On Prediction of User Destination by Sub-Trajectory Understanding: A Deep Learning based Approach

Jing Zhao[†], Jiajie Xu[†, §] [*], Rui Zhou[‡] [*], Pengpeng Zhao[†], Chengfei Liu[‡], Feng Zhu[¶]
[†]Institute of Artificial Intelligence & School of Computer Science and Technology, Soochow University, China
[§]State Key Laboratory of Software Architecture (Neusoft Corporation), China
[‡]Swinburne University of Technology, Australia
[¶]Siemens Corporate Technology, Suzhou, China
20175227005@stu.suda.edu.cn, {xujj, ppzhao}@suda.edu.cn, {rzhou, cliu}@swin.edu.au, zhufeng@siemens.com

ABSTRACT

Destination prediction is known as an important problem for many location based services (LBSs). Existing solutions generally apply probabilistic models to predict destinations over a sub-trajectory, but their accuracies in fine-granularity prediction are always not satisfactory due to the data sparsity problem. This paper presents a carefully designed deep learning model called TALL model for destination prediction. It not only takes advantage of the bidirectional Long Short-Term Memory (LSTM) network for sequence modeling, but also gives more attention to meaningful locations that have strong correlations w.r.t. destination by adopting attention mechanism. Furthermore, a hierarchical model that explores the fusion of multi-granularity learning capability is further proposed to improve the accuracy of prediction. Extensive experiments on Beijing and Chengdu real datasets finally demonstrate that our proposed models outperform existing methods without considering external features.

KEYWORDS

trajectory prediction; trajectory embedding; deep learning

ACM Reference Format:
Jing Zhao, Jiajie Xu, Rui Zhou, Pengpeng Zhao, Chengfei Liu, Feng Zhu. 2018. On Prediction of User Destination by Sub-Trajectory Understanding: A Deep Learning based Approach. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271708

1 INTRODUCTION

The widespread use of smart phones and in-car navigation systems has become part of daily life, which leads to the popularity of the embedded GPS devices and the rapid development of positioning technology. We benefit increasingly from various types of location based services (LBSs) which function in all aspects of our lives [12, 14]. *Destination prediction*, which is used for predicting the destination of a trajectory while a sub-trajectory is already given, has become increasingly important. A great quantity of LBSs require the function of accurate destination prediction to perform efficient and feasible services, such as recommending scenic spots, sending targeted advertisements (taverns, eating house, etc.), and automatically set destination in navigation systems.

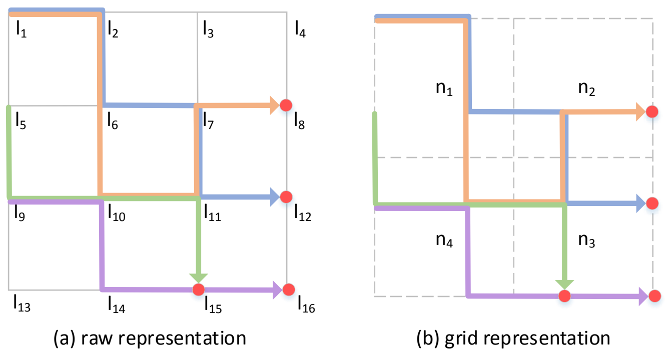


Figure 1: An illustrative example

With large scale user trajectories available nowadays, accurate destination prediction can be achieved by understanding user mobility patterns. Classical methods generally adopt a trajectory search based manner, i.e., to search the trajectories in database that can cover the given sub-trajectory. By aggregation, the top-k destinations of these trajectories are chosen as the final predicted destinations. However, we often end up with finding no or very few trajectories in dataset that can cover the given sub-trajectory. So far, many statistical approaches generally decompose raw trajectories into grid-based trajectories, and predict destinations in spatial grid granularity. The basic idea is to calculate the probabilities of possible paths by the Bayesian inference framework, and then derive the destination probabilities by Bayes' rule [23]. However, the limited learning ability brings the concern that these models can only support coarse-granularity prediction in most cases.

Example 1: As shown in Figure 1 (a), there are four historical trajectories: $t_1 = \{l_1, l_2, l_6, l_7, l_{11}, l_{12}\}$, $t_2 = \{l_1, l_2, l_6, l_{10},$

[*]The corresponding author of the work.

$l_{11}, l_7, l_8\}$, $t_3 = \{l_5, l_9, l_{10}, l_{11}, l_{15}\}$, and $t_4 = \{l_9, l_{10}, l_{14}, l_{15}, l_{16}\}$. Each location l_i indicates an intersection of the road network, and the lines connecting them are road segments. If a sub-trajectory $\{l_5, l_9\}$ is given, it matches the front part of history trajectory t_3, thus we take its ending point l_{15} as the destination. However, if the given sub-trajectory is $\{l_1, l_5, l_6, l_{10}\}$, trajectory search based methods will return no result since no trajectory in Figure 1 (a) can cover it. Now assume that the whole area is decomposed into four spatial grids, denoted by n_1, n_2, n_3, n_4 in Figure 1 (b), and the sub-trajectory $\{l_1, l_5, l_6, l_{10}\}$ will be transformed to $\{n_1, n_4\}$ in grid granularity. By applying [23], we can predict n_3 as the region that the user most likely to terminate, since most of the trajectories passing by regions n_1 and n_4 end in region n_3. However, the accuracy in [23] is not satisfactory when the spatial region is not sufficiently small. While in reality, the LBS systems actually call for improved models that can provide accurate destination prediction in fine granularity, so that the quality of personalized advertising and scenic spot recommendation can be ensured.

Now we face several new technical challenges. Firstly, it causes more significant data sparsity. Each spatial grid would be smaller when it is in fine granularity. Consequently, a trajectory will correspond to a long sequence of grids accordingly, which aggravates data sparsity and impedes the effectiveness of traditional statistical inference models. Secondly, useful mobility patterns may appear in different spatial granularities, rather than the one for prediction only. A more complex model is thus required to learn the implicit features and patterns in different granularities, and then fuse them together rationally. In addition, as human mobility patterns may vary significantly under different weather conditions, holidays and etc., external features should be considered in the training to improve the accuracy.

In recent years, among various kinds of deep neural network architectures available, *recurrent neural network* (RNN) has been widely used to analyze the structure of the time series data. Empirical studies have also proved that LSTM network (one of the prominent variants of RNN) has comparable performance in modeling time series data with variable length, such as machine translation [18], image annotation [9] and speech recognition [21]. Since destination prediction also relies on time series modeling (of trajectory), it has also been demonstrated to be a good opportunity for our problem. Nevertheless, the standard LSTM model is insufficient to support accurate fine-grained prediction itself, especially when the learning of features and patterns in different granularities is considered, thus a more specific model is required. In addition, the strong correlations may be missed out if the locations are far from each other in the trajectory sequence, and some important locations that have strong correlation to destination must be given more attention.

To fully address the above mentioned issues, this paper proposes a novel deep learning based method called TALL model for fine-grained prediction. In this model, bidirectional LSTM is used to model trajectories, so that latent features of both preceding and following locations in trajectory are considered in learning, and external features can be easily fused to reflect context aware mobility patterns. With the help of attention mechanism, the model can identify and emphasize on the features that have strong correlations to the destination, regardless of their locality in trajectory to avoid the flaws of LSTM network. By integrating bidirectional LSTM and attention mechanism rationally, this model has shown significant advantages to solve the data sparsity problem in fine granularity prediction. In addition, we further seek to integrate the multi-granularity knowledge to improve accuracy, and a hierarchical deep learning model is further proposed to merge the mobility patterns learned from different spatial granularity.

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

- We propose a deep learning based method that incorporates a bidirectional LSTM model and attention mechanism. The method not only can learn the latent features of both preceding and following locations in trajectory and fuse the external features to reflect context aware mobility patterns, but also can identify and emphasize on the features that have strong correlations to destination.
- We further adopt a hierarchical deep learning model to explore meaningful mobility patterns in different spatial granularity, and then rationally fuse the patterns together to improve the accuracy of predicted results by its enhanced learning ability.
- By taking holiday events and weather conditions factors into consideration, the model adaptively fuses the result with these information by a learnable parameter matrix to obtain a more accurate prediction.
- We conduct extensive experiments with the real trajectory dataset of Beijing and Chengdu to evaluate the effectiveness and efficiency of these solutions. The experimental results demonstrate the advantages of our models.

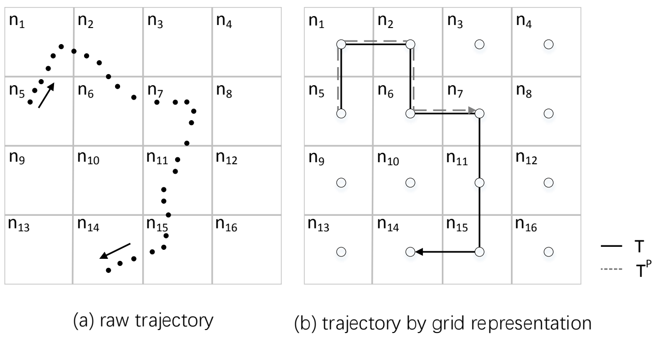
The remainder of this paper is organized as follows. Section 2 formally defines our problem. Section 3 derives a baseline algorithm, and Section 4 presents an attention-based LSTM model and a multi-granularity fusion framework to predict the destination of trajectory. Extensive experimental results are illustrated in Section 5 to demonstrate the performance of our proposed solutions. Section 6 and Section 7 discuss the related work and conclude the paper respectively.

2 PROBLEM STATEMENT

This section introduces some preliminaries and gives a formal statement of the problem this paper focuses on.

A raw trajectory is a sequence of sampling points collected by the global positioning system. Each sampling point has two coordinates formed by a latitude and a longitude. Figure 2 (a) shows the projection of a trajectory T in the geographical space. But obviously, it is not practical to predict the exact coordinates of the final destination in advance. Similar to

[23], we hereby adopt a grid based trajectory representation instead.



n_1 n_2 n_3 n_4 n_5 n_6 n_7 n_8 n_9 n_{10} n_{11} n_{12} n_{13} n_{14} n_{15} n_{16}
T
T^P
(a) raw trajectory
(b) trajectory by grid representation

Figure 2: Grid representation of trajectory

The geographical space is partitioned into a set of $g \times g$ grid cells, as shown in Figure 2. All the locations within the same cell are considered to be the same object. Each grid cell has the side length of 1 and adjacent cells have the step of 1. Thus, a *trajectory* can be defined as:

Definition 1. (*Trajectory*) A trajectory $T = \{T_1, T_2, ..., T_i, T_{i+1}, ..., T_d\} (1 \leq i < d)$ is a sequence of grid cells that captures the movement of a user, where each trajectory point T_i is a grid cell that sees at least one sampling point of the raw trajectory. We use T_d to denote the destination of trajectory T.

All trajectories are represented in grid granularity in the rest of the paper. Obviously, any two consecutive trajectory points T_i and T_{i+1} must be located in adjacent grid cells.

Example 2: Given the raw trajectory in Figure 2 (a), we can derive the cell sequence $\{n_5, n_1, ..., n_{15}, n_{14}\}$ as its corresponding trajectory T (in grid representation), shown in Figure 2 (b). Obviously, we can see that $T_1 = n_5$, $T_2 = n_1$, and $T_d = n_{14}$. The destination of trajectory T is cell n_{14}.

Definition 2. (*Prefix trajectory*) Given a trajectory T, a sub-trajectory $T^P = \{T^P_1, ..., T^P_c\}$ of it is said to be the prefix of T, if it satisfies the following two conditions (1) the sequences of T^P and T start from the same cell; (2) the cell sequence of T^P is fully contained by the cell sequence of T, such that $T^P \subset T$. The last point in sub-trajectory T^P, denoted as T^P_c, is the current location of the user.

Example 3: In Figure 2 (b), $T^P = \{n_5, n_1, n_2, n_6, n_7\}$ is the sub-trajectory of T that we have seen at a particular time: they start from the same cell n_5, and T^P is fully contained by T.

Assume that T^P is the observed sub-trajectory and n_7 is the location currently located. To recommend the user helpful location-aware message, the LBS system is required to estimate the final destination T_d of its trajectory T in advance. Next, we formalize the problem of the paper.

Problem formalization. Given a trajectory dataset D, the prefix of a trajectory T^P, we would like to return top-k cells with the highest probabilities being the destination cell T_d based on T^P through some prediction models.

3 BASELINE ALGORITHM

In this section, we introduce the *Sub-Trajectory Synthesis(SubSyn)* algorithm proposed in [23] to solve the destination prediction problem. It follows a general Bayesian inference framework, and applies Bayes' rule as the prediction tool. Bayes' rule contains a prior probability and a posterior probability, the algorithm mainly focuses on the calculation of the posterior probability $P(T^P|d \in n_j)$, since the prior probability $P(d \in n_j)$ could be easily obtained.

In the training phase, the algorithm applies the Markov model to accomplish the preparation of probabilities that contribute to the computation of $P(T^P|d \in n_j)$. These necessary probabilities are stored in two two-dimensional $g^2 \times g^2$ probability matrices: a *transition matrix* M and a *total transition matrix* M^T. Each element p_{ij} in matrix M indicates the ratio of trajectories at cell n_i that will turn to its adjacent cell n_j. The algorithm makes M multiplied r times by itself to find the sum of transition probabilities through various paths within r steps for two cells that are not adjacent, for example, n_i and n_k. Since there are many possible paths from n_i to n_k, the algorithm limits the path within a typical value of steps, and the total transition probabilities between n_i and n_k can be obtained and stored in M^T.

During the prediction phase, when the prefix of a trajectory T^P is given, the destination of T^P will be analysed and answered online. The algorithm introduces a *path probability* that considers the prefix of one trajectory as a synthesis of adjacent cells. It is the probability of a person travelling from one location to another via a specific path. After we get the matrix M^T and the path probability $P(T^P)$, the posterior probability of a user travelling from the starting cell to the current cell via T^P conditioned on the destination being the cell n_j can be obtained. Consequently, the probability of each cell n_j being the destination conditioned on the prefix of one trajectory T^P in Bayes' rule can be computed. When a user issues a query, the cells are sorted according to their destination probabilities $P(d \in n_j|T^P)$, the top-k elements in the sorted list are returned as the most likely destinations.

Discussion. However, traditional Markov model approaches have defects when facing fine-granularity prediction. First, each trajectory corresponds to a longer sequence (of find-grained spatial grids) accordingly, and data sparsity problem becomes more significant. Besides, since each state completely depends on the previous state in the Markov model, the prediction of next trajectory point considers previous trajectory point only, while ignoring its correlations to all other locations that have passed. This severely limits the accuracy of the Sub-Trajectory Synthesis algorithm, especially for fine-grained prediction. We thus propose more effective models to improve accuracy.

4 SOLUTIONS

In this section, we introduce two well-designed approaches to achieve accurate prediction of user destination. A model that integrates the attention mechanism and the LSTM network is described in Section 4.1. Furthermore, a hierarchical learning

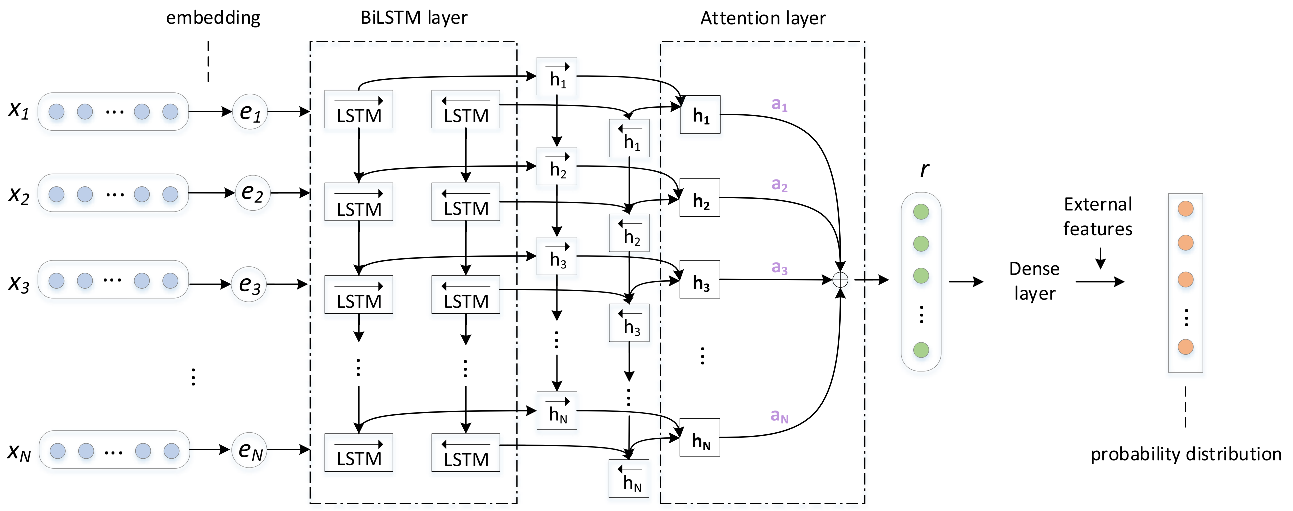


Figure 3: Trajectory based Attentional LSTM Learning model (TALL model)

model that achieves superior learning ability by exploring and merging meaningful features in different spatial granularity is introduced in Section 4.2.

4.1 Trajectory based Attentional LSTM Learning model

Overview. For our destination prediction problem, using LSTM network directly is unlikely to be satisfactory. Firstly, the standard LSTM model is insufficient to support accurate fine-grained prediction. Besides, the strong correlations may be missed out if the locations are far from each other in the trajectory sequence, and some important locations that have strong correlations to destination must be given more attention. Therefore, we first propose a *Trajectory based Attentional LSTM Learning model* (TALL model in short), as shown in Figure 3, to solve this problem.

The TALL model first uses the embedding technique to convert the training trajectory sequences into the form the network can take. Then, a BiLSTM layer is adopted to learn the latent features of both the preceding and following locations in the trajectory, and take some external features into consideration at the same time. Towards the output vectors of BiLSTM layer, the attention mechanism automatically leads more attentions to be paid to more meaningful passed locations in trajectory that have strong correlations to destination. At last, a softmax function is used to get the probability distribution of each location being the destination.

Embedding Layer. Given an observed prefix trajectory $T^P = \{x_1, x_2, \cdots, x_N\}$ that contains N grid cells, the first step of the model is to convert each grid cell x_i into a real-valued vector e_i. Then the embedded prefix trajectory is transformed into a two-dimensional vector, which is denoted as $emb_s = \{e_1, e_2, \cdots, e_N\}$. As shown in the embedding part of Figure 3, the vector emb_s is the input vector of the deep neural network.

BiLSTM Layer. In modeling trajectory sequential patterns, a method could be directly applying the LSTM network. It improves standard RNN by handling the gradient vanishing and exploding phenomena through the gating mechanism, and is thus more suitable for our problem (i.e. longer trajectory sequence in fine-grained prediction). Unfortunately, LSTM network generally ignores the future context in the sequence of trajectory (i.e. captures preceding part of the trajectory only), while in contrast, it is supposed to preserve meaningful latent characteristics of the whole trajectory. As such, this paper adopts a bidirectional LSTM network to take both left and right sequence context into consideration by forward and backward pass respectively.

A bidirectional LSTM network consists of forward and backward LSTM's, as shown in the BiLSTM layer part of Figure 3. We adopt a variant of model in [6] as the forward LSTM, which updates its hidden state as well as an LSTM cell state c for any time step t by the following equations:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
\tilde{c}_t &= tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \circ tanh(c_t)
\end{aligned} \tag{1}$$

where $\circ$ denotes the element-wise product between vectors, $tanh$ is the hyperbolic tangent function, and σ is the sigmoid activation function. The output value of σ is limited between 0 and 1, which plays the role of controlling the information flow in the network. i, f, o, c, and $\tilde{c}$ respectively represent the *input gate*, the *forget gate*, the *output gate*, the *cell state* and the *cell input* activation vectors, all of which are of the same size as the hidden vector h. These gates are set to generate some degrees, using current input x_t, the hidden state h_{t-1} generated in previous step, and current state of this cell c_{t-1}

respectively, to forget the memory stored before, and output the state generated later.

As the bidirectional LSTM network extends the network by introducing a second layer, where the hidden to hidden connections flow in opposite temporal order. The model is therefore able to exploit the contexts from both the past and the future. The output of the i^{th} location is shown in the following equation:

$$\mathbf{h_i} = [\overrightarrow{h}_i \oplus \overleftarrow{h}_i] \tag{2}$$

In this way, the annotation $\mathbf{h_i}$ summarizes the information of both the preceding locations and the following locations. Due to the tendency of LSTMs to better represent recent input, the annotation $\mathbf{h_i}$ will be focused on the locations around x_i, in order to assign different weights to different locations.

Attention Layer. So far, the contributions of feature vectors generated in the bidirectional LSTM network depend on their locality in trajectory: features of latter part in trajectory have higher weights than others. But in reality, more attention should be given to those locations that have strong correlations to the destination, regardless of their locality in trajectory. For example, if a sub-trajectory starts from a campus of a university and moves toward another campus of this university, then this campus has high probability to be the destination of the user. In light of this, the TALL model seeks to incorporate the attention mechanism by adding an attention layer. The functionality of this layer is to identify the features of locations that have decisive effect on the last probability distribution of destination, and thus pays more attention to these features by adjusting the weights reasonably.

After the iteration in LSTM layer, let H be a matrix consisting of the output vectors $[h_1, h_2, \cdots, h_N]$ that the LSTM layer produced, where N is the length of the prefix of one trajectory. The representation r of the prefix is formed by a weighted sum of these output vectors:

$$\begin{aligned} m_i &= tanh(W_h h_i + b_h), \quad m_i \in [-1, 1] \\ a_i &= \frac{exp(m_i)}{\sum_{t=1}^{N} exp(m_t)}, \quad \sum_{i=1}^{N} a_i = 1 \\ r &= \sum_{i=1}^{N} a_i h_i, \quad r \in \mathbb{R}^{d^w} \end{aligned} \tag{3}$$

where W_h and b_h are the attention layer's weights and bias, optimized during training to assign bigger weights to the most important locations of a prefix of one trajectory. d^w is the dimension of the location vectors, w is a trained parameter vector. As shown in the attention layer in Figure 3, a_i is the weight that determines the effects of different locations in the prefix of one trajectory on the probability distribution of being the destination. The dimensions of a_i, r are N, d^w respectively. We obtain the final location-pair representation used for classification from:

$$h^* = tanh(r) \tag{4}$$

External Features. In this part, we introduce how to take the external features into consideration. The external features such as weather conditions, holidays or big events may have significant impact on the variety of human mobility patterns. For example, people are more likely to go to office buildings on weekdays, and to commercial areas on weekends or there are big events. Furthermore, different weather conditions may affect people's final destinations: people are more inclined to go to the uptown or some rest areas in rainy days, but some sightseeing spots in sunny days. Consequently, the prediction accuracy of the destination will be affected. Thus the consideration of the external features is obviously important.

Formally, we use two fully connected layers to extract the information of external features. The first layer can be seen as an embedding layer for each factor, and the second layer is used to map from low to high dimension to get Y_E. Thus, Y_E contains the useful features of the external factors. In our experiment, we mainly incorporate holidays and weather conditions into the model. The holidays can be directly obtained, and we use forecasting weather information to obtain the unknown weather conditions.

Predicting Layer. A dense layer is used as the last output layer of the model, and a multi-class logistic regression is adopted to get the distribution of the destination label $\hat{y}$ from a discrete set of classes G. G represents all cells in the grid. w^E is a learnable parameters of external features. The classifier takes the hidden state h^* and Y_E as inputs and applies an affine transformation with weights W^{T^P}, W^E and biases b^{T^P} :

$$\begin{aligned} \hat{p}(y|T^P) &= softmax(W^{T^P} h^* + W^E Y_E + b^{T^P}) \\ \hat{y} &= \arg\max_y \hat{p}(y|T^P). \end{aligned} \tag{5}$$

In the TALL model, we apply categorical_crossentropy as our loss function, which is the multi-class logarithmic loss function frequently used corresponding to the softmax classifier. The cost function is the negative log-likelihood of the true destination label $\hat{y}$:

$$L(\theta) = -\frac{1}{m} \sum_{i=1}^{m} t_i \log(y_i) + \lambda \|\theta\|_F^2 \tag{6}$$

where $t_i \in \Re^m$ is the one-hot represented ground truth, m is the number of target destinations and λ is a regularization hyperparameter. We sort the probability distributions vector $\{y_1, y_2, \cdots, y_m\}$, where each element $y_i \in \Re^m$ is the estimated probability for each possible destination produced by softmax function. Due to the distance deviation problems in actual situation, we are inclined to judge whether the target destination of one sub-trajectory T^P is in the first k elements.

4.2 Hierarchical Trajectory based Attentional LSTM Learning model

In despite of its good learning ability, the TALL model utilizes the features and patterns at the prediction spatial granularity only. But as we know, human mobility patterns may appear in different spatial granularity, and it is necessary

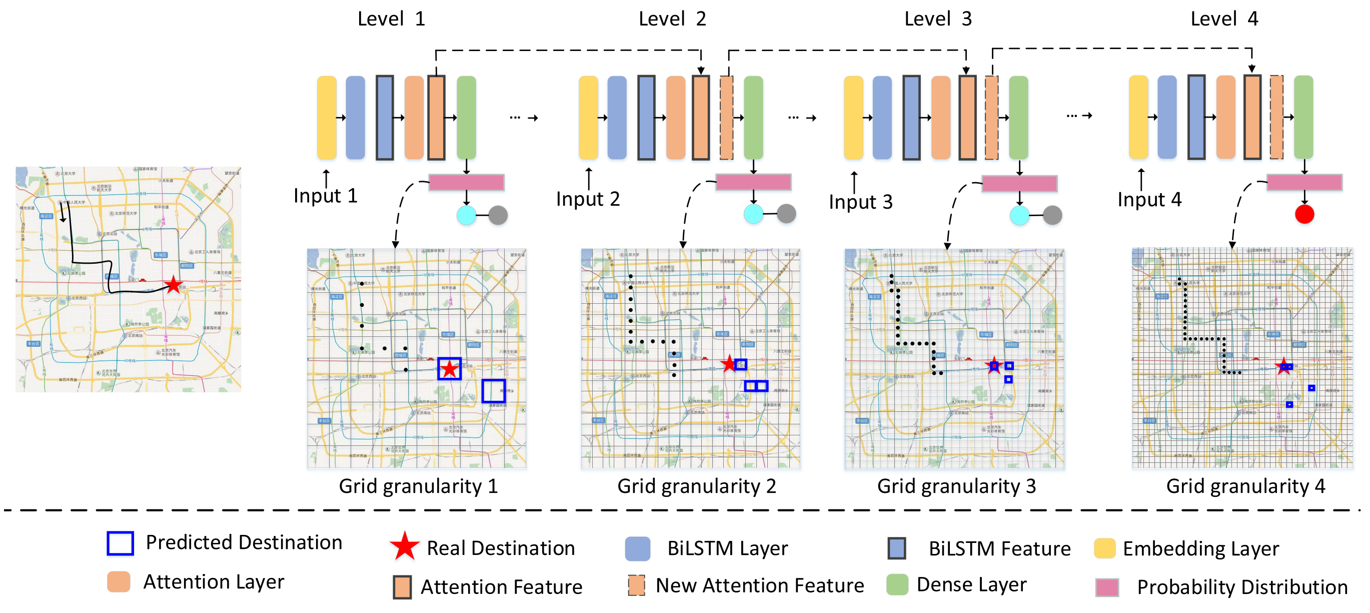


Figure 4: Hierarchical-Trajectory based Attentional LSTM Learning model (H-TALL model)

to integrate those patterns in different granularity to achieve accurate prediction. In this section, we further propose a hierarchical model learning called H-TALL model, which has the advantage of better understanding of mobility patterns in multiple spatial granularity.

Overview. The basic idea of the H-TALL model is to merge the implicit features learned from multiple spatial granularity rationally, so as to realize enhanced learning ability for improving accuracy of prediction. The problem is how to explore and manage the multi-granularity information, including the hierarchical spatial partition, learning at each granularity, as well as their integration to derive multi-granularity mobility patterns, and finally obtain accurate prediction result at the targeted spatial granularity.

An overview of this model is illustrated in Figure 4, which is an example of the region within the fourth ring road of Beijing, and one schematic trajectory is chosen as the prediction trajectory. Such an H-TALL model is composed of M levels(M=4 in Figure 4) and each level contains a prediction model on top of a grid granularity. As these grids are independent from each other, when the prefix of one trajectory is given, prediction models in each level will carry out the predicting process in the order of grid granularity from coarse to fine. Since there are different sequence representation for one raw trajectory, these cell sequences are taken as the input sequences of the prediction models in different levels. Detailedly, after processing the input sequence under previous m (m=1,2,3,...,M) grid granularities, the location-pair representation of the attention layer in (m+1)-th level summarizes the features of all past input sequences to a new location-pair representation. This representation will subsequently fed to the predicting layer in (m+1)-th level. Thus, top-k possible destinations can be obtained, and they are marked in the grid under the corresponding divisory granularity. To get the most accurate predicting result, we choose the top-k destinations in the finest granularity M as our final output. Three more detailed description are shown as follows:

Decompose Raw Trajectory. For a raw trajectory, we first decompose a spatial grid into cells by M different granularities, corresponding to the M divided grids in Figure 4. Thus the same raw trajectory will have distinguished cell sequence representations after it is transformed, denoted by $\mathbb{T}^1$, $\mathbb{T}^2$, ..., and $\mathbb{T}^M$ respectively. Notice that the length of the trajectory sequence that the same raw trajectory transformed does not change in the training phase since we do not delete duplicate (we use the same symbol to denote the raw trajectory points within the same cell) cell points in each trajectory sequence. Then the trajectory sequences are embedded into prediction models' inputs separately. Formally manifested by:

$$emb^m = (e_{m,1}, e_{m,2}, e_{m,3}, ..., e_{m,N}) \quad m = 1, 2, ..., M \quad (7)$$

where M is the number of levels in the model. N is the length of the trajectory sequence under different granularity. Thus the multi-inputs of the H-TALL model can be obtained, i.e., *input 1*, *input 2*, ..., *input M*.

Deep Neural Network Structure. Firstly, as each level of the model is still based on combination of bidirectional LSTM and attention mechanism, the biLSTM encoder takes as input the sequence of trajectory embeddings $(e_{m,1}, e_{m,2}, e_{m,3}, ..., e_{m,N})$ output from the embedding layer and computes the hidden state annotation h_n^m :

$$h_n^m = bilstm_{enc}^m(h_{n-1}^m, w_n^m) \quad n = 1, 2, 3, ..., N \quad (8)$$

where $bilstm_{enc}^m$ denotes the bidirectional LSTM function (equation (1) and (2)) for encoding the left and right sequence context, $h_n^m \in \mathcal{R}^D$ is the hidden state annotation and the initial time step $h_0^m = \{0\}$, w_n^m is the corresponding weight vector. The hidden state annotation h_n^m contains the summaries of both preceding locations and following locations in a trajectory sequence.

To continuously taking advantage of identifying and emphasizing on the features that have strong correlations to destination in each level of the model, we further apply an attention layer in each single prediction model. Formally:

$$\begin{aligned} r^m &= att_{enc}^m(h_n^m, a_n^m) \\ h^{m,*} &= tanh(r^m) \end{aligned} \tag{9}$$

where att_{enc}^m denotes the function of equations summarized in equation (3). r^m is formed by a weighted sum of vectors outputted in BiLSTM layer. h_n^m is the hidden state annotations under the m-th granularity. a_n^m is the weights representation that determines the effects on different locations in the trajectory sequence on the probability distribution of being the destination under the m-th granularity. The dimension of a_n^m is N, and r^m has the same dimension with the location vector, which contains a trained parameter vector. Here we still adopt a *tanh* activation function to obtain the final location-pair representation $h^{m,*}$ under the m-th granularity.

Subsequently, since the core of this model is the fusion of the features outputted by the attention layer in different levels, the prediction model of m-th granularity takes the current location-pair representation $h^{m,*}$ and the location-pair representation $h^{m-1,*}$ under (m-1)-th granularity which are both outputted by the attention layer to comprise a new representation. Formally:

$$h_{new}^{m,*} = h^{m,*} \oplus h^{m-1,*} \tag{10}$$

the new location-pair representation $h_{new}^{m,*}$ contains the implicit correlations of the same raw trajectory under previous m granularities by iteration and it will subsequently be fed to the predicting layer. $\oplus$ represents the element-wise sum function.

Prediction. As the H-TALL model has multi-outputs, different sequence representations of the same raw trajectory $\mathbb{T}^1$, $\mathbb{T}^2$, ..., and $\mathbb{T}^M$ have different corresponding output destination, denoted by $\mathbb{T}_d^1$, $\mathbb{T}_d^2$, ..., and $\mathbb{T}_d^M$ respectively. The predicting layer of i-th prediction model obtains the top-k cells with highest probabilities being the $\mathbb{T}_d^i$ of cell sequence $\mathbb{T}^i$ according to the *softmax* function (here we also incorporate the influence of external features) before the final probability distributions, then determines whether the cell marked by red star in the grid is within the top-k sets respectively. However, the finer granularity the spatial space decomposed by, the more lessened the prediction destination is within a small region. This will obviously improve the prediction accuracy. Thus in the testing phase of the model, we give the prefix of one trajectory with the finest granularity $\mathbb{T}^M$ to find out the real destination $\mathbb{T}_d^M$.

5 EXPERIMENTAL STUDY

In this section, we conduct extensive experiments on real trajectory datasets to evaluate the performance of the SubSyn algorithm and our proposed methods, including TALL model and H-TALL model.

5.1 Experiment Settings

Two datasets that we applied are introduced in section A, and the parameter description of our experiments is shown in section B.

A. Dataset

- **Beijing:** We mainly extracted the area within the fourth ring road of Beijing. The trajectory data was collected from 1st Mar. to 31st Jul. in 2016. There are more than 2 million trajectories covering the road network every day. About 12 important holidays and 24 weekends exist in the dataset. The data of the first 4 months are used as the training set, and the remaining 1 month as the test set.
- **Chengdu:** The trajectory data was collected from 1st Nov. to 30th Nov. in 2016, and published by Didi (Uber of China). There are more than 20 thousands trajectories can be obtained every day. There are 4 weekends in this month. The last 9 days are test set and the others are training set.

B. Parameter description

To evaluate the performance of our models on various user queries, we use the *prediction accuracy* as our measurement. It counts the number of the prefix of trajectory for which the real destination is within the predicted top-k results. The parameter k is determined by the number of predicted destinations that we set. For instance, when we examine top five predicted destinations, k is set to five. Furthermore, grid granularity and the length of given prefix of trajectory, which could be represented by the completed percentage of this trajectory, have obvious effects on the prediction accuracy. Consequently, the measurement will be evaluated against varying one parameter among three at each time. In experiments, we evaluate the performance by varying the *grid granularity* (g), *trip completed percentage* (tcp), and *top-k predicted destinations*, and their settings in experiments (default values: $k = 5$, $g = 50$, $tcp = 70\%$) are summarized in Table 1.

Parameter	Parameter values	Description
k	1, 2, 3, 4, **5**	top-k predicted destinations
g	10, 20, 30, 40, **50**	grid granularity
tcp(%)	10, 30, 50, **70**, 90	trip completed percentage

Table 1: Default Parameter Values

5.2 Performance Evaluation

In this section, we introduce the evaluation of effectiveness on the varying of three parameters mentioned above. Since the SubSyn algorithm does not take the external features into consideration, the experiments are divided into two parts consequently. In Section A, we detailedly described the comparision results of the SubSyn algorithm and our models without external features integrated in. The experimental results of our TALL model and H-TALL model that taken the external features into consideration are shown respectively in Section B.

A. Comparision without external features

Effect of top-k predicted destinations: We investigate the effect of the number of predicted destinations on the performance of SubSyn algorithm and our two proposed models. Since the number of predicted destinations may be insufficient, this metric takes this limitation into consideration for all methods in this experiment. Note that in this comparision experiment, the grid granularity and the trip completed percentage are default values.

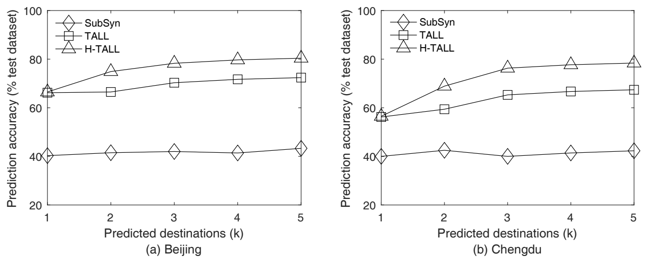


Figure 5: Varying the value of k

The experimental results are shown in Figure 5. Intuitively, for SubSyn algorithm, the data sparsity problem of fine grid granularity makes the value of k has no obvious effect on prediction accuracy both for the dataset of Beijing and Chengdu. The value has been stable at 40% as can be seen from the two corresponding polylines in the figure. However, due to the fact that H-TALL model could eliminate the characteristics of the impossible regions through the iteration between levels, with the increasing value of k, for both the test datasets, the prediction accuracies rise faster than the TALL model and SubSyn algorithm. Even if the value of k is 1, H-TALL still performs better. In a typical setting, $k = 5$, the prediction accuracy of H-TALL model is two times accurate than the SubSyn algorithm.

Effect of trip completed percentage: We also investigate the effect of trip completed percentage on the performance of these three algorithms. When the length of the given prefix of trajectory is not long enough, the probability of each cell in the grid being the destination will tend to be average. Consequently, the possible destinations will be distributed in multiple discrete area. On the contrary, since the growing length of the given prefix of trajectory will relieve this problem, the similar matching trajectories in the dataset will correspondingly decrease, and that will also cause the data sparsity problem. Therefore, we need to observe the impacts that this parameter brought on prediction accuracy, and find a suitable value for our datasets.

As shown in Figure 6, we vary the percentage from 10% to 90% with 20% increment. For the SubSyn algorithm, the prediction accuracies on two datasets have no obvious rising or falling trend, with the value being stable at 40%. It represents a balance due to the defaults which brought by the

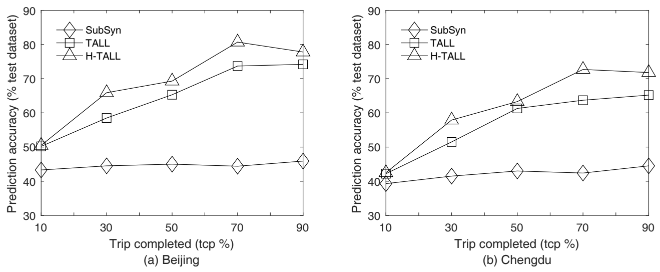


Figure 6: Varying the percentage of trip completed

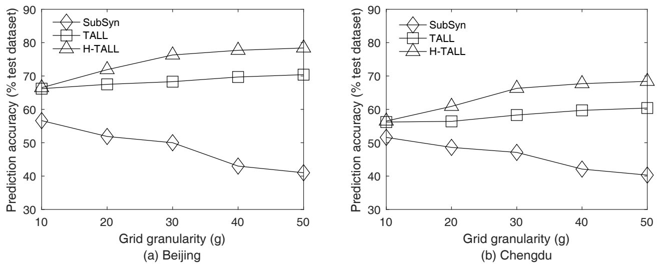


Figure 7: Varying the grid granularity g

length of the given sub-trajectory becomes longer. However, our TALL model and H-TALL model perform much better under the varying. With the increase of percentage value, the prediction accuracy grows with fast speed, and the accuracy could reach 80% when the trip completed percentage is our default value 70%. After this peak, the accuracies for both the TALL model and H-TALL model declined. This phenomenon proves our models could handle relatively long trajectories, because more features of the locations in the trajectory could be learned, and that will obviously contribute to the final prediction accuracy. Meanwhile, when the length of given prefix trajectory exceeds our default value of the entire trajectory, the impacts of data sparsity problem brought become large.

Effect of grid granularity: In this part, the divisory granularity of the grid has been varied to observe the effect of g for prediction accuracy on our datasets. When the grid is divided by a coarse granularity, the number of matching prefixes of trajectories will get higher as more trajectory points in the training dataset may fall into identical cell. A fine grid (e.g., g=50) has the advantage of higher prediction accuracy that small area of each cell brings. Nevertheless, it may cause the fine granularity sparsity problem we mentioned above. Since less trajectory points will lie in the a same cell, making the task of destination prediction becomes more difficult. Besides, fine granularity also has another drawback that it requires more time to complete the training phase, as the length of each trajectory becomes longer. Therefore, we need to test the effects of these two algorithms on the prediction accuracy when g is varying, and then determine a suitable value of granularity for our dataset. Note that, though the H-TALL model is a fusion model under multiple grid granularities, we use the identical value of grid granularity in other two models as our target granularity for prediction.

Figure 7 shows the trends of prediction accuracy in both Beijing and Chengdu dataset with respect to grid granularity. Here the top-k predicted destinations and trip completed percentage of trajectory are both the default values in Table 1. The prediction accuracy of SubSyn algorithm drops when the grid granularity becomes finer, that is due to the fine granularity data sparsity problem caused by smaller cell in the grid. However, our TALL model and H-TALL model still have much better prediction accuracy under fine divisory granularity, even with a rising trend. Besides, owing to the advantages of H-TALL model, it performs better than TALL model with varying g. Thus, we choose 50 to be the optimal grid granularity for our training datasets, according to the global maximum point in the figure. The value does not need to be modified often because the update of the datasets is rare. All the following experiments are done using the grid granularity $g = 50$.

B. Comparision with external features

As we know, under different weather conditions, holidays and big events, human mobility patterns may vary significantly. It will consequently has impacts on the prediction accuracy. Thus we need to take these factors into consideration. However, since the SubSyn algorithm does not consider about external features, we only make comparative experiments on our TALL model and H-TALL model in this section. There are both 10 types of weather conditions being divided for the dataset of Bejing and Chengdu. Temperature ranges from -4°C to 36°C in the dataset of Beijing and 2°C to 21°C for Chengdu. In this part, we still consider the varying of the three parameters mentioned above, and observe their impacts on the prediction accuracy. The experimental results are shown in Figure 8-10 as follows.

It can be seen that the prediction accuracy has improved overall under the varying of those three parameters in the following three figures. Specifically, though our models take relatively less external features into consideration, the improvements of the prediction accuracies are still evident to be seen. Besides, the prediction results for both TALL and H-TALL models on the dataset of Beijing outperform Chengdu, not only because the amount of trajectory data of Chengdu is smaller than Bejing, less external features have been integrated in is another reason that can not be ignored. In light of this, we could draw a conclusion that models that have considered external features outperform that without the features integrated in.

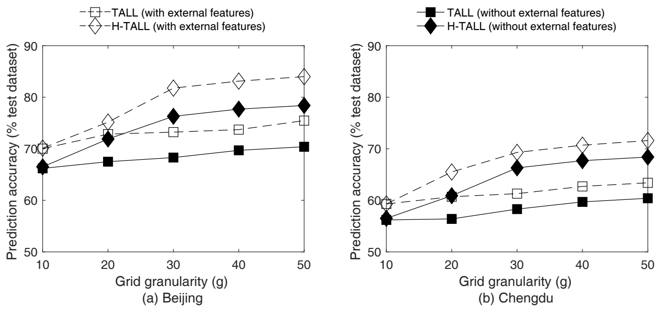


Figure 8: Varying the grid granularity g

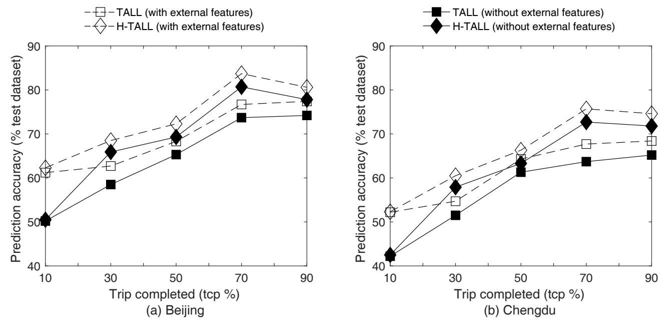


Figure 9: Varying the percentage of trip completed

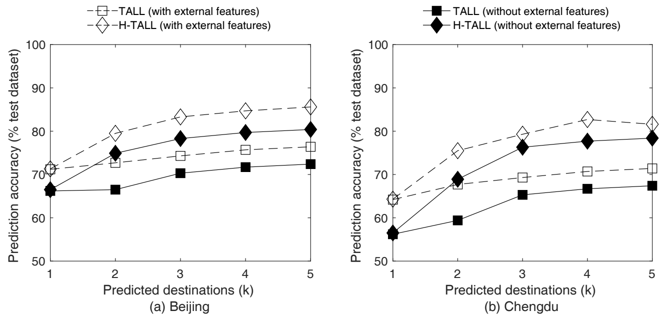


Figure 10: Varying the value of k

6 RELATED WORK

In this section, we survey the related work on available destination prediction methods and deep learning technology.

Destination Prediction. Many studies have been conducted to address the destination prediction problem[1, 2, 8, 16]. Among them, the widely used models for this problem are the Bayesian network and Markov model. They adopts the models to predict destinations for specific individuals based on the modes of historical transport, and take the external information into consideration, such as length of trajectory, trip time, and the distribution of different districts[7, 10]. Similarly, a method based on the analysis of distance called NNT is proposed to ensure the accuracy[20]. In addition to these well-known models that could be directly applied in our problem, some other effective methods have also proven to be effective. For example, [4] proposes a method based on the analysis of real time moving patterns of users, and [19] utilises the driving contexts and contexts information. All of the above methods have their specific using situations, and have good prediction results.

Deep Learning. In recent years, with the rapid development of deep learning technique, more and more researchers begin to adopt it for various problems[6, 18, 26]. Specifically, many applications use RNNs with the combination of attention mechanism [11, 17]. In the light of this, a lot of new applications have utilized RNNs to solve various problems on trajectory data. For example, [22] has designed two RNN-based models to model trajectories based on unique topological constraints, [13] used a bidirectional LSTM model to analyze historical traffic flow data of prediction point to get the traffic flow periodicity feature, and [15] proposed an LC-RNN model which integrates CNN and RNN models to achieve accurate traffic speed prediction. Besides, an efficient vehicle trajectory prediction framework has been applied in [3], it employs the LSTM network to analyse the temporal behavior, and predicts the future coordinate of the surrounding vehicles. Particularly, [5] proposed an attentional recurrent network which called DeepMove, for mobility prediction from lengthy and sparse trajectories to achieve the prediction of human mobility, that was similar to our problem. Furthermore, [25] proposed a DMVST-Net framework which utilises both the convolutional neural networks and the LSTM networks to model the spatial and temporal relations in taxi demand prediction, and [24] proposed a novel STDN framework which involves a flow gating mechanism and a periodically shifted attention mechanism to solve the same problem. However, as far as we know, there is no deep learning based destination prediction model yet. It is thus necessary to design deep neural network models that can support multi-granularity trajectory modelling for accurate prediction of user destination.

7 CONCLUSION AND FUTURE WORK

This paper has studied the problem of destination prediction. A TALL-model has been first proposed to solve the data sparsity problem in the fine-grained prediction. It integrates the bidirectional LSTM network and attention mechanism to learn meaningful latent features to achieve accurate prediction. In addition, a hierarchical model called H-TALL is introduced afterwards to explore and merge the mobility patterns learned from multiple spatial granularities, so that the accuracy of prediction can be further improved. Extensive experimental results on real datasets demonstrate the effectiveness of our proposed methods.

In the future, it would be interesting to investigate more advanced models with additional external features to be considered.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61872258, 61572335, 61472263, 61772356, 61876117, and 61802273, the Australian Research Council discovery projects under grant numbers DP160102412, DP170104747, DP180100212, and the Open Program of State Key Laboratory of Software Architecture under item number SKLSAOP1801.

REFERENCES

[1] J. A. Alvarez-Garcia, J. A. Ortega, L. Gonzalez-Abril, and F. Velasco. 2010. Trip destination prediction based on past GPS log using a Hidden Markov Model. *Expert Systems with Applications* 37, 12 (2010), 8166–8171.

[2] Graeme Best and Robert Fitch. 2015. Bayesian intention inference for trajectory prediction with an unknown goal destination. In *IROS*. 5817–5823.

[3] Jaekyum Kim Seung-Hi Lee Chung Choo Chung Jun Won Choi Byeoungdo Kim, Chang Mook Kang. 2017. Probabilistic Vehicle Trajectory Prediction over Occupancy Grid Map via Recurrent Neural Network. In *ITSC*. 399–404.

[4] Zhenjiang Dong, Jia Deng, Xiaohui Jiang, and Yongli Wang. 2017. RTMatch: Real-Time Location Prediction Based on Trajectory Pattern Matching. In *DASFAA*. 103–117.

[5] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *WWW'18*. 1459–1468.

[6] A. Graves and N. Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*. 1764–1772.

[7] Eric Horvitz and John Krumm. 2012. Some help on the way:opportunistic routing under uncertainty. In *Ubicomp*. 371–380.

[8] Kai Jiang, Nenghai Yu, and Weihai Li. 2013. Online travel destination recommendation with efficient variable memory Markov model. In *ICME*. 1–4.

[9] Jiren Jin and Hideki Nakayama. 2017. Annotation order matters: Recurrent Image Annotator for arbitrary length image tagging. In *ICPR*. 2452–2457.

[10] John Krumm and Eric Horvitz. 2006. Predestination: Inferring Destinations from Partial Trajectories. In *Ubicomp*. 243–260.

[11] Xuelong Li, Bin Zhao, and Xiaoqiang Lu. 2017. MAM-RNN: Multi-level Attention Model Based RNN for Video Captioning. In *IJCAI*. 2208–2214.

[12] Huiwen Liu, Jiajie Xu, Kai Zheng, Chengfei Liu, Lan Du, and Xian Wu. 2017. Semantic-aware Query Processing for Activity Trajectories. In *WSDM*. 283–292.

[13] Yipeng Liu, Haifeng Zheng, Xinxin Feng, and Zhonghui Chen. 2017. Short-term traffic flow prediction with Conv-LSTM. In *WCSP*. 1–6.

[14] Zhongjian Lv, Jiajie Xu, Pengpeng Zhao, Guanfeng Liu, Lei Zhao, and Xiaofang Zhou. 2017. Outlier Trajectory Detection: A Trajectory Analytics Based Approach. In *DASFAA*. 231–246.

[15] Zhongjian Lv, Jiajie Xu, Kai Zheng, Hongzhi Yin, Pengpeng Zhao, and Xiaofang Zhou. 2018. LC-RNN: A Deep Learning Model for Traffic Speed Prediction. In *IJCAI*. 3470–3476.

[16] Donald J. Patterson, Lin Liao, Dieter Fox, and Henry Kautz. 2003. Inferring High-Level Behavior from Low-Level Sensors. In *Ubicomp*. 73–89.

[17] Zhaochun Ren Furu Wei-Jun Ma Maarten de Rijke Pengjie Ren, Zhumin Chen. 2017. Leveraging Contextual Sentence Relations for Extractive Summarization Using a Neural Attention Model. In *SIGIR*. 95–104.

[18] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*. 3104–3112.

[19] Kohei Tanaka, Yasue Kishino, Tsutomu Terada, and Shojiro Nishio. 2009. A destination prediction method using driving contexts and trajectory for car navigation systems. In *ACM SAC*. 190–195.

[20] Dalia Tiesyte and Christian S. Jensen. 2008. Similarity-based prediction of travel times for vehicles traveling on known routes. In *SIGSPATIAL*. 14.

[21] Chao Weng, Dong Yu, Shinji Watanabe, and Biing Hwang Fred Juang. 2014. Recurrent deep neural networks for robust speech recognition. In *ICASSP*. 5532–5536.

[22] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. 2017. Modeling Trajectories with Recurrent Neural Networks. In *IJCAI*. 3083–3090.

[23] Andy Yuan Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. 2013. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *ICDE*. 254–265.

[24] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, Yanwei Yu, and Zhenhui Li. 2018. Modeling Spatial-Temporal Dynamics for Traffic Prediction. *arXiv preprint arXiv:1803.01254* (2018).

[25] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. *arXiv preprint arXiv:1802.08714* (2018).

[26] Heng Yu and Xuan Zhu. 2015. Recurrent Neural Network based Rule Sequence Model for Statistical Machine Translation. In *ACL*. 132–138.