Predicting Destinations by a Deep Learning based Approach

Jiajie Xu, Jing Zhao, Rui Zhou, Chengfei Liu, *Member, IEEE*, Pengpeng Zhao, and Lei Zhao

Abstract—Destination prediction is known as an important problem for many location based services (LBSs). Existing solutions generally apply probabilistic models or neural network models to predict destinations over a subtrajectory, and adopt the standard attention mechanism to improve the prediction accuracy. However, the standard attention mechanism uses fixed feature representations, and has a limited ability to represent distinct features of locations. Besides, existing methods rarely take the impact of spatial and temporal characteristics of the trajectory into account. Their accuracies in fine-granularity prediction are always not satisfactory due to the data sparsity problem. Thus, in this paper, a carefully designed deep learning model called LATL model is presented. It not only adopts an adaptive attention network to model the distinct features of locations, but also implements time gates and distance gates into the Long Short-Term Memory (LSTM) network to capture the spatial-temporal relation between consecutive locations. Furthermore, to better understand the mobility patterns in different spatial granularities, and explore the fusion of multi-granularity learning capability, a hierarchical model that utilizes tailored combination of different neural networks under multiple spatial granularities is further proposed. Extensive empirical studies verify that the newly proposed models perform effectively and settle the problem nicely.

Index Terms—Trajectory prediction, trajectory embedding, deep learning

1 Introduction

The widespread use of smart phones and in-car navigation systems has become part of people's 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 (business, healthcare, and work, etc.) [1], [2]. *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 location-based services 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 setting destinations in navigation systems.

With large scale user trajectories available nowadays, accurate destination prediction can be achieved by understanding user mobility patterns from trajectory datasets. 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 for the user. However, we often end up with finding no or very few trajectories in dataset that can cover the given sub-trajectory, and this severely limits the accuracy and even feasibility for the prediction. This phenomenon also indicates that trajectory data is not fully used. It is thus important to explore the implicit mobility patterns from trajectory dataset. So far, many statistical approaches have been used to tackle this issue. They generally decompose raw trajectories into grid-based trajectories, and predict destination 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. However, the limited learning ability brings the concern that the statistical models can only support coarse-granularity prediction (i.e., with too large spatial grid) in most cases.

As shown in Fig. 1a, there are four historical trajectories: $t_1 = \{l_1, l_2, l_6, l_7, l_{11}, l_{12}\}$, $t_2 = \{l_1, l_2, l_6, l_{10}, l_{11}, l_7, l_8\}$, $t_3 = \{l_5, l_9, l_{10}, l_{11}, l_{15}\}$, 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 can 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 Fig. 1a 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 Fig. 1b, and the sub-trajectory $\{l_1, l_5, l_6, l_{10}\}$ will be transformed to $\{n_1, n_4\}$ in grid granularity. By

- *J. Xu is with the Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, Suzhou 215006, China, and also with Neusoft Corporation, Shenyang 110179, China. E-mail: xujj@suda.edu.cn.*
- *J. Zhao, P. Zhao, and L. Zhao are with the Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, Suzhou 215006, China. E-mail: 20175227005@stu.suda.edu.cn, {ppzhao, zhaol}@suda.edu.cn.*
- *R. Zhou and C. Liu are with the Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne, VIC 3011, Australia. E-mail: {rzhou, cliu}@swin.edu.au.*

Manuscript received 19 Dec. 2018; revised 19 June 2019; accepted 29 July 2019. Date of publication 5 Aug. 2019; date of current version 11 Jan. 2021. (Corresponding author: Jiajie Xu.)
Recommended for acceptance by Jeff M. Phillips.
Digital Object Identifier no. 10.1109/TKDE.2019.2932984

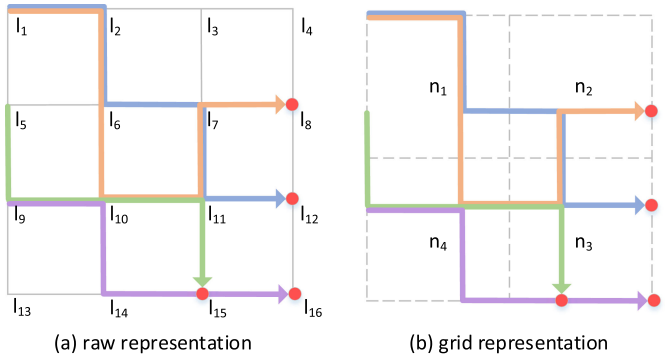


Fig. 1. An illustrative example of destination prediction.

applying [3], 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 of prediction in [3] 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. First, as [3] utilizes the grid representation, 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. Second, 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 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 big events, external features should be considered in the training process as well to improve the accuracy. More effective models are needed since no existing one can well address all above challenges as far as we know.

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 time series data. Empirical studies have also proved that *Long Short-Term Memory* (LSTM) network (one of the prominent variants of RNN) has comparable performance in modeling time series data with variable length, such as machine translation [4], image annotation [5] and speech recognition [6]. Meanwhile, involving the attention mechanism effectively improves the learning ability of the neural networks, and the success has been witnessed by many other problems [7], [8]. Since destination prediction relies on time series modeling (of trajectory), and the useful characteristics of different locations in trajectory should be given more attention, this has 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. In addition, the standard attention mechanism has a limited ability to represent distinct features of locations in the trajectory by fixed feature representations. Thus, a more advanced tailored model is required.

Although the work [2] has already addressed the data sparsity problem by adopting a bidirectional LSTM model to learn the latent features of both preceding and following locations in trajectory, it ignores the information of spatial-temporal intervals between consecutive locations, which is essential for modeling users' mobility patterns. Besides, the standard attention mechanism emphasizes the locations by fixed feature representations. That limits the ability of capturing distinct and useful features of different locations during the learning process, and leads to an insignificant accuracy improvement. Thus, to address these issues and obtain a higher prediction accuracy, in Sections 4.1 and 4.2, we design a novel Location-Aware Attention mechanism and a Trajectory-LSTM block respectively. Unlike the standard attention mechanism, the Location-Aware Attention mechanism dynamically adapts to each location in trajectory by a non-fixed feature representation. The Trajectory-LSTM block utilizes time gates and distance gates to model the long-term and short-term effects of previous locations on the current location concurrently. Furthermore, by integrating Location-Aware Attention mechanism and the Trajectory-LSTM block rationally, an LATL model which solves the data sparsity problem in fine granularity prediction is further proposed in Section 4.3.

Another limitation of the previous study [2] is that, similar prediction models are used in the hierarchical framework to combine the meaningful patterns under each spatial granularity. Although this method works effectively, in order to achieve a better prediction result, the prediction models under different granularities should be designed for specific characteristics of trajectory sequences. For example, the sequence representation of trajectory in finer granularity is usually longer, we can first use a dimensionality reduction layer to improve the training speed of the model. But this operation is not required under the coarse granularity. In the light of this, in Section 4.4 of this paper, we re-design the hierarchical learning model which has the advantage of adopting more tailored combination of neural network layers under each granularity. This makes the method better understand the mobility patterns in multiple spatial granularities, and improve the prediction accuracy effectively.

In summary, our new technical contributions in this extension work are four folds.

- We design a Location-Aware Attention mechanism which can dynamically adapt to locations locally in the trajectory, and capture the distinct features of locations by non-fixed feature representations to improve the prediction accuracy. Besides, we also design a Trajectory-LSTM block which utilizes time gates and distance gates to model the spatial-temporal intervals between consecutive locations.
- An LATL model which incorporates the Location-Aware Attention mechanism and the Trajectory-LSTM block is proposed to solve the data sparsity problem in fine granularity. The model can not only well capture the different features of locations adaptively during the learning process, but also model the long-term and short-term effects of passed locations effectively.

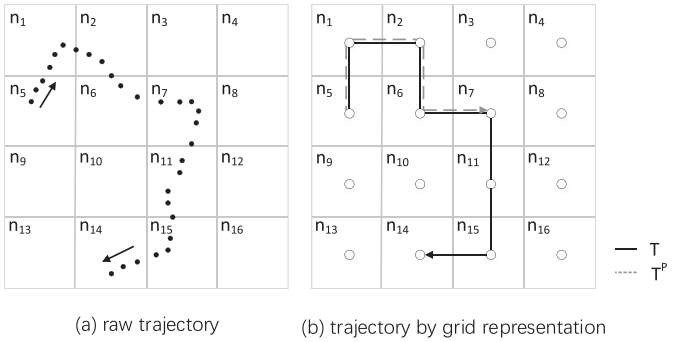


Fig. 2. Grid representation of trajectory.

- We carefully re-design a hierarchical learning model H-LATL by adopting more tailored combination of neural network layers under each spatial granularity to improve the prediction accuracy.
- Extensive comparative experiments on the real trajectory datasets of Beijing and Chengdu are conducted to evaluate the effectiveness and efficiency of our newly proposed models.

The remainder of this paper is organized as follows. Section 2 gives some preliminaries and formally defines our problem. Section 3 introduces the baseline algorithms, and Section 4 presents our extension, a Location-Aware Attentional Trajectory-LSTM model and a multi-granularity fusion framework to predict the destination of trajectory. Extensive experimental results are illustrated in Section 5 to show the performance of our extension work. Sections 6 and 7 discuss the related work and conclude the paper respectively.

2 Preliminaries

In this section, we first give the formal statement of the problem this paper focuses on, and then briefly introduce the standard LSTM used in [2].

2.1 Problem Formulation

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. Fig. 2a 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 [3], we hereby adopt a grid based trajectory representation instead.

The geographical space is partitioned into a set of $g \times g$ grid cells, as shown in Fig. 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 $T = \{T_1, T_2, \ldots, T_i, T_{i+1}, \ldots, T_d\} (1 \leq i < d)$ 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.

TABLE 1
Summary of Notations

Notation	Description
g	Granularity of a grid
D	The historical trajectory dataset
T	Trajectories in D
T^P	The prefix of T from T_1^P to T_c^P
T_i, T_i^P	The ith grid cell in trajectory T and T^P
T_c^P	The current/last grid cell in T^P
T_d	The destination cell of T

Example 1. Given the raw trajectory in Fig. 2a, we can derive the cell sequence $\{n_5, n_1, \ldots, n_{15}, n_{14}\}$ as its corresponding trajectory T (in grid representation), shown in Fig. 2b. 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 $T^P = \{T_1^P, \ldots, T_c^P\}$ 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_c^P, is the current location of the user.*

Example 2. In Fig. 2b, $T^P = \{n_5, n_1, n_2, n_6, n_7\}$ 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 useful location-aware message, the LBS systems are 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.

Table 1 lists the notations used throughout this paper.

2.2 LSTM

LSTM [9], a variant of RNN, is capable of learning long and short-term dependencies. It has become an effective and scalable model for sequential prediction problems, and many improvements have been made to the original LSTM architecture. In this paper, we extend it to another variant. For the concise and general purpose, here we introduce the basic update equations of LSTM block used in [2]

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$\hat{c}_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \quad (4)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (5)$$

$$h_t = o_t \odot tanh(c_t), \quad (6)$$

where i_t, f_t, o_t represent the input, forget and output gates of the tth object, decide what information to store, forget and output respectively. c_t is the cell activation vector representing cell state. x_t and h_t represent the input feature

vector and the hidden output vector, respectively. σ is the sigmoid activation function to map the values between 0 to 1, which plays the role of controlling the information flow in the network. $tanh$ is the hyperbolic tangent function. The update equation of c_t has two parts, one is a fraction of the previous cell state c_{t-1}, which is controlled by forget gate f_t, and the other is a new input state created from element-wise product, denoted by $\odot$, of i_t and the output of the $\hat{c}_t$.

3 Baseline Approaches

This section presents four approaches for solving the destination prediction problem. Section 3.1 details a *Sub-Trajectory Synthesis* (SubSyn) algorithm, which is a probabilistic model proposed in [3]. Section 3.2 focuses on neural network models. The first is the *Trajectory based Attentional LSTM Learning* (TALL) model, a deep learning based approach proposed in our previous work [2]. Then, the *Spatial-Temporal Long Short-Term Memory* (ST-LSTM) model proposed in [10], it combines spatial-temporal influence into LSTM to mitigate the problem of data sparsity. Finally, the MLP (Clustering) model which integrates the multi-perceptron architecture and a mean-shift clustering algorithm is introduced. Details are shown as follows.

3.1 Sub-Trajectory Synthesis Algorithm

Sub-Trajectory Synthesis algorithm generally follows a Bayesian inference framework. It utilizes a Markov model to establish the transfer relationship between locations, and applies the Bayes' rule as the prediction tool. The Bayes' rule contains a prior probability and a posterior probability. Since the prior probability could be easily obtained, the algorithm focuses on the calculation of the posterior probability. Consequently, this approach has two phases, a training phase and a prediction phase.

In the training phase, the algorithm constructs the Markov model by associating a state to each cell in the grid, thus two directed transitions of states corresponding to a pair of adjacent cells in the trajectory are established. The necessary probabilities are obtained and 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 then makes M multiply itself by r times to find the sum of transition probabilities through various paths within r steps for two cells n_i and n_k that are not adjacent, and stores the total transition probabilities in M^T.

During the prediction phase, when the prefix of a trajectory, T^P, is given, the algorithm introduces a path probability. It is the probability of a user travelling from one location to another via a specific path. The value can be obtained through multiplying the transition probabilities between all pairs of adjacent cells in T^P. After we get the total transition matrix M^T and the path probability, 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 computed. Consequently, we compute the probability of each cell being the destination conditioned on the prefix trajectory T^P in Bayes' rule. When a user issues a query, the cells are sorted according to their destination probabilities, and the top-k elements in the sorted list are returned as the most likely destinations.

3.2 Neural Network Models

3.2.1 Trajectory Based Attentional LSTM Learning Model

Trajectory based Attentional LSTM Learning (TALL) model is a deep learning based method that integrates the bidirectional LSTM network and the standard attention mechanism. It solves the data sparsity problem in fine granularity, and has a good prediction result. Furthermore, the hierarchical extension model (H-TALL) proposed in [2] has superior learning ability by exploring and merging meaningful features in different spatial granularities. These two models are introduced in details below.

First, the TALL model has five main parts. It contains an embedding layer, a bidirectional LSTM layer, an attention layer, the fusion of external features and a prediction layer. The embedding layer converts the training trajectory sequences into the form that the neural networks can accept. That means each cell is converted into a real-valued vector, and each embedded prefix trajectory is transformed into a two-dimensional vector. Then, the model adopts the bidirectional LSTM network which consists of a forward and a backward LSTM layer to learn both the preceding and following locations in trajectory sequence. This ensures the correlations of all locations that the user have passed will not be ignored. Due to the tendency of LSTM better representing recent input, the output annotation focuses on features around the current location. Here the adopted forward LSTM is a variant model in [11].

In addition, there are some locations have strong correlations to the final destination. They should be given more attention regardless of their locality in the sequence. In order to achieve this goal, the TALL model incorporates the standard attention mechanism to identify the features of these locations, and gives them more weights. Furthermore, human mobility patterns vary significantly under different weather conditions, holidays, big events and etc. The TALL model utilizes two fully connected layers to extract these external features. The represented feature information can easily be fused into a dense layer, which is the last output layer of the TALL model. Finally, a multi-class logistic regression is adopted to get the probability distribution of all cells in the grid. After sorting the values, top-k grid cells with the highest value are returned as the most possible destinations.

However, in despite of the good learning ability of TALL model, it captures the mobility patterns at one prediction spatial granularity only. The useful human mobility patterns may appear in different spatial granularities, and it is necessary to integrate those patterns to gain a more accurate prediction result. Consequently, H-TALL model divides the same research region with M different spatial granularities, and employs one prediction model under each granularity. Each prediction model is a variant of TALL model. By connecting the hidden layers in prediction models from coarse to fine granularity, H-TALL model excludes some impossible predicted destinations under the previous granularity. Obviously, to get the most accurate prediction results, H-

TALL model chooses the top-k grid cells with the highest value in probability distribution vector under the finest granularity. Thus, the goal of improving the accuracy can be achieved.

3.2.2 Spatial-Temporal Long Short-Term Memory Model

Spatial-Temporal Long Short-Term Memory (ST-LSTM) model is a neural network model proposed in [10]. It considers the length of time and distance intervals, and combines the influence into the gating mechanism of the LSTM network by an efficient add operation. It addresses the problem of predicting the next move people will go ahead in minutes or hours. Furthermore, the hierarchical extension model (HST-LSTM) models the periodicity of visit sequence, and boosts the prediction performance. Details are shown below.

First, modern cities consist of functional zones such as residential areas, business districts, and educational areas, etc. ST-LSTM model assumes the consecutive visits in the same zone represent the same socioeconomic activity. Then, the point trajectories are compressed into zone visiting sequences. The standard LSTM model is adopted as the basic structure. As we mentioned before, it has three gates to let the cell feed, update and forget information over time. However, it is always hard to learn well-functioned gates due to the sparse zone visiting sequence data. Thus, the ST-LSTM model adds the spatial-temporal factors into the three gates. Formally

$$k_t = \sigma(W_{xk}x_t + W_{hk}h_{t-1} + F_k(s_{t-1}, q_{t-1}) + b_k) \tag{7}$$

$$F_k(s_{t-1}, q_{t-1}) = W_{sk}s_{t-1} + W_{qk}q_{t-1}, \quad k = i, f, o, \tag{8}$$

where s_{t-1} and q_{t-1} represent the geographical distance and time interval respectively. Eq. (7) is the general function of three modified gates in LSTM. $F(\cdot)$ is the combination of the two factors, where W_{xk} and W_{hk} are linear transition matrices. Besides, the parameters are uncountable if to learn each possible time and distance interval. ST-LSTM model partitions the intervals into discrete slots and only encodes the upper and lower bound of these slots to solve the problem.

Based on ST-LSTM, the HST-LSTM model employs three steps to learn the periodicity of visit sequence and improve the performance of prediction. First, given a user and his/her visit record, which consists of visit sessions, the model uses the one-hot representation for each possible visit location, and encodes the visit sessions for the user. Then, the visit sessions are fed into a global contextual LSTM to model the long-term evolution of the user's visit sequences, and output the corresponding contextual vectors. The annotation which acts as contextual information is also decoded to help predict potential visiting features of visit sessions. Finally, for the user, the model decodes each visit session by the ST-LSTM model, and outputs the travel intention at each time step by his/her contextual vectors. At the same time, the probability distribution over locations which are likely to go next is obtained by decoding the output annotation of this layer with a softmax function. Similar to previous models, HST-LSTM model sorts the distributed probabilities, and chooses the maximum value as the prediction result.

3.2.3 MLP (Clustering) Model

MLP (Clustering) model is a neural network model proposed in [12]. It is based on a variant of the multi-layer perceptron (MLP) architecture, and it conducts a mean-shift clustering algorithm to calculate the destination probabilities. Due to its simple network structure, we make a brief introduction below.

The model first constructs one embedding table for each existing metadata, such as client ID, taxi ID date and time. The model then takes the trajectory prefixes and trajectory associated metadata to form the input vectors to feed into the hidden layer. Here, the metadata part of a trajectory is represented by concatenated embedding vectors, which will be learnt jointly by the model. Next, the model uses standard hidden layers consisting of a matrix multiplication followed by a bias and a Relu nonlinearity to train the trajectories. Finally, the destination probability distribution can be obtained. Instead of directly predicting the destination, the model predefines a set of destination cluster centers, and adopts a mean-shift clustering algorithm to find the final destination.

Discussion. In spite of these models solve the destination prediction problem, they still have inadequacies and differences. For the SubSyn algorithm, each trajectory sequence becomes longer under fine granularity, that aggravates the data sparsity problem. The prediction of a location considers the previous location only, and ignores the correlations between locations that the user have passed. However, the adopted bidirectional LSTM network in TALL model can fix the above defects. But the TALL model does not consider the information of spatial-temporal intervals between consecutive locations, which is essential for modeling users' mobility patterns. Fortunately, the ST-LSTM model provides us a good example, but it only focuses on the improvement of long-term effects of spatial-temporal intervals by an add operation, and ignores the interaction with short-term effects in the network. Thus it needs to be further enhanced. For the H-TALL model, the prediction models under different granularities are similar, which limits the multi-granularity learning capability of the model. Consequently, we need to design more comprehensive and effective models with all above-mentioned limitations considered, so that the prediction accuracy can be further improved.

4 Solutions

In this section, we first propose a Location-Aware Attention mechanism to adaptively capture the distinct features of locations in the trajectory in Section 4.1, and then develop a Trajectory-LSTM block which utilizes time and distance intervals to model the long-term and short-term effects of previous locations on the current location in Section 4.2. A well-designed deep learning approach called Location-Aware Attentional Trajectory-LSTM (LATL) model that integrates the Location-Aware Attention mechanism and the Trajectory-LSTM block is proposed in Section 4.3. Furthermore, in Section 4.4, we propose a hierarchical model that achieves superior learning ability by combining different network layers to explore and merge meaningful features under each granularity.

4.1 Location-Aware Attention Mechanism

In our previous work [2], in order to highlight the decisive role of some strongly correlated locations, we used the standard attention mechanism. It emphasizes the locations by fixed feature representations, and that leads to a limited ability of representing the distinct features of locations in the trajectory. This will also lead to an insignificant accuracy improvement. Meanwhile, for a random location in the research region, there are usually some other locations having features strongly relevant to it. Thus, we take this into consideration and propose an adaptive attention mechanism which is named as *Location-Aware Attention* mechanism. It can dynamically adapt to locations locally in the learning process and also improve the prediction accuracy. Details are given below.

First, let $\mathcal{L} = \{l_1, l_2, \ldots, l_{|\mathcal{L}|}\}$ denote the set of locations, and $|\mathcal{L}|$ is the total number of locations in the research region. Due to the way of partitioning the geographical space, each location l_i is associated with a sequence of some locations from $\mathcal{L}$, denoted by $\mathcal{L}^{l_i} = \{l_1^{l_i}, \ldots, l_t^{l_i}, \ldots, l_n^{l_i}\}$, where $l_t^{l_i} \in \mathcal{L}$ and n is the total number of locations interacted with location l_i. Note that the index t for $l_t^{l_i}$ denotes the relative element subscript rather than absolute timestamp order.

As we mentioned above, the locations in $\mathcal{L}^{l_i}$ are diverse, and only a subset of $\mathcal{L}^{l_i}$ may be strongly related to location l_i. The fixed feature representation in standard attention mechanism fails to convey the diverse features of locations locally. Motivated by this intuition, the Location-Aware Attention mechanism adopts non-fixed feature representation on the memory component of the neural network. The key idea is to calculate the relevance scores for location l_i and its relevant locations in $\mathcal{L}^{l_i}$. This can capture the relevant features between them, and let the model learn more accurately about the features of l_i at each time step. More locations in $\mathcal{L}^{l_i}$ with high relevance scores to location l_i indicates that the current location of user movement is more likely to be l_i, which also shows the adaptive learning ability of the neural network.

We let $Y^{l_i} = \{v_1^{l_i}, \ldots, v_t^{l_i}, \ldots, v_n^{l_i}\} \in \mathcal{R}^{n \times K}$. K is the dimension of latent vector. Each element $v_t^{l_i}$ is the vector representation of location $l_t^{l_i}$ in the neural network. Thus the adaptive attention network measures the relevance scores between location l_i and each location in $\mathcal{L}^{l_i}$ by

$$a_i = \frac{exp(Y^{l_i} v_{l_i})}{\sum_{l_m \in \mathcal{L}/\mathcal{L}^{l_i}} exp(Y^{l_i} v_{l_m})}, \tag{9}$$

where $v_{l_i} \in \mathcal{R}^{K \times 1}$ is the vector serving as a representation of location l_i. $\mathcal{L}/\mathcal{L}^{l_i}$ represents the set of all locations in $\mathcal{L}$ except the locations in $\mathcal{L}^{l_i}$. $a_i \in \mathcal{R}^{n \times 1}$ is the adaptive attention column vector for location l_i. The larger value of a_i, the higher relevance between location l_i and the corresponding location in $\mathcal{L}^{l_i}$, which also means that the network will attach a higher weight for location l_i as the prediction result of user's current moving location. By attaching this memory component into the neural network model, the features learned in each step can be closer to the adaptive features of the location, so as to finally achieve the purpose of improving the destination prediction accuracy.

4.2 Trajectory-LSTM

Another problem of our previous model [2] is that, the adopted bidirectional LSTM model ignores the information of spatial-temporal intervals between consecutive locations, which is essential for modeling users' mobility patterns. A location passed a short time ago and a short distance away always has much influence, and vice versa. We learn from the method in [13], and design a *Trajectory-LSTM* block to model the long-term and short-term effects of passed locations simultaneously. Details are as follows.

In Trajectory-LSTM block, we add two time gates, T_t^1, T_t^2, and two distance gates, D_t^1, D_t^2, based on the standard LSTM in Section 2.2. T_t^1 and D_t^1 are designed to capture the time and distance intervals to model the long-term effects. Similarly, T_t^2 and D_t^2 are used to control the short-term effects of the last passed location on the current location. That also means that the symbol of forget gate f_t has been removed. We use $(1 - i_t)$ to replace the i_t in Eq. (4), and replace the f_t with $(1 - i_t \odot T_t^2 \odot D_t^2)$. This can reduce the number of parameters, and learn more useful information efficiently. We use Δt_t to represent the time interval between two adjacent locations. Δd_t is the corresponding actual geographical distance. The time gate T_t^1 first stores Δt_t, then transfers it to the cell state c_t, and finally transfers it to c_{t+1}, c_{t+2}, ..., to model the long-term time effects. The same approach for the distance gate D_t^1 is applied to model the long-term distance effects. By this way, the cell state c_t can capture not only the order of locations, but also the long-term effects of time and distance interval information between adjacent locations.

Furthermore, the time gate T_t^2 and the distance gate D_t^2 are regarded as two input information filters. A location that has just passed and is geographically close has a greater impact on the prediction of the current location. Thus this block adopts two constraints, $W_{t_2} \leq 0$ in T_t^2 and $W_{d_2} \leq 0$ in D_t^2, to achieve such learning. If Δt_t is smaller, T_t^2 would be larger, and it would have a greater influence on the current location prediction. That means we increase its influence to better reflect the short-term effects. On the other hand, if Δt_t is larger, x_t would have a smaller influence, and the cell state c_{t-1} would have more significant effects accordingly, that means the long-term effects will dominate the learning process. This also indicates the mutual restriction between the long-term and short-term effects. For T_t^1, it doesn't make sense to impose such constraint in terms of modeling the long-term effect. That is also the reason we design two time gates and two distance gates in Trajectory-LSTM block. The same is true for Δd_t, D_t^1 and D_t^2, and we do not explain redundantly here.

4.3 Location-Aware Attentional Trajectory-LSTM Model

Overview. As we mentioned before, using the standard attention mechanism is not satisfactory enough for the destination prediction problem. The reason is that the fixed feature representation limits the ability of capturing distinct features of different locations during the learning process, and leads to an insignificant accuracy improvement. Besides, the bidirectional LSTM network ignores the spatial-temporal features between consecutive locations. Hence, it is insufficient

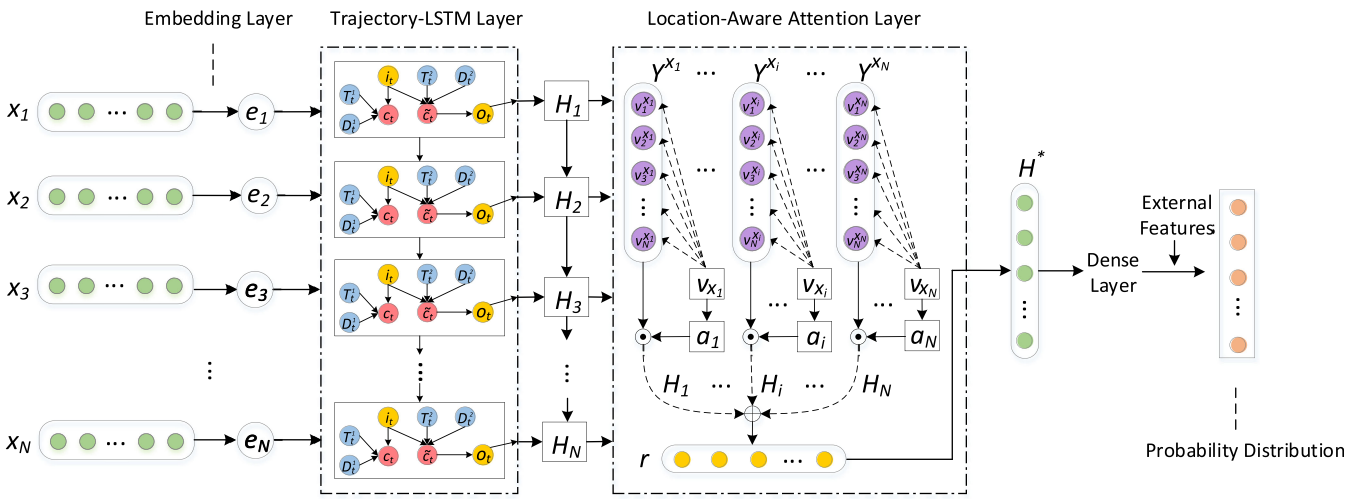


Fig. 3. Location-aware attentional trajectory-LSTM model.

to support more accurate fine-grained prediction. Therefore, we re-design the model, and propose a *Location-Aware Attentional Trajectory-LSTM* model (LATL model in short) based on the elementary Location-Aware Attention mechanism and the Trajectory-LSTM block. The structure of this model is shown in Fig. 3.

The LATL model first uses the embedding technique to convert the training trajectory sequences into the form that the network can take. Then, the Trajectory-LSTM layer is adopted to capture the long-term and short-term effects of time and distance intervals between consecutive locations in the trajectory. Towards the output vectors of Trajectory-LSTM layer, the Location-Aware Attention layer alters the fixed length representation, and captures the features between the current location and its relevant locations to improve the prediction accuracy. At last, the model fuses some external features, and adopts a softmax function to get the probability distribution of each location being the destination.

Embedding Layer. Given an observed prefix trajectory $T^P = \{x_1, x_2, \ldots, x_N\}$ 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, here the dimension of the vector is the quadratic value of the grid granularity. Then the embedded prefix trajectory is transformed into a two-dimensional vector, which is denoted as $emb_s = \{e_1, e_2, \ldots, e_N\}$. As shown in the embedding part of Fig. 3, the vector emb_s is the input vector of the deep neural network.

Trajectory-LSTM Layer. In modeling trajectory sequential patterns, applying the LSTM network directly can handle the gradient vanishing and exploding phenomena by the gating mechanism, and is suitable for our problem (i.e., longer trajectory sequence in fine-grained prediction). Unfortunately, LSTM network can not preserve meaningful latent characteristics of the whole trajectory. As such, a bidirectional LSTM is adopted in the TALL model which is introduced in Section 3.2.1, to learn the latent features of both preceding and following locations in trajectory. However, the adopted bidirectional LSTM is primitive, without considering the explicit relation between consecutive locations. It leads to a low degree of learning efficiency, and has an impact on the prediction result. Therefore, we have modified the internal structure of the LSTM block. It now takes spatial and temporal relation between locations in the trajectory into consideration, to learn the long-term and short-term effects of passed locations. There are two time gates and two distance gates in the Trajectory-LSTM block, and the forget gate in the primitive LSTM has been removed to reduce the learning of parameters and improve the efficiency.

The internal structure of Trajectory-LSTM block has been introduced in Section 4.2. Thus the output of the ith location in this layer is shown in the following equation:

$$h_i = o_i \odot tanh(\tilde{c}_i), \tag{10}$$

where $\tilde{c}_i$ is a new cell state that stores the intermediate result in Trajectory-LSTM block, and o_i is the result from the output gate. In this way, the hidden state annotation h_i summarizes both the long-term and short-term effects of all passed locations (x_i is the last passed location) on the prediction of the current location, and it will be the input of the next layer. Due to the design of the Trajectory-LSTM block, the annotation h_i will focus on the long-term effects, since we are more uncertain about the short-term effects.

Location-Aware Attention Layer. The standard attention mechanism in our previous work highlights the decisive role of the strongly correlated locations to the destination. However, the feature representation is fixed and identical for the input locations at each time step, thus the distinct features of these locations are neglected. In the light of this, the LATL model incorporates a Location-Aware Attention layer. It adopts non-fixed feature representations, and calculates the relevance scores between the current location and input locations. This enables the trained network to generate weight distribution more tailored for the current location. The network can learn the features of locations in the sequence selectively, and make more accurate prediction at each time step until the final destination is predicted.

The internal structure of the Location-Aware Attention mechanism has been introduced in Section 4.1. As shown in the Location-Aware Attention layer in Fig. 3, we first let $H_i = [h_1, h_2, \ldots, h_N]$ be the hidden state matrix of location x_i, which is produced by the Trajectory-LSTM layer after the iteration. N is the length of the prefix of trajectory, and

also the total number that we set for the relevant locations of x_i. Then, we form the adaptive representation of location x_i by a weighted sum of $Y^{x_i} = \{v_1^{x_i}, \ldots, v_t^{x_i}, \ldots, v_N^{x_i}\}$ in Eq. (11)

$$c^{x_i} = {Y^{x_i}}^T \odot a_i \tag{11}$$

$$r = \sum_{i=1}^{N} c^{x_i} H_i, \quad r \in \mathcal{R}^{d^w}. \tag{12}$$

In this way, c^{x_i} is an intermediate value and it reflects the preference of user for location x_i at current time step. The annotation r contains the relevance features between each location x_i in the prefix trajectory and the certain relevant locations to itself. At last, we obtain the location-pair representation H^* which is used for subsequent classification by a hyperbolic tangent function on annotation r.

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 when there are big events. Furthermore, different weather conditions may affect people's final destination. 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. We use forecasting weather information to extract 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 matrix H^* and Y_E as inputs and applies an affine transformation with weights W^{T^P}, W^E and biases b^{T^P}

$$\begin{aligned} &\hat{p}(y|T^P) = softmax(W^{T^P} H^* + W^E Y_E + b^{T^P}) \\ &\hat{y} = \arg\max_y \hat{p}(y|T^P). \end{aligned} \tag{13}$$

In the LATL 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 $\hat{y}$

$$L(\theta) = -\frac{1}{m} \sum_{i=1}^{m} t_i \log(y_i) + \lambda \|\theta\|_F^2, \tag{14}$$

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

4.4 Hierarchical Location-Aware Attentional Trajectory-LSTM Model

Overview. In our previous work [2], we use similar prediction model under each spatial granularity in the proposed hierarchical learning model. However, for the distinct features of trajectory sequence under different granularities, we should design more tailored prediction models to better understand the mobility patterns. For example, the sequence representation of the trajectory in finer granularity is usually longer, we can first use a dimensionality reduction layer to improve the training speed of the model, while this operation is not required under a coarse granularity. Thus, in this section, we re-design a hierarchical learning model named H-LATL model. It has the advantage of adopting more tailored combination of neural network layers under each spatial granularity, and merging the patterns in different granularities to improve the prediction accuracy.

The structure of this hierarchical model is shown in Fig. 4. It is an example of the region within the fourth ring road of Beijing. One schematic trajectory is selected as the prediction trajectory. Such an H-LATL model is composed of M levels. The larger the number of the level, the finer the spatial granularity. Each level corresponds to a spatial granularity, and contains a prediction model. Each prediction model is a combination of neural network layers. Since trajectory sequence is a typical kind of time series data, in order to find the most suitable structure of the model, we have tried a variety of currently popular neural network layers, including the standard LSTM layer [9], bidirectional LSTM layer [11], Convolutional Neural Network (CNN) layer [14], as well as our designed Location-Aware Attention layer and Trajectory-LSTM layer.

In addition, as the prediction models are independent, when the prefix trajectory is given, the models carry out the prediction process in the order of grid granularity from coarse to fine, and give the most possible top-k destinations respectively, i.e., the area enclosed by the blue squares on the corresponding map in Fig. 4. Obviously, we choose the prediction results in the finest granularity (level M) as the final output of the H-LATL model. The combination of the neural network layers for each spatial granularity we obtained empirically from our experiments, as well as the transmission of features between the prediction models are given below.

Raw Trajectory Conversion. As mentioned before, each level in Fig. 4 corresponds to a spatial granularity. At the very beginning, we divide the research region with different spatial granularities. For example, in level 1, the spatial granularity is 10, in level 2, the granularity is 20, and so on for each of the M levels. Consequently, the same raw trajectory will have distinguished cell sequence representations after the conversion, denoted by $\mathbb{T}^1$, $\mathbb{T}^2$, …, and $\mathbb{T}^M$. Notice that all these sequence representations have equal length. Though

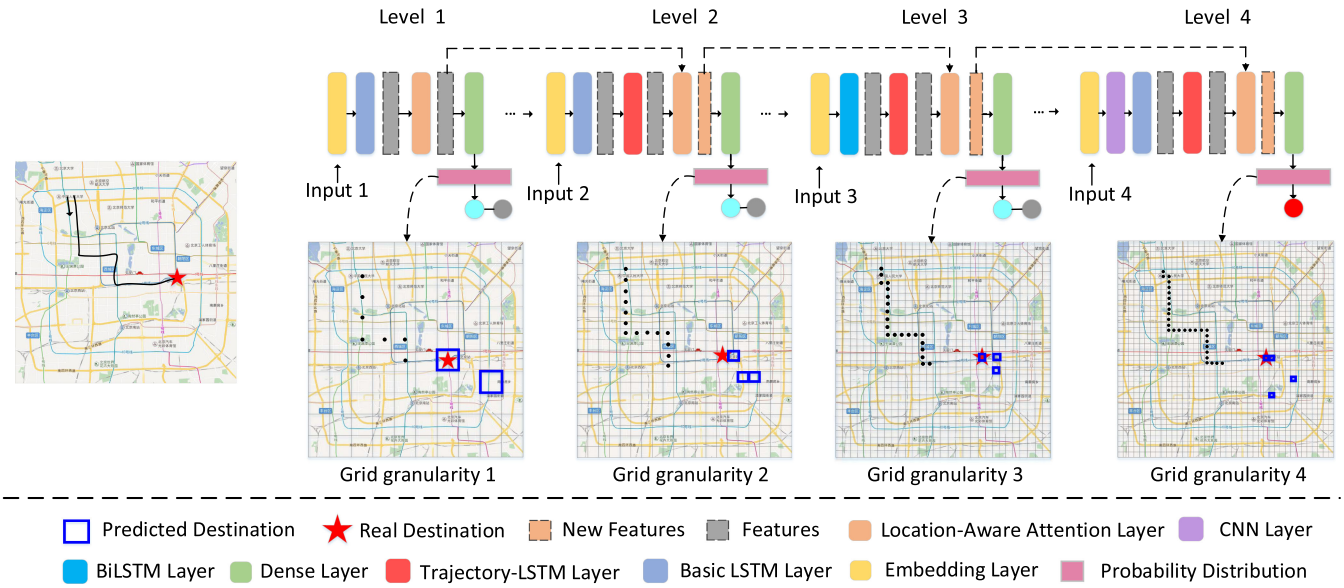


Fig. 4. Hierarchical location-aware attentional trajectory-LSTM model.

we use the same symbol to denote all the raw location points that fall in the same cell before, we do not delete duplicate location points in each trajectory sequence representation. In order to further convert the sequence to the format acceptable to the neural networks, we use the same embedding technique in Section 4.3. Formally manifested by

$$emb^m = (e_{m,1}, e_{m,2}, \ldots, e_{m,N}) \quad m = 1, 2, \ldots, M, \tag{15}$$

where M is the number of levels in the model. N is the length of the trajectory sequence. The dimension of each vector $e_{m,i}(i = 1, 2, \ldots, N)$ is the quadratic value of the corresponding spatial granularity. Moreover, these two-dimensional vectors are treated as the multi-input of the H-LATL model, denoted by *input 1*, *input 2*, ..., *input M* in Fig. 4.

Tailored Combinations of Layers. For the H-LATL model, in order to eliminate the areas that are unlikely to be the destination, and optimize the accuracy of predicted results, we would like to better understand the mobility patterns of the trajectory sequence under different spatial granularities. Thus, we combine different neural network layers under each spatial granularity to find the most suitable structure. Detailed description of the layer combination in each prediction model is given below:

Grid Granularity 1. Under this granularity, after the grid partition, the actual corresponding geographical area of each cell in the grid is relatively large. We carry out a more basic operation in this level, i.e., adopting the *standard LSTM layer* in Section 2.2. The formula used to indicate the LSTM layer is

$$h_n^1 = lstm_{enc}(x_n, h_{n-1}, c_{n-1}) \quad n = 1, 2, 3, \ldots, N, \tag{16}$$

where $lstm_{enc}$ denotes the LSTM functions (Eqs. (1), (2), (3), (4), (5), and (6)) for encoding the sequence contexts. h_{n-1} and c_{n-1} are the hidden state annotation and the previous cell state for current input location x_n respectively. The hidden state annotation h_n summarizes the contexts of all passed locations in the trajectory sequence, and it will be input to the next layer.

Grid Granularity 2. Based on the process under the first granularity, we would like to take the time and distance intervals into account to improve the prediction accuracy. Consequently, we adopt the *Trajectory-LSTM layer* and combine it with the standard LSTM layer under this spatial granularity. The formula is

$$h_n^2 = trajlstm_{enc}(h_{n-1}^1, T_n, D_n), \tag{17}$$

where $trajlstm_{enc}$ represents the function of the Trajectory-LSTM in Section 4.2. h_{n-1}^1 summarizes the features information obtained from the standard LSTM layer. T_n and D_n denote the time gates and distance gates respectively. We use them to capture the long-term and short-term effects of the time and distance intervals on the prediction of the current location.

Grid Granularity 3. Obviously, the standard LSTM layer under previous two granularities does not take the future contexts into account. Actually, the latent features of both proceeding and following locations in a trajectory sequence are useful. Thus, inspired by the deep learning model in [2], we further adopt a bidirectional neural network to solve the problem. The formal equations that indicate the combination of the *bidirectional LSTM layer* as well as the Trajectory-LSTM layer are

$$\begin{aligned} h_n^{3'} &= bilstm_{enc}(h_{n-1}, w_n) \\ h_n^3 &= trajlstm_{enc}(h_{n-1}^{3'}, T_n, D_n), \end{aligned} \tag{18}$$

where $bilstm_{enc}$ denotes the bidirectional LSTM function for encoding the left and right sequence contexts. $h_{n-1} \in \mathcal{R}$ is the hidden state annotation, and at the initial time step, $h_0 = \{0\}$. w_n is the corresponding weight vector. The hidden state annotation $h_n^{3'}$ summarizes the context features of both the proceeding and following locations in a trajectory sequence, and it is also the input annotation of the Trajectory-LSTM layer.

Grid Granularity 4. As the granularity becomes finer (40 in this level), there is a larger vector dimensionality for the

location sequence representation, and the networks need to learn more parameters. Therefore, we consider to use the convolutional neural networks (CNN) to achieve the dimensionality and parameter reduction. Consequently, the combination of a CNN layer, an LSTM layer and a Trajectory-LSTM layer is adopted under this granularity. Inspired by successful CNN models for dimensionality reduction in other works [15], [16], for the two-dimensional matrix generated in the embedding layer, we let the CNN layer perform convolutional operation with $f \times f$ filter stride by s at a time. Thus, the output size can be calculated by the following equation:

$$w = \frac{w - f + 2p}{s} + 1, \tag{19}$$

where w is the size of the width, f is the size of the filter, p is the size of the padding, and s is the stride. Specifically, the purpose of the layer is the dimensionality reduction, thus we do not use the pooling layer, and set filter as 1×1, stride as 1 and p as 0. After a concatenation operation, the convolutional layer sums up the contributions from previous vector matrices. The output of this layer goes through the LSTM layer and the Trajectory-LSTM layer. We can also get the final hidden state annotation h_n^4 in Trajectory-LSTM layer. Due to the space limitation, we do not give specific formulations here.

Location-Aware Attention Layer. Furthermore, in order to continuously capture the features between the candidate location and its relevant locations, we further apply the *Location-Aware Attention layer* in each prediction model to improve the prediction accuracy at each time step. Formally

$$\begin{aligned} r^m &= adap\_att_{enc}^m(H_n^m, a_n^m) \\ H^{m,*} &= tanh(r^m), \end{aligned} \tag{20}$$

where $adap\_att_{enc}^m$ denotes the functions of Eqs. (11) and (12). H_n^m is the hidden state matrix of locations under the mth granularity ($m = 1, 2, \ldots, M$). a_n^m is the adaptive attention column vector for locations under the mth granularity. r^m is formed by a weighted sum of vectors outputted from the previous combined layers. Here we still adopt a *tanh* activation function to obtain the final location-pair representation $H^{m,*}$.

Subsequently, since the core of this model is the fusion of features outputted by the last neural network layer in different levels, the prediction model under mth granularity takes the current location-pair representation $H^{m,*}$, and the location-pair representation $H^{m-1,*}$ under (m-1)th granularity to comprise a new representation. Formally

$$H_{new}^{m,*} = H^{m,*} \oplus H^{m-1,*}, \tag{21}$$

where $\oplus$ represents the element-wise sum function. The new location-pair representation $H_{new}^{m,*}$ contains the implicit correlations of the same raw trajectory under previous m granularities by iterations. The correlation features will subsequently be fed to the predicting layer in the following prediction model.

Prediction. As the H-LATL model is a multi-output model, different sequence representations of the same raw trajectory, $\mathbb{T}^1$, $\mathbb{T}^2, \ldots$, and $\mathbb{T}^M$, have different corresponding output destinations, denoted by $\mathbb{T}_d^1$, $\mathbb{T}_d^2, \ldots$, and $\mathbb{T}_d^M$ respectively. The predicting layer in the mth prediction model obtains the top-k cells with the highest probability being the destination $\mathbb{T}_d^m$ of location sequence $\mathbb{T}^m$ according to the softmax function. Then, the layer determines whether the real destination marked by red star in Fig. 4 is within the top-k set, and excludes some unlikely destination locations based on this. Note that we still incorporate the influence of external features here. Moreover, the finer granularity the geographical space decomposed by, the smaller the geographical area that the predicted destination within. We give the prefix of one trajectory under the finest granularity $\mathbb{T}^M$, and find out the destination $\mathbb{T}_d^M$ as the final output of the H-LATL model.

5 Experimental Studies

In this section, we conducted extensive experiments on two real trajectory datasets to evaluate the performance of the baseline algorithms and our newly proposed methods.

5.1 Datasets

We used two trajectory datasets from Beijing and Chengdu to test the effectiveness of our models, both of them are collected from taxis in the city. Details are shown as follows.

- *Beijing:* We mainly extracted the area within the fourth ring road of Beijing. The trajectory data and context factors were collected from 1st Mar. to 31st Jul. in 2016. There are more than 2 million trajectories covering the road network every day. The weather condition can be divided into 10 types (e.g., Rainy, Sunny) and temperature ranges from -4 °C to 36 °C. About 12 important holidays and 24 weekends exist among the dataset. We use data of the first 4 months 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 thousand trajectories that can be obtained every day. The weather condition is divided into 5 types and the temperature ranges from 6 °C to 19 °C. There are 4 weekends in this month. The last 9 days are test set and the others are training set.

5.2 Training

In the preprocessing stage, we first find the maximum value of the longitude and latitude respectively, and determine the research region. Then, we divide the grid into $g \times g$ cells, where g is the grid granularity and the default value is 30. Each cell in the grid has one id number $i (1 \leq i \leq g \times g)$. Thus, the geographical locations indicated by longitude and latitude within a same grid cell will have an identical id number, and the raw trajectory sequence can be converted into a sequence of grid cells. We find out the trajectory of the largest length, and take two-thirds of the length as the input size of the model. For our dataset, the length we set is 47. It is also the size for the number of locations interacted with a candidate location in the Location-Aware Attention mechanism component. For the trajectories with a length shorter than the fixed-size, we pad the input vector by

repeating either the first or the last location. We leverage the mini-batch learning method (the size is 48), and train the model until convergence. A multi-class logistic regression is adopted to get the probability distribution on all grid cells. The *categorical-crossentropy* is applied as our loss function. In addition, we set the learning rate as 0.001, and adopts Adam, a variant of Stochastic Gradient Descent (SGD) to optimize the parameters in LATL model. Moreover, we use the projection operator described in [17] to meet the constraints in Trajectory-LSTM block. If the constraint appears larger than 0, we set it to be equal to 0.

5.3 Baseline Methods

We compared our models with the baseline methods introduced in Section 3.

- *SubSyn* [3]: It utilizes a Markov model to establish the transfer relationship between the locations in the trajectory, and follows a Bayesian inference framework. The Bayes' rule is applied to compute the transition probabilities and construct the transition matrices.
- *TALL* [2]: It incorporates a bidirectional LSTM layer and a standard attention layer to learn the features of both preceding and following locations in trajectory, and emphasize the features that have strong correlations to destination. Top-k locations with the highest probabilities will be outputted to be the most possible destinations.
- *H-TALL* [2]: It is the hierarchical extension model of TALL. The research region covered by the datasets is divided by M different spatial granularities respectively. It fuses the patterns rationally by the connection of hidden layers in the prediction model under each granularity. The final predicted results are outputted under the finest granularity.
- *ST-LSTM* [10]: It is based on a standard LSTM structure. The spatial-temporal factors are fused into the three gates of LSTM by a novel and efficient add operation. It partitions the time and distance intervals into discrete slots, and only encodes the upper and lower bound of these slots to reduce the parameters learning.
- *HST-LSTM* [10]: It is the hierarchical extension model of ST-LSTM. It considers the periodicity of users' movements that they tend to visit the same or similar places, and encodes users' visit historical information as contexts to boost the prediction accuracy.
- *MLP (Clustering)* [12]: It is a neural network model based on a multi-layer perceptron (MLP) architecture. The input layer receives the representations of prefix trajectories with associated context information, and the standard hidden layers are adopted to train the trajectories. The output layer adopts a mean-shift clustering algorithm to predict the destinations instead of directly computing the probability distribution.

5.4 Evaluation Metrics

To evaluate the performance of our proposed models (LATL model and H-LATL model), and compare with the six baseline algorithms described above, we use two standard metrics Acc@k and Mean Prediction Accuracy (MPA@k). The first is defined as: $Acc@k = \frac{\#hit\_pois@k}{\#test\_pois}$, where $\#hit\_pois@k$ means the number of predicted destinations ranking at top-k and $\#test\_pois$ stands for the number of total testing trajectory points. Another evaluation criteria MPA@k is defined as $MPA@k = \frac{\#hit\_trajs@k}{\#test\_trajs}$, where $\#hit\_trajs@k$ indicates the total number of trajectories that the real destination (ground truth) is in top-k predicted locations, and $\#test\_trajs$ represents the number of all testing trajectories. In this paper, we choose $k = 1, 5, 10, 20$.

5.5 Performance Evaluation

In this section, we will give an overall performance presentation of all methods, and evaluate our LATL model and H-LATL model under different parameter variations.

Overall Performance. Due to the large amount of data in Beijing dataset, we choose it as the main representative experimental dataset. The performance of our models (LATL, H-LATL) and the six baseline algorithms on two datasets evaluated by Acc@k with training set size of 20, 40, 60, 80 percent is shown in Table 2. In addition, Table 3 presents the experimental results evaluated by MPA@k. The cell size and the hidden state size are set as 128 in our experiments. The number of epoch is set as 50, and the batch size is set as 100 for our models. In the following, we use g and tcp to represent the grid granularity and trip completed percentage (the length of the prefix trajectory) respectively. The default values of them are: k = 5, g = 30, tcp = 70%. The parameters of other baseline algorithms follow the best settings in their papers respectively. From the experimental results, we have the following observations.

First, the accuracy of TALL model is generally higher than the SubSyn algorithm under two metrics, due to the data sparsity problem in finer grid granularity. The bidirectional LSTM and the standard attention layer adopted in TALL improves this shortcoming. The experimental results of its extension network, H-TALL model, are superior to the above two methods, since each step in the hierarchical model has the operation of excluding impossible destinations. ST-LSTM model and H-LSTM model have similar effects as TALL model on our two datasets by effectively adding the spatial-temporal factors. The overall effect of MLP (Clustering) model is not very well compared with other models, due to its simple network structure and less consideration of the context information of trajectory data. In addition, our LATL model and H-LATL model are superior to other methods to some extent on both Beijing and Chengdu dataset, which demonstrates the effectiveness of the proposed Location-Aware Attention mechanism and Trajectory-LSTM block. H-LATL model achieves much better prediction performance than LATL model, which shows that the prediction model combined by different neural network layers in each level is indeed suitable for the corresponding spatial granularity. Particularly, we did not try all the combination possibilities, but the current structure of H-LATL model described in Section 4.4 has already shown the potential to consider different combinations of neural network layers for different granularities. Actually, for all of the neural network methods, we did experiments with embedding and no embedding technique respectively.

TABLE 2
Performance in Terms of Acc@k for All Methods With Training Set Size of 20, 40, 60, and 80 Percent on Beijing Dataset and Chengdu Dataset

Beijing

Method	Acc@1				Acc@5				Acc@10				Acc@20			
	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%
SubSyn	0.0887	0.1889	0.2878	0.3248	0.1746	0.3386	0.5753	0.5547	0.3078	0.3859	0.6036	0.6483	0.3278	0.4864	0.6345	0.6443
TALL	0.1073	0.1864	0.3098	0.3187	0.1833	0.3484	0.5749	0.6184	0.2764	0.4432	0.5836	0.6539	0.3423	0.4973	0.6322	0.6756
H-TALL	0.1169	0.2284	0.3202	0.3264	0.2466	0.3683	0.6039	0.6380	0.3087	0.4622	0.6023	0.6867	0.3864	0.5712	0.6963	0.7489
ST-LSTM	0.0977	0.1729	0.2781	0.3247	0.1946	0.3274	0.4988	0.5147	0.2176	0.3988	0.5555	0.6333	0.2753	0.5276	0.6234	0.7373
HST-LSTM	0.1053	0.3191	0.3241	0.3833	0.2533	0.5424	0.6347	0.6545	0.2878	0.5806	0.6386	0.7084	0.3422	0.6214	0.7259	0.7898
MLP (Clustering)	0.0735	0.1694	0.2533	0.3208	0.2211	0.3156	0.4951	0.5389	0.2378	0.3711	0.5233	0.5621	0.3254	0.4544	0.5782	0.6347
LATL	**0.1134**	**0.2545**	**0.3398**	**0.3570**	**0.2366**	**0.4311**	**0.6143**	**0.6444**	**0.3267**	**0.5722**	**0.6532**	**0.7165**	**0.4445**	**0.6154**	**0.7031**	**0.8001**
H-LATL	**0.1263**	**0.3078**	**0.3751**	**0.4256**	**0.2652**	**0.5574**	**0.6475**	**0.6798**	**0.3544**	**0.6245**	**0.7133**	**0.7444**	**0.5245**	**0.6950**	**0.7525**	**0.8233**

Chengdu

Method	Acc@1				Acc@5				Acc@10				Acc@20			
	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%
SubSyn	0.0453	0.1134	0.1984	0.2333	0.1347	0.2904	0.4563	0.4764	0.2456	0.3553	0.4865	0.5565	0.2854	0.4464	0.5834	0.6154
TALL	0.0679	0.1152	0.2174	0.2459	0.1485	0.3054	0.4653	0.5053	0.2889	0.4132	0.5075	0.5644	0.3564	0.4667	0.5764	0.6445
H-TALL	0.0846	0.1583	0.2456	0.2767	0.2057	0.3222	0.4967	0.5246	0.3111	0.4343	0.5245	0.6045	0.3943	0.5332	0.6254	0.7133
ST-LSTM	0.0767	0.1346	0.1990	0.2444	0.1578	0.3655	0.4489	0.5164	0.2333	0.3889	0.4664	0.5333	0.3153	0.5143	0.6424	0.7034
HST-LSTM	0.0832	0.1835	0.2223	0.3343	0.2153	0.4353	0.4877	0.5475	0.2884	0.4505	0.5349	0.6166	0.3645	0.5463	0.6777	0.7422
MLP (Clustering)	0.0651	0.1554	0.2143	0.2522	0.1452	0.2811	0.3778	0.4255	0.1954	0.3278	0.4467	0.5351	0.2855	0.3951	0.5156	0.5851
LATL	**0.0865**	**0.1962**	**0.2756**	**0.3354**	**0.2251**	**0.4092**	**0.4532**	**0.5344**	**0.2874**	**0.4462**	**0.5165**	**0.6251**	**0.3751**	**0.5354**	**0.6355**	**0.7163**
H-LATL	**0.1033**	**0.2778**	**0.3489**	**0.3853**	**0.2599**	**0.4889**	**0.4964**	**0.5811**	**0.3281**	**0.4955**	**0.5556**	**0.6667**	**0.4165**	**0.5889**	**0.6733**	**0.7556**

TABLE 3
Performance in Terms of MPA@k for All Methods With Training Set Size of 20, 40, 60, and 80 Percent on Beijing Dataset and Chengdu Dataset

Beijing

Method	MPA@1				MPA@5				MPA@10				MPA@20			
	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%
SubSyn	0.033	0.054	0.095	0.185	0.124	0.167	0.235	0.296	0.167	0.188	0.337	0.374	0.304	0.353	0.425	0.435
TALL	0.095	0.121	0.143	0.211	0.165	0.192	0.257	0.346	0.183	0.209	0.381	0.418	0.286	0.344	0.402	0.496
H-TALL	0.133	0.177	0.233	0.293	0.198	0.216	0.354	0.444	0.222	0.235	0.426	0.486	0.262	0.386	0.471	0.603
ST-LSTM	0.076	0.122	0.167	0.233	0.156	0.181	0.266	0.312	0.187	0.218	0.373	0.474	0.306	0.366	0.437	0.525
HST-LSTM	0.154	0.215	0.266	0.311	0.197	0.232	0.376	0.433	0.208	0.266	0.424	0.563	0.366	0.385	0.484	0.577
MLP (Clustering)	0.134	0.165	0.223	0.313	0.155	0.204	0.281	0.354	0.214	0.243	0.327	0.414	0.258	0.314	0.443	0.533
LATL	**0.142**	**0.213**	**0.286**	**0.343**	**0.204**	**0.248**	**0.352**	**0.519**	**0.236**	**0.275**	**0.442**	**0.524**	**0.352**	**0.396**	**0.483**	**0.556**
H-LATL	**0.201**	**0.263**	**0.333**	**0.385**	**0.255**	**0.329**	**0.433**	**0.589**	**0.253**	**0.339**	**0.472**	**0.617**	**0.442**	**0.465**	**0.542**	**0.649**

Chengdu

Method	MPA@1				MPA@5				MPA@10				MPA@20			
	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%
SubSyn	0.029	0.095	0.158	0.247	0.144	0.227	0.294	0.341	0.216	0.287	0.364	0.382	0.301	0.342	0.406	0.468
TALL	0.102	0.146	0.209	0.272	0.186	0.242	0.312	0.405	0.257	0.317	0.370	0.451	0.281	0.363	0.416	0.523
H-TALL	0.155	0.219	0.294	0.347	0.217	0.281	0.357	0.481	0.265	0.344	0.428	0.511	0.361	0.401	0.472	0.587
ST-LSTM	0.095	0.161	0.237	0.305	0.182	0.253	0.364	0.421	0.235	0.283	0.392	0.463	0.302	0.389	0.430	0.513
HST-LSTM	0.179	0.254	0.329	0.379	0.219	0.325	0.434	0.492	0.257	0.366	0.456	0.525	0.343	0.431	0.492	0.554
MLP (Clustering)	0.143	0.222	0.277	0.348	0.191	0.258	0.313	0.39	0.234	0.286	0.352	0.431	0.318	0.356	0.433	0.475
LATL	**0.163**	**0.246**	**0.342**	**0.421**	**0.214**	**0.314**	**0.401**	**0.542**	**0.275**	**0.324**	**0.421**	**0.522**	**0.352**	**0.428**	**0.495**	**0.558**
H-LATL	**0.224**	**0.332**	**0.423**	**0.476**	**0.253**	**0.355**	**0.452**	**0.597**	**0.332**	**0.414**	**0.484**	**0.632**	**0.413**	**0.463**	**0.532**	**0.627**

Embedding methods are better than non-embedding methods in most cases, this shows the advantages of embedding technique when dealing with sparse data.

Impact of Location-Aware Attention Layer.

The Location-Aware Attention layer plays an important role in our models. By non-fixed feature representation, it can dynamically adapt to locations locally of a trajectory in the learning process, and improve the prediction accuracy at each time step. To further verify the validity of this conception, we conduct comparative experiments on the LATL model in three cases: with Location-Aware Attention layer, with Standard Attention layer and without Attention layer.

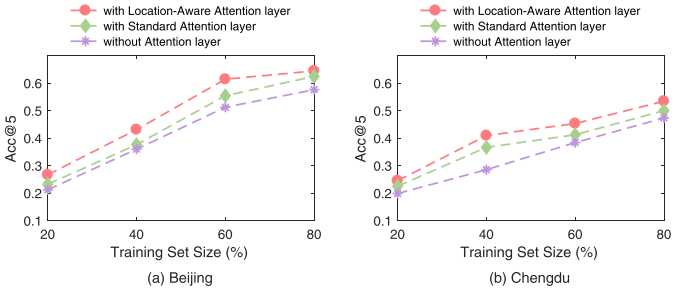


Fig. 5. Impact of location-aware attention layer.

The parameters are set as default values. As can be seen in Fig. 5, on both Beijing and Chengdu datasets, the performance with Location-Aware Attention layer attached is always better than the other two cases.

Impact of Parameters.

To evaluate the performance of our models on various user queries, we set multiple variable parameters in our experiments. As mentioned above, we have set default parameters for k, g and tcp. Consequently, the measurements will be evaluated against varying one parameter among three at each time. Note that in this part, we conducted comparative experiments on six methods in terms of Acc@k, including four baseline algorithms (SubSyn, H-TALL, HST-LSTM and MLP (Clustering)), the LATL model and the H-LATL model. TALL model and ST-LSTM model are the basic models of H-TALL model and HST-LSTM model respectively, their performance is always not better than the hierarchical models, thus we do not give their curves here. The default training set size is 60 percent. From the experimental results, we can get the following observations.

First, from these three graphs, we can see that with three parameters increasing, the accuracy of all six methods has improved on both two datasets. The prediction accuracy can be up to 75 percent on Beijing dataset and 67 percent on Chengdu dataset. The accuracy of LATL model and H-LATL model is always better than other models, and the overall upward trends of these six models are the same on three figures.

In addition, since the number of predicted destinations may be insufficient, the metric k takes this limitation into consideration for all methods in the experiments. In Fig. 6, we can see that when $k = 1$, all of the methods cover at least 25 percent of all tested locations. As k keeps increasing, the range of these predicted locations they covered begins to change and differ from each other on both two datasets. The accuracy of all models increases rapidly before $k = 10$ and decreases after it. In general, the proportion of locations covered by H-LATL model is always larger, this shows its superiority.

Furthermore, when the grid is divided under a coarse granularity, the number of matching prefixes of trajectories will get larger, since more locations in the training set may fall into an identical cell. A fine granularity has the advantage of higher prediction accuracy that small area of each cell brings, but it may cause the fine granularity sparsity problem consequently. Since less locations in trajectory lie in a same grid cell, the task of destination prediction becomes more difficult. Besides, requiring more time to complete the training phase is also a drawback of fine granularity. Therefore, we need to test these algorithms on the

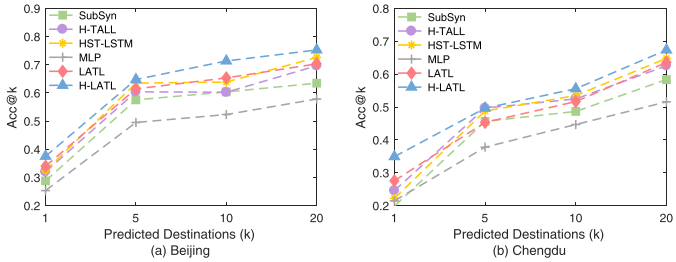


Fig. 6. Varying the value of k.

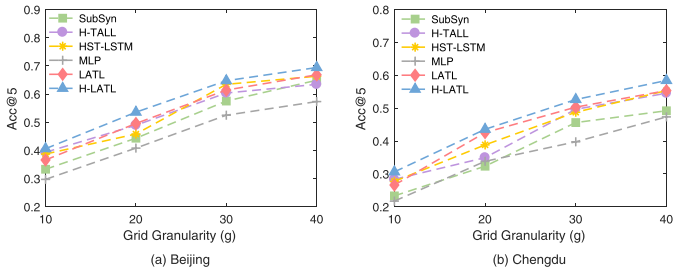


Fig. 7. Varying the grid granularity g.

prediction accuracy by varying the parameter g, and then determine a suitable value of granularity for the two datasets. All of the neural network models solve the problem better than the probabilistic model except MLP (Clustering) model. The trends of H-TALL model and the HST-LSTM model are particularly similar on two datasets. In Fig. 7, as the grid granularity becomes finer, the value of Acc@5 becomes larger. However, for all methods, it can be seen that before the value of g grows to 30, the value of Acc@5 increases by a large margin, the value after $g = 30$ grows at a small rate. Thus we set the granularity to 30 as the best predictor of the model, and take it as the default value.

Finally, We also investigate the effect of trip completed percentage of these models in Fig. 8. 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. The possible destinations will be distributed in multiple discrete areas. On the contrary, since the growing length of the given prefix of trajectory will relieve this problem, the number of similar matching trajectories in the dataset will correspondingly decrease, and that will also cause the data sparsity problem. Therefore, we need to observe the impact that this parameter will bring on prediction accuracy, and find a suitable value for the tested datasets. From Fig. 8, we can see that the H-TALL model performs better than the SubSyn algorithm and HST-LSTM model on Beijing dataset. On Chengdu dataset, it just happens to be the opposite. Before the value of tcp grows to 50 percent, the growth rate of Acc@5 is large. However, the subsequent growth rate slows down. When the value of tcp is higher, prediction has little practical value so we do not consider further. The LATL model and the H-LATL model solve the problem very well and they are significantly superior to the other models.

6 Related Work

In this section, we survey the related work on destination prediction methods and deep learning technologies in this field.

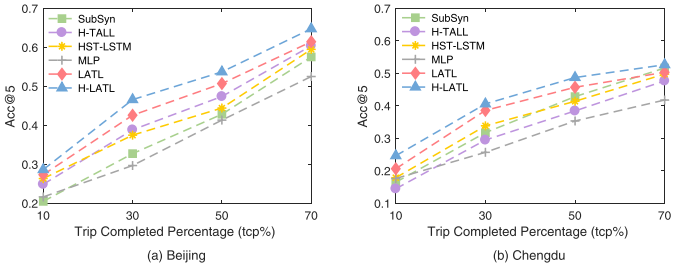


Fig. 8. Varying the percentage of trip completed.

6.1 Destination Prediction

Many studies have been conducted to address the destination prediction problem. Among them, the widely used models for this problem are the Bayesian network and the Markov model [18], [19]. They 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 [20]. For example, the NNT model proposed in [21] ensures the accuracy by analysing the distance between similar trajectories. In addition to these well-known models that could be directly applied in our problem, some other methods have also be proven to be effective. [22] proposes a method based on the analysis of real time moving patterns of users, and the driving contexts. Similarly, the day of week, the number of passengers and the weather are considered in [23]. Besides, an approach based on trip matching and ensemble learning in which they leverage the patterns observed in a dataset of taxi journeys to predict the corresponding final destination and travel time for ongoing taxi trips has been presented [24]. Furthermore, [25] proposes a user-personalized model for predicting route and destination, including places where user has never visited before, which is similar to our problem in this paper. All of the above methods have their specific application scenarios and good prediction results.

6.2 Deep Learning

In recent years, with the rapid development of deep learning technique, more and more researchers began to adopt it for various problems [4], [26]. First, a lot of works have been done on a variant of RNN, LSTM [9]. For example, an efficient vehicle trajectory prediction framework has been applied in [27]. It employs the LSTM network to analyse the temporal behavior, and predict the future coordinates of the surrounding vehicles. The DMVST-Net framework in [28] utilizes CNN and LSTM networks to model the spatial-temporal relation in taxi demand prediction. Besides, [29] proposes two RNN-based models to model trajectories on unique topological constraints, and a LC-RNN model which integrates RNN and CNN has been designed in [30] to achieve accurate traffic speed prediction. Furthermore, the incorporation of attention mechanism also becomes a common method [31], [32], [33]. For example, [34] not only integrates the CNN and LSTM, but also adaptively captures the dynamics and gaze shift behaviour of the region of interest to predict scanpaths. [35] combines soft attention and hard-wired attention to map the trajectory information from local neighbourhood to future positions of the pedestrian of interest. Particularly, [36] proposes an attentional RNN called DeepMove, for mobility prediction from lengthy and sparse trajectories to achieve the prediction of human mobility. That is similar to our problem. In addition, modifying the internal structure of the gating mechanism in LSTM is also very common [37]. A novel STDN framework has been proposed in [38] to involve a flow gating mechanism and a periodically shifted attention mechanism to solve the taxi demand problem. [39] introduces a new gating mechanism to learn the reliability of the 3D skeleton data to handle the noise and occlusion. However, as far as we know, there is no deep learning based destination prediction model yet. It is thus necessary to design a deep neural network model that can support multi-granularity trajectory modelling for accurate prediction of user destination.

7 Conclusion

This paper has studied the problem of destination prediction. A Location-Aware Attention mechanism has been first proposed to dynamically adapt to locations locally in trajectory and capture the distinct features of these locations to improve the prediction accuracy. Besides, a Trajectory-LSTM block with two time gates and two distance gates has been designed to model the long-term and short-term effects of time and distance intervals between two consecutive locations. Based on these two components, an LATL model has been further proposed to solve the data sparsity problem in fine-grained prediction. It can learn meaningful latent features and achieve accurate prediction. In addition, a hierarchical model called H-LATL has been introduced afterwards to explore and merge the mobility patterns learned from multiple spatial granularities. It combines different neural network layers to find a tailored structure of the prediction model under each granularity, so that the prediction accuracy can be further improved. Extensive experimental results on real datasets have demonstrated the effectiveness of our proposed models.

In the future, it would be interesting to investigate more advanced models with additional external features to be considered, or make an effective simplification of the H-LATL model.

Acknowledgments

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] H. Liu, J. Xu, K. Zheng, C. Liu, L. Du, and X. Wu, "Semantic-aware query processing for activity trajectories," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2017, pp. 283–292.

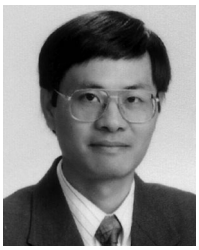
[2] J. Zhao, J. Xu, R. Zhou, P. Zhao, C. Liu, and F. Zhu, "On prediction of user destination by sub-trajectory understanding: A deep learning based approach," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1413–1422.

[3] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," in *Proc. IEEE Int. Conf. Data Eng.*, 2012, pp. 254–265.
[4] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[5] J. Jin and H. Nakayama, "Annotation order matters: Recurrent image annotator for arbitrary length image tagging," in *Proc. Int. Conf. Pattern Recognit.*, 2017, pp. 2452–2457.
[6] C. Weng, D. Yu, S. Watanabe, and B. F. Juang, "Recurrent deep neural networks for robust speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 5532–5536.
[7] Z. Li, Y. Wei, Y. Zhang, and Q. Yang, "Hierarchical attention transfer network for cross-domain sentiment classification," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 5852–5859.
[8] Z. Cheng, Y. Ding, X. He, L. Zhu, X. Song, and M. S. Kankanhalli, "A3NCF: An adaptive aspect attention model for rating prediction," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3748–3754.
[9] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013.
[10] D. Kong and F. Wu, "HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 2341–2347.
[11] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1764–1772.
[12] A. de Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio, "Artificial neural networks applied to taxi destination prediction," in *Proc. 2015th Int. Conf. ECML PKDD Discovery Challenge*, 2015, pp. 40–51.
[13] P. Zhao, H. Zhu, Y. Liu, Z. Li, J. Xu, and V. S. Sheng, "Where to go next: A spatio-temporal LSTM model for next POI recommendation," *CoRR*, vol. abs/1806.06671, 2018.
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.
[15] S. I. Park, J. Song, and Y. Kim, "A neural language model for multi-dimensional textual data based on CNN-LSTM network," in *Proc. 19th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distrib. Comput.*, 2018, pp. 212–217.
[16] S. Wang, L. Sun, W. Fan, J. Sun, S. Naoi, K. Shirahata, T. Fukagai, Y. Tomita, A. Ike, and T. Hashimoto, "An automated CNN recommendation system for image classification tasks," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2017, pp. 283–288.
[17] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 1571–1578.
[18] G. Best and R. Fitch, "Bayesian intention inference for trajectory prediction with an unknown goal destination," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 5817–5823.
[19] K. Jiang, N. Yu, and W. Li, "Online travel destination recommendation with efficient variable memory Markov model," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2013, pp. 1–4.
[20] J. Krumm and E. Horvitz, "Predestination: Inferring destinations from partial trajectories," in *Proc. Int. Conf. Ubiquitous Comput.*, 2006, pp. 243–260.
[21] D. Tiesyte and C. S. Jensen, "Similarity-based prediction of travel times for vehicles traveling on known routes," in *Proc. 16th ACM SIGSPATIAL Int. Conf. Advances Geograph. Inf. Syst.*, 2008, Art. no. 14.
[22] Z. Dong, J. Deng, X. Jiang, and Y. Wang, "RTMatch: Real-time location prediction based on trajectory pattern matching," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2017, pp. 103–117.
[23] K. Tanaka, Y. Kishino, T. Terada, and S. Nishio, "A destination prediction method using driving contexts and trajectory for car navigation systems," in *Proc. ACM Symp. Appl. Comput.*, 2009, pp. 190–195.
[24] H. T. Lam, E. Diaz-Aviles, A. Pascale, Y. Gkoufas, and B. Chen, "(Blue) taxi destination and trip time prediction from partial trajectories," in *Proc. 2015th Int. Conf. ECML PKDD Discovery Challenge*, 2015, pp. 63–74.
[25] F. D. N. Neto, C. de Souza Baptistaz, and C. E. C. Campelo, "A user-personalized model for real time destination and route prediction," in *Proc. Int. IEEE Conf. Intell. Transp. Syst.*, 2016, pp. 401–407.
[26] H. Yu and X. Zhu, "Recurrent neural network based rule sequence model for statistical machine translation," in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Language Process.*, 2015, pp. 132–138.
[27] B. Kim, C. M. Kang, J. Kim, S. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *Proc. Int. IEEE Conf. Intell. Transp. Syst.*, 2017, pp. 399–404.
[28] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2588–2595.
[29] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3083–3090.
[30] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, "LC-RNN: A deep learning model for traffic speed prediction," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3470–3476.
[31] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2015, pp. 1412–1421.
[32] X. Li, B. Zhao, and X. Lu, "MAM-RNN: Multi-level attention model based RNN for video captioning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 2208–2214.
[33] P. Ren, Z. Chen, Z. Ren, F. Wei, J. Ma, and M. de Rijke, "Leveraging contextual sentence relations for extractive summarization using a neural attention model," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 95–104.
[34] Z. Chen and W. Sun, "Scanpath prediction for visual attention using IOR-ROI LSTM," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 642–648.
[35] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft + hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection," *Neural Netw.*, vol. 108, pp. 466–478, 2017.
[36] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proc. Int. Conf. World Wide Web*, 2018, pp. 1459–1468.
[37] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-LSTM," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3602–3608.
[38] H. Yao, X. Tang, H. Wei, G. Zheng, Y. Yu, and Z. Li, "Modeling spatial-temporal dynamics for traffic prediction," *CoRR*, vol. abs/1803.01254, 2018.
[39] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal LSTM with trust gates for 3D human action recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 816–833.

Jiajie Xu received the MS degree from the University of Queensland, Australia, in 2006, and the PhD degree from the Swinburne University of Technology, Australia, in 2011. He is currently an associate professor with the School of Computer Science and Technology, Soochow University, China. His research interests include spatio-temporal database systems, big data analytics, and mobile computing.

Jing Zhao received the bachelor's degree in computer science and technology from Huaiyin Normal University, China, in 2017. She is currently working toward the postgraduate degree at Soochow University, China. Her research interests include spatial data mining and recommender systems.

Rui Zhou received the BS and MS degrees from Northeastern University, China, in 2004 and 2006, respectively, and the PhD degree from the Swinburne University of Technology, Australia, in 2010. He is currently a lecturer with the Swinburne University of Technology, Australia. His research interests include database systems, data mining, algorithms, and big data.



Chengfei Liu received the BS, MS, and PhD degrees in computer science from Nanjing University, China, in 1983, 1985, and 1988, respectively. He is currently a professor with the Department of Computer Science and Software Engineering, Swinburne University of Technology, Australia. His research interests include graph data management, keyword search on structured data, query processing and refinement for advanced database applications, and data-centric workflows. He is a member of the IEEE and ACM.

Pengpeng Zhao received the PhD degree in computer science from Soochow University, China, in 2008. He is currently an associate professor with the School of Computer Science and Technology, Soochow University, China. His research interests include data mining, deep learning, big data analysis, and recommender systems.

Lei Zhao received the PhD degree in computer science from Soochow University, China, in 2006. He is currently a professor with the School of Computer Science and Technology, Soochow University, China. His research interests include graph databases, social media analysis, query outsourcing, parallel, and distributed computing.