-road features for each road segment and construct an intra-road feature matrix denoted as $M_{intra}$. We extract



(a) Road types (b) Tunnel

(c) Bridge (d) One-way

**Fig. 3.** Illustration of four intra-road features. (Color figure online)

four intra-road features: "road type", "tunnel", "bridge" and "one-way". Figure 3 gives an example of these features, using the data downloaded from OSM. The data describes the road network in the center of Chengdu in 2017.

We explain these four features respectively as follows:

**RoadType**. This feature describes the main traits of road which has 27 types in Fig. 3(a). Different colors of the line mean different types. Such as "primary","residential" and "track", they have been built for different application. Roads in different types also have different lengths and widths which affect the driver's choice of route.

**Tunnel**. The feature has only two value 0 and 1 in the data structure. 0 means the road is not a tunnel while 1 means it is. Red lines in Fig. 3(b) mean tunnels and blue lines mean other roads. Tunnels are scattered throughout the area and usually extend very long. Traveling in tunnels will limit vehicle's speed and sometimes lose GPS signals.

**Bridge**. Similar as **Tunnel**, red lines represent bridges and blue lines represent others in Fig. 3(c). Bridges concentrate in the edge area and have many intersections. Bridges have height information and may guide the travel trajectory of vehicles, such as a circle.

**One-way**. Similar as **Tunnel** and **Bridge**, it is shown in Fig. 3(d). One-way means the road is unidirectional for drivers and guides vehicle's driving tendency.

Since these features contain only category information, intra-road feature vectors concatenate the representations of each feature in one-hot encoding. Assume that the length of intra-road feature vector is $f$ and number of segments in segment set $\mathcal{RS}$ is $n$ in total. We take the intra-road feature vectors as row vectors to form intra-road feature matrix $M_{intra}$ of size $(n \times f)$.

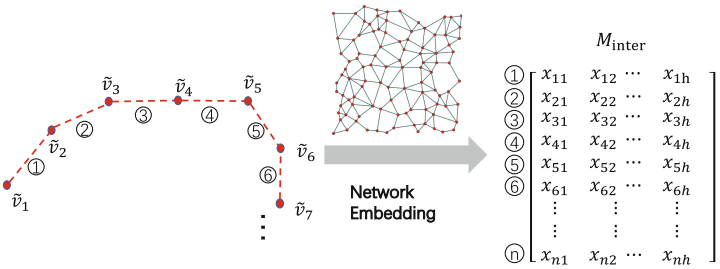



**Fig. 4.** Illustration of inter-road feature extraction

### 4.2 Inter-road Feature Extraction

In this section, we extract inter-road features based on social connections between road segments.

We use DeepWalk [9] which is proposed to learn latent representations of vertices in network, to learn inter-road feature embedding matrix denoted as $M_{inter}$. DeepWalk deploys truncated random walks to generate sequences of vertices in the network, and trains sequences with Skip-gram model [5] like sentences. After training, each vertex in network has an embedding representation.

If we only consider the location information of roads, road segment set $\mathcal{RS}$ can be regarded as an undirected graph with no weight. Since what we want is the inter-road feature vector representations of road segments not road vertices, in our context, road segments can be considered as "vertices" and their intersection vertices can be considered as "edges" in our network graph. To simplify road network graph and prominent its effect on trajectories, cause the road segments vehicles passed are connected, we take each two adjacent segments in a trajectory sequence $S$ as a pair of adjacent points in the graph. So all training sequences can construct a relatively simple graph.

After training, represented vectors of every segment according to order of road segments form a matrix $M_{inter}$(Fig. 4). There are in total $n$ segments in $\mathcal{RS}$ and assume each one is represented by a $(h \times 1)$ vector. Similar as $M_{intra}$, the size of $M_{inter}$ is $(n \times h)$.

### 4.3 Road-Aware Neural Network Prediction Model

In this part we first show the process of constructing RA-LSTM based on LSTM and two road feature matrices $M_{inter}$ and $M_{intra}$. Then, we present details of training and prediction.

**RA-LSTM.** Figure 5 provides detailed structure of RA-LSTM. Given input trajectory sequence $s_{t_1}, s_{t_2}, ..., s_{t_l}$, we firstly encode each input road segment $s_i$ as one-hot representation vector $d_i$, $d_i \in R^N$, where $N$ is the number of road segments. For each input road segment $s_i$, we get corresponding intra-road

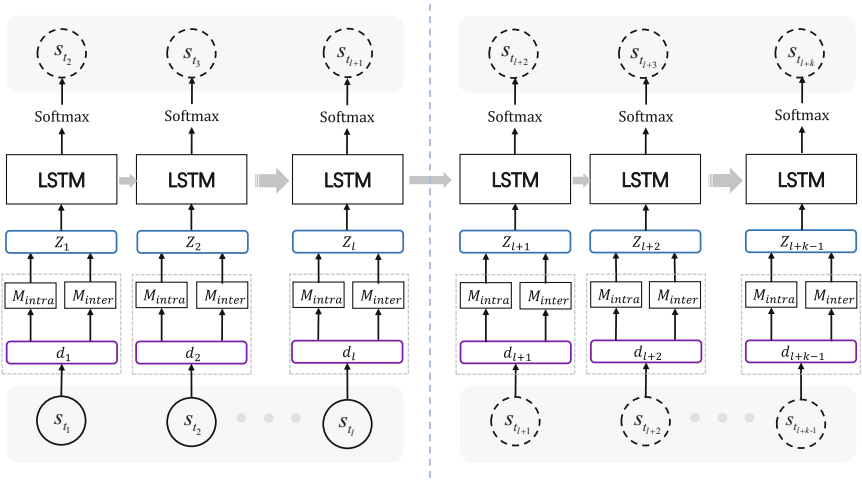



**Fig. 5.** Road-aware neural network prediction model

feature vector $M \cdot d_i$ and inter-road feature vector $M \cdot d_i$. And the concatenation of $M \cdot d_i$ and $M \cdot d_i$ is taken as final input feature vector denoted as $Z_i$, in Eq. 2.

$$Z_i = [M_{intra}^T \cdot d_i; M_{inter}^T \cdot d_i] \tag{2}$$

where $[;]$ indicates the operation of concatenating two vectors. $M_{intra}^T \cdot d_i \in R^{f \times 1}$, $M_{inter}^T \cdot d_i \in R^{h \times 1}$ and $Z_i \in R^{(f+h) \times 1}$.

Then LSTM network takes feature vector $Z_i$ as input one by one. The key equations of generating hidden states in LSTM are the following.

$$\begin{aligned} i_t &= \sigma(W_{xi} \cdot Z_t + W_{hi} \cdot H_{t-1} + b_i) \\ f_t &= \sigma(W_{xf} \cdot Z_t + W_{hf} \cdot H_{t-1} + b_f) \\ o_t &= \sigma(W_{xo} \cdot Z_t + W_{ho} \cdot H_{t-1} + b_o) \\ C_t &= f_t \circ C_{t-1} + i_t \circ tanh(W_{xc} \cdot Z_t + W_{hc} \cdot H_{t-1} + b_c) \\ H_t &= o_t \circ tanh(C_t) \end{aligned} \tag{3}$$

where $Z_t$ is the input vector at $t$-th step; $i_t$, $f_t$ and $o_t$ are input, forget and output gates; $C_t$ and $H_t$ are internal memory and hidden state respectively; '$\circ$' is the Hadamard product and $\sigma(\cdot)$ is sigmoid function; different kernel matrices $W$ and biases $b$ are parameters to be learned.

Afterwards, we use a softmax layer to get final output vector $y$, $y \in R^N$, where each value represents the probability of road segment at next step. The road segment with highest probability is the predicted $\hat{s_i}$ at $i$-th step.

**Training and Prediction.** We train and test our RA-LSTM model for multi-step trajectory prediction.

For training, we use weighted cross-entropy loss function as follows.

$$loss(\hat{s}, s) = -\frac{1}{l}\sum_{i=1}^{l}(s_i \log \hat{s}_i) \tag{4}$$

where $\hat{s}_i$ is our prediction, $s_i$ is the real segment label and $l$ is the length of trajectory sequence. And the mini-batch gradient descent algorithm is used to update parameters in RA-LSTM.

For prediction, assume that $s_{t_1} s_{t_2} \cdots s_{t_l}$ is the query trajectory sequence. We input it into RA-LSTM and get future road segments $s_{t_{l+1}}, s_{t_{l+2}} \cdots s_{t_{l+k}}$ one by one. Specifically, first we generate $s_{t_{l+1}}$ as the first predicted road segment, and we take it into neuron network as the input road segment for next step. After k steps, we get the predicted trajectory sequence $s_{t_{l+1}} s_{t_{l+2}} \cdots s_{t_{l+k}}$. At last, we convert predicted trajectory sequence back to geographical points.

## 5 Experiments

### 5.1 Experimental Setup

**Datasets.** We use two real-world trajectory datasets to evaluate our model. The corresponding road network datasets are acquired from OSM. Table 1 summaries the details:

**Table 1.** Trajectory dataset table

| Dataset | Trajectories | | | | Road network | | |
|---|---|---|---|---|---|---|---|
| | Coverage | # of vehicles | # of trajectories | Time interval | # of roads | # of vertices | # of features |
| Shanghai | 10 km × 8 km | 74 | 4859 | 10s | 193 | 1616 | 36 |
| Chengdu | 5 km × 5 km | 2428 | 18991 | 6s | 942 | 6094 | 31 |

**Shanghai**. The data was collected from device OBD of private cars in Fengxian district, Shanghai from July 2014 to January 2015. The covered area is the suburbs of Shanghai with coverage of 10 km×8 km. The unit of time is set as 10 s, which means the time interval between successive points is 10 s.

**Chengdu**. We use the collected taxi data from DiDi company in November 2016. The data contains 18991 orders and each order records a taxi trajectory of geographical coordinates. The covering area is downtown of Chengdu, where the number of roads are much more than that in Shanghai with narrower coverage. Due to the completeness of this dataset, we set the time interval as 6 s.

**Baseline Methods.** We compare our RA-LSTM model with both basic and advanced methods as follows:

- RMF [13] It is a descriptive model-based path prediction method, which computes motion function to capture movements.
- R2-D2 [15] R2-D2 is a HMM-based probabilistic method which predicts future trajectory path based on a few reference trajectories.
- LSTM [2] It is the vanilla version of LSTM model without road-aware features, where we use one-hot encoding of road segments as input.

**Experimental Settings.** We interpolate the trajectories to get synchronized timestamp 10 s and 6 s on Shanghai and Chengdu respectively, using Moving Object Cache, which is also used in R2-D2. We divide 80% dataset for training and 20% dataset for testing. In training dataset, we randomly select 20% for validation. We set the length of road segment $g = 50$ m The dimension of inter-road feature embedding and the number of hidden states in LSTM are both set as 128. The parameter of comparison methods are selected using validation set.

The experiments were conducted on a PC with Intel Core i5-7300 2.5 GHz, Nvidia GTX 1050ti and 16 GB RAM. RMF, LSTM and RA-LSTM were implemented in python with toolkit tensorflow 1.4.0, and R2-D2 was implemented in Java with JDK 1.7.

**Measurement.** We consider the input query trajectory with 5 previous geographical points and predict future locations for the next 5 steps. We evaluate all methods using *Average distance error* as follows.

$$Average\ distance\ error = \frac{1}{N} \sum_{i} \sum_{t} \|p_i^t - \hat{p}_i^t\|_2$$

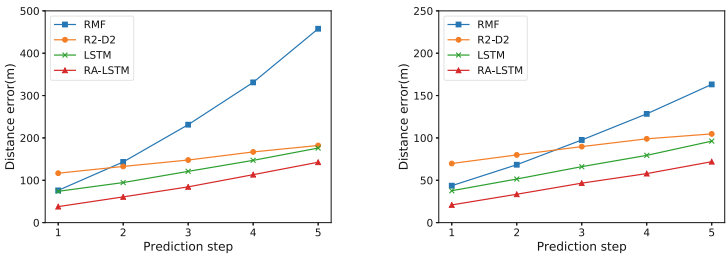
where $\hat{p}_i^t$ and $p_i^t$ are predicted coordinates and ground truth of query trajectory $i$ at $t$-th step; $N$ is the number of all the predicted value; and the errors are measured by Euclidean distance.

### 5.2 Results

**Comparison of Different Methods.** Table 2 shows the comparison results of 4 approaches over two datasets. All methods predict future trajectory for no more than 0.02 s and can be used in real-world applications. As we can see, all the methods get lower distance error in Chengdu than that in Shanghai. The main reason is that time interval dataset in Shanghai is longer than Chengdu, which makes it harder to capture latent patterns. Overall, RA-LSTM achieves the best results, which give 41% and 47% lower error than R2-D2. The reasons are two-fold. First, the input of R2-D2 prediction filter is restricted to a small set of reference trajectories, which cannot accurately reflect the latent patterns while RA-LSTM uses road-aware features in a global perspective. Second, probability

**Table 2.** Average distance error

| Dataset | RMF | R2-D2 | LSTM | RA-LSTM |
|---|---|---|---|---|
| Shanghai | 247.9 8m | 149.25 m | 122.53 m | **87.78 m** |
| Chengdu | 100.22 m | 88.63 m | 66.18 m | **46.18 m** |

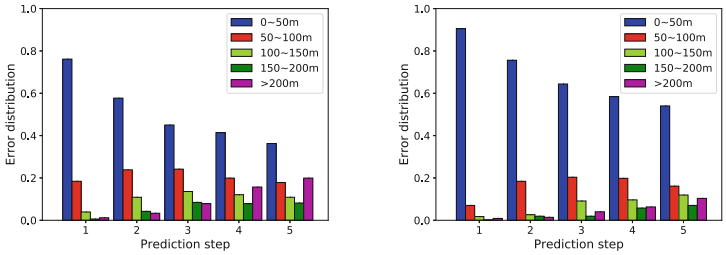



(a) Shanghai (b) Chengdu

**Fig. 6.** Step-wise average distance error

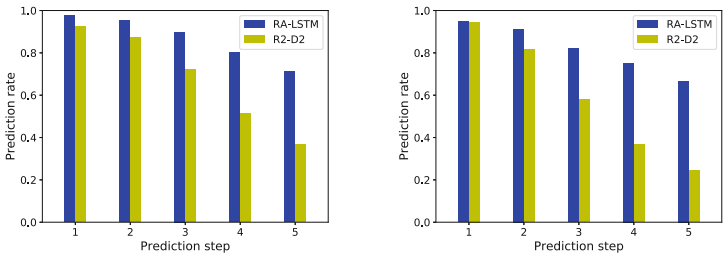
of predicted path is computed based on HMM model, where the states are set as the cluster of reference points, which occurs inevitable error. It is also the main reason that R2-D2 performs worse than LSTM. For both datasets, RMF obtains the worst results. This is because the moving function does not take environment constraints into account and it lacks necessary historical experience. It is worth mentioning that RA-LSTM outperforms significantly than vanilla LSTM, giving the improvements by 28% and 30% as showed in the Table 2. The improvements confirms that the utilization of road-aware features enhances performances for trajectory prediction.

**Results on Step-Wise Prediction.** For further analysis, we present the step-wise prediction results over two datasets in Fig. 6. Generally, RA-LSTM performs the best for all steps, achieving 22%–67% and 25%–65% lower distance error than other methods for dataset Shanghai and Chengdu, respectively. The two figures show that RMF is better than R2-D2 for the first step, but worse in the further steps. RMF is designed for predicting next step by constructing motion function. But in multi-step prediction, it takes the predicted location as input for further step, which accumulates distance error. The distance error of LSTM and RA-LSTM grow a little faster than R2-D2, and RMF behaves worst. One of the reason is, not all query trajectories are predictable in R2-D2, which is explained in the next part. The other important reason is that, different from R2-D2 which predicts the whole path, RMF and RA-LSTM make trajectory prediction step by step. The predicted point is taken as input for next step, which causes accumulated error. And we see that the distance error of LSTM and RA-LSTM increase more slowly than RMF, demonstrating that memory cells in LSTM are able to capture long-term features effectively.




(a) Shanghai (b) Chengdu

**Fig. 7.** Distance error distribution




(a) Shanghai (b) Chengdu

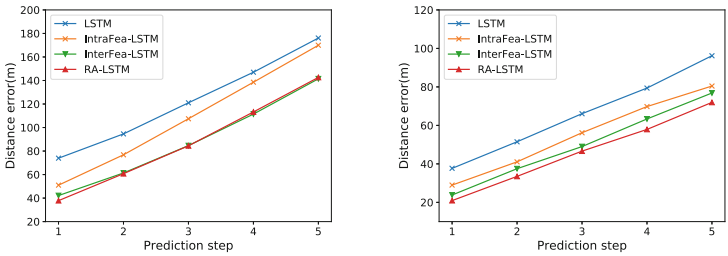
**Fig. 8.** Comparison of prediction rate

**Comparison of Prediction Rate.** R2-D2 selects the reference trajectories of high similarity (i.e., higher than a given threshold $\theta$) with query trajectory. Those query trajectories with no reference trajectories are regarded as unpredictable. Therefore, R2-D2 adopts a metric called *prediction rate*, which is the fraction of the number of predictable query trajectories over the total number of query trajectories. The prediction rate reflects the robustness of a prediction model. Obviously, in R2-D2, those unpredictable query trajectories are still 'predictable' if we set a weakened threshold, although the distance error is probably large.

To compare with R2-D2 on the metric of prediction rate, we list error distribution for each step in Fig. 7. We set a threshold $\hat{\theta}$ and consider the query trajectories with distance error higher than $\hat{\theta}$ as unpredictable. The thresholds $\hat{\theta}$ are set as 200 and 100 for Dataset Shanghai and Chengdu, respectively. The thresholds are approximately twice of average distance error reported by RA-LSTM and no more than twice of that in R2-D2. Then we compare the prediction rate with R2-D2 in each step as showed in Fig. 8. Prediction rate of RA-LSTM at each step is higher than R2-D2 and the superiority is more obvious in the further step, which validates the effectiveness of RA-LSTM on multi-step trajectory prediction.

**Results of Ablation Test.** In Fig. 9, we present ablation results for RA-LSTM. Specifically, 'IntraFea-LSTM' refers to LSTM model with only intra-road feature embedding and 'InterFea-LSTM' represents LSTM model with only inter-road

feature embeddings. From the Fig. 9(a) and (b), RA-LSTM with road-aware features that contains both intra and inter features achieves the best or comparable performance for all steps on both datasets. Either IntraFea-LSTM or InterFea-LSTM performs better than vanilla LSTM, which demonstrates both intra and inter features of roads assist in recognizing latent patterns. InterFea-LSTM gives better results than IntraFea-LSTM on both datasets. We see that inter-road feature learned by exploiting connections between roads is more helpful than intra-road feature.




(a) Shanghai (b) Chengdu

**Fig. 9.** Results of ablation test

## 6 Related Work

There are many researches about vehicle trajectory prediction, which can be divided into three types: trajectory pattern mining, prediction based on moving function and Markov model and using road networks.

**Pattern-Matching Prediction.** Pattern-based prediction methods are dedicated to find frequent trajectories and discover their movement rules. In [7,8], Morzy et al. proposed an improved Apriori algorithm and improved PrefixSpan algorithm. A hybrid prediction model which combines motion function and trajectory patterns was utilized in [3] using object's recent movement to predict. For next location prediction without involving the path to the destination, the method proposed in [6] built and trained a decision tree named T-pattern tree to find best matching result in prediction. In addition, besides using only geographical patterns, Ying et al. [14] proposed an approach also using both geographical and semantic patterns of trajectory. Pattern-based methods consider the association with moving objects which loses the generality of the prediction. If you query for moving objects that do not fit any pattern, their trajectories are unpredictable.

**HMM-Based Prediction.** This part mainly introduces motion function and HMM. RMF [13], a most accuracy motion function which constructs a state matrix by query trajectory and uses Singular Value Decomposition(SVD) to calculate the predicted point. The drawback of moving function is that it has no communication with environment and it can not learn information from history trajectories. Methods based on HMM are suitable for estimating future locations in discrete space. In [15], Zhou et al. proposed a "semi-lazy" approach to make prediction in dynamic environment which is used as baseline in our experiments. This method finds grid-based reference trajectories, and the longest reference path whose probability of retrieval is greater than a preset threshold is retrieved as the predicted path by HMM. Authors in [11] improved the HMM for with self-adaptive parameter selection. These methods focus on prediction algorithm design regardless of external environment information. Our approach adds road-aware features for vehicle trajectory prediction which has been proven more effective than not adding.

**Prediction with Road Network.** Before our works, there are several related works using road network for prediction. A maximum likelihood model and a greedy algorithm were built for long-term travel path prediction in [4]. But the method only considers the last location of query trajectories which may lose history information. Qiao et al. [10] proposed a three-in-one Trajectory-Prediction model in road-constrained transportation networks called TraPlan. Predictive Tree has also been proposed to maintain the reachability of road segments using additional information such as road length [1]. Most of them use large-scale road network dasebase to match trajectories with high similarity for prediction, while our model adopts network embeddings to encapsulate road-aware features for accurate trajectory predictions.

## 7 Conclusion

In this paper, we study the problem of multi-step vehicle trajectory prediction and propose a road-aware neural network solution. Unlike previous approaches relying on historical trajectories, we introduce a novel way of extracting road-aware features for vehicle trajectories, which consist of intra-road feature and inter-road feature exploited from road networks. And we leverage LSTM units to capture temporal dependencies on previous trajectory path to predict future trajectory. The experiments on two real-world datasets demonstrated that (1) our proposed model RA-LSTM outperforms other competitive methods; (2) the road-aware feature extracted from road networks can effectively improve the multi-step prediction performance.

**Acknowledgment.** This research is supported in part by 973 Program (No. 2014CB340303), NSFC (No. 61772341, 61472254, 61170238, 61602297 and 61472241), and Singapore NRF (CREATE E2S2). This work is also supported by the Program for Changjiang Young Scholars in University of China, the Program for China Top Young Talents, and the Program for Shanghai Top Young Talents.

## References


1. Hendawi, A.M., Bao, J., Mokbel, M.F., Ali, M.: Predictive tree: an efficient index for predictive queries on road networks. In: 2015 IEEE 31st International Conference on Data Engineering (ICDE), pp. 1215–1226. IEEE (2015)
2. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
3. Jeung, H., Liu, Q., Shen, H.T., Zhou, X.: A hybrid prediction model for moving objects. In: IEEE 24th International Conference on Data Engineering, ICDE 2008, pp. 70–79. IEEE (2008)
4. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S.: Path prediction and predictive range querying in road network databases. VLDB J. **19**(4), 585–602 (2010)
5. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
6. Monreale, A., Pinelli, F., Trasarti, R., Giannotti, F.: WhereNext: a location predictor on trajectory pattern mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 637–646. ACM (2009)
7. Morzy, M.: Prediction of moving object location based on frequent trajectories. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 583–592. Springer, Heidelberg (2006). https://doi.org/10.1007/11902140_62
8. Morzy, M.: Mining frequent trajectories of moving objects for location prediction. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 667–680. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73499-4_50
9. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. ACM (2014)
10. Qiao, S., Han, N., Zhu, W., Gutierrez, L.A.: TraPlan: an effective three-in-one trajectory-prediction model in transportation networks. IEEE Trans. Intell. Transp. Syst. **16**(3), 1188–1198 (2015)
11. Qiao, S., Shen, D., Wang, X., Han, N., Zhu, W.: A self-adaptive parameter selection trajectory prediction approach via hidden Markov models. IEEE Trans. Intell. Transp. Syst. **16**(1), 284–296 (2015)
12. Raymond, R., Morimura, T., Osogami, T., Hirosue, N.: Map matching with hidden Markov model on sampled road network. In: 2012 21st International Conference on Pattern Recognition (ICPR), pp. 2242–2245. IEEE (2012)
13. Tao, Y., Faloutsos, C., Papadias, D., Liu, B.: Prediction and indexing of moving objects with unknown motion patterns. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 611–622. ACM (2004)
14. Ying, J.J.C., Lee, W.C., Weng, T.C., Tseng, V.S.: Semantic trajectory mining for location prediction. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 34–43. ACM (2011)
15. Zhou, J., Tung, A.K., Wu, W., Ng, W.S.: A semi-lazy approach to probabilistic path prediction in dynamic environments. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 748–756. ACM (2013)
16. Zhou, X., Shen, Y., Zhu, Y., Huang, L.: Predicting multi-step citywide passenger demands using attention-based neural networks. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 736–744. ACM (2018)