SASRM: A Semantic and Attention Spatio-temporal Recurrent Model for Next Location Prediction

1st Xu Zhang  
Department of Computer Science and Technology  
Chongqing University of Posts and Telecommunications  
Chongqing, China  
zhangx@cqupt.edu.cn

2nd Boming Li  
Department of Computer Science and Technology  
Chongqing University of Posts and Telecommunications  
Chongqing, China  
s170201094@stu.cqupt.edu.cn

3rd Chao Song  
National Engineering Laboratory for Public Safety Risk Perception and Control by Big Data(PSPRC)  
China Academy of Electronics and Information Technology  
Beijing, China  
songch35@mail.ustc.edu.cn

4th Zhengwen Huang  
Department of Electronic and Computer Engineering  
Brunel University London  
London, United Kingdom  
zhengwen.huang@brunel.ac.uk

5th Yan Li  
Department of Computer Engineering  
Inha University  
Incheon, South Korea  
leeyeon@inha.ac.kr

Abstract—Predicting user’s next location is of great importance for a wide spectrum of location-based applications. However, most prediction methods do not take advantage of the rich semantic information contained in trajectory data. Meanwhile, the traditional LSTM-based model can not capture the spatio-temporal dependencies well. In this paper, we propose a Semantic and Attention Spatio-temporal Recurrent Model (SASRM) for next location prediction. Firstly, the SASRM put forward a method for encoding semantic vectors and concatenating vectors (location, time and semantic vectors) as input to the model. To capture the spatio-temporal dependencies, we design a variant recurrent unit based on LSTM. Further, an attention layer is used to weight hidden state to capture the influence of the historical locations on the next location prediction. We perform experiments on two real-life semantic trajectory datasets, and evaluation results demonstrate that our model outperforms several state-of-the-art models in accuracy.

Index Terms—Location Prediction, Semantic Trajectory, Attention, LSTM

I. INTRODUCTION

With the development of mobile positioning technology, people’s trajectories [1] are abundantly preserved. Spatial and temporal contextual information plays a key role for analyzing user behaviors and is helpful for predicting where he or she will go next [2] [3]. For example, we can provide route recommended [4], location advertisement recommendation [5] and urban traffic planning [6] based on prediction of the future locations people tend to visit. Human mobility trajectories enriched with spatial and temporal contextual information are called semantic trajectories. Twitter [7], Foursquare and Instagram allow users to record their locations as well as their semantic information, such as location contextual information (e.g. restaurants, bookstores), ongoing activity contextual information (playing basketball, dancing, singing). This semantic information also has a greater impact on the user’s next location prediction, which was not fully considered in most existing work. The famous recurrent network model LSTM [8] was originally designed for language model modeling [9], which has been introduced into the field of trajectory data analysis and achieved good results. Trajectory sequences have the same properties as language sequences. As shown in Fig.1, compared with the language sequence $W_i$, the trajectory sequence $P_i$ includes spatio-temporal contextual information.
such as time $\Delta t$ and distance $\Delta d$ of movement. Therefore, more and more methods do trajectory prediction task based on LSTM. The attention technique [10] has been applied on LSTM to find the relevant part of the information that helps generate a better output. The combination of the recurrent network model and attention technique improves the performance of many challenging tasks, such as machine translation [11], generation of image captions [12], video description [13], and speech recognition [14].

Location prediction from semantic trajectories is not trivial because of three challenges: 1) How to encode the semantic information into a prediction model. 2) How to effectively capture the spatio-temporal dependencies of the trajectories. 3) How to incorporate historical information in recurrent model training. To solve these challenges, we propose a Semantic and Attention Spatio-temporal Recurrent Model (SASRM) for next location prediction.

In general, our contributions are in the following areas:

- Different from the previous method, we adopt the sen2vec method to encode the semantic vector and concatenate location, time and semantic vectors as the part of recurrent unit input.
- We design a variant recurrent unit with a time gate and a distance gate based on LSTM to capture spatio-temporal dependencies effectively. In addition, an attention layer is used to incorporate historical information into recurrent model training to enhance the role of historical information.
- Experimental evaluation on two real-life semantic trajectory datasets shows that SASRM model outperforms several state-of-the-art models in accuracy.

The rest of this paper is organized as follows. We first discuss the related work in location prediction in section II. After that, we formulate the problem and briefly introduce the principle of the LSTM model. III. Following the preliminaries, we introduce details of the architecture of SASRM in section IV and apply our model on two real-world semantic trajectory datasets and conduct extensive analysis on the performance in section V. Finally, we conclude our paper in section VI.

II. RELATED WORK

A. Traditional Prediction Models

Most of the existing works consider human mobility patterns. The two-stage algorithm Periodica was proposed by Li et al. [15] to generalize the user’s periodic behavior and predicts the user’s location movement. Zhang et al. [16] proposed a user pattern mining method (Splitter) based on semantic trajectory. This method considers that the traditional algorithms cannot effectively mine the patterns in the semantic trajectory then decomposes the spatial coarse-grained patterns into fine-grained patterns by top-down method and group users to predict. Mayhew et al. [17] proposed a prediction method based on Hidden Markov Model (HMM). This method clustered user locations according to the historical information, and then trained an HMM model for each cluster, which greatly improved the accuracy of location prediction. Duong-Trung et al. [18] describe a content-based generative regression model, which used matrix factorization technology to solve the prediction problem. Feng et al. [3] proposed a personalized ranking metric embedding method (PRME) to model personalized check-in sequences for next POI recommendation. Unfortunately, these traditional methods cannot deal with the long-term dependence of trajectories, and some of them do not consider the semantic information in trajectories or consider it weakly.

B. Neural Network Models

Recently, recurrent neural networks are widely adopted for location prediction. Liu et al. [2] proposed ST-RNN (Spatial Temporal RNN) model using time-specific transition matrix and distance specific transition matrix based on RNN unit to enhance the ability of model to receive spatio-temporal information. In Zhu et al. [19], a model called Time-LSTM was generated for time interval to learn the temporal dependencies. Based on Time-LSTM, Zhao et al. [20] put forward STGN model. They designed a gate recurrent network based on LSTM unit by adding time and distance gates, which improved the ability of LSTM model to capture spatio-temporal dependencies. Regrettably, these models hardly ever consider the rich semantic information in the trajectory data. The SERM [21] model uses a bag of keywords method to encode semantic information and set pre-trained word vector Glove [22] as the weight of semantic vectors, which improves the next location prediction accuracy. However, if the number of words in the trainset is too large, the dimension of the word vector will become very large (assuming the number of candidate words is 100000, and then the generated semantic vector will reach 100000). The size of trainset will also become huge. It is difficult for SERM model to train and adjust. Finally, these models hardly ever consider how to incorporate historical information in recurrent model training, for enhancing the role of historical information.

III. PRELIMINARIES

In this section, we first formally formulate the next location prediction problem, and then briefly introduce the principle of the LSTM model.

A. Problem Definition

Consider a set of users $U = \{u_1, u_2, \cdots, u_M\}$ and a set of locations $L = \{l_1, l_2, \cdots, l_N\}$. The candidate locations $N$ may be points-of-interests (POI) or equally-sized grids. In this paper, we use grids to represent the locations. Then, we use the following formula to define semantic trajectories.

**Definition 1: (Semantic Location)** A location record of a user at one time can be represented as a four-tuple $t^u_k = (u_i, l_k, t_k, c_k)$, where $u_i$ indicates user ID, $l_k$ indicates the current location of user $u_i$ at time $t_k$, $c_k$ indicates text message either recorded by LBS provider or user, which contains semantic information.
Definition 2: (Semantic Trajectory) A user’s semantic trajectory is expressed as $T^{ui} = \{r_1^{ui}, r_2^{ui}, \ldots, r_K^{ui}\}$, s.t. $\forall 1 \leq k < K$, $0 < t_{k+1} - t_k < \Delta t_g$. $\Delta t_g$ is a time gap to truncate trajectory because some locations are separated by very long time, they need to be truncated into two different trajectories.

Now we formally describe our issue to predict the user’s next location on semantic trajectories. Given a historical trajectories.

LSTM is shown on (1)-(6).

IV. PROPOSED APPROACH

Fig. 2 presents the architecture of the proposed SASRM model. It consists of four major components: 1) feature embedding and vector generator; 2) recurrent unit; 3) historical attention; and 4) prediction.

Features extracted from semantic trajectory is encoded with a vector generator in phase IV-A. The second part introduces the recurrent unit we designed. Then in third part, tells the implementation method of the attention layer. Finally, we describe how to adapt user information and training method in Prediction.
A. Feature Embedding and Vector Generator

Semantic trajectory is comprised of a sequence of semantic locations as shown in section III-A. We adopt a semantic embedding method called sen2vec to encode semantic vectors. For one text information $c_k$ contain semantics, we use the pre-trained word vector $v_{gk}$ (here we use Glove [22]) to represent each word $w$. Firstly, all word vectors $v_{gk}$ are weighted by $a$ and $p(w)$. Here $a$ is pre-defined parameter and default setting is 0.0001. $p(w)$ is the frequency of the word appearing in the corpus. Then average sum to vector $v_{sk}$ in (10). If $p(w)$ is larger (the word appears a lot of times), the proportion of the corresponding $v_{gk}$ in vector $v_{sk}$ would be smaller. This highlights the impact of personalized vocabulary on the resulting vector. Then we use PCA (principal component analysis) [25] to find the principal component $\hat{u}$ of the $S$ in (11) and compute $v_{ck}$ in (12). $S$ is the set of intermediate vectors $v_{sk}$. Next, we obtain the embedding vector $b_{ck}$ by linear transformation of $v_{ck}$. It is worth noting that we finally get that vector $b_{ck}$ will maintain the same dimension as $b_{li}, b_{lk}$ and $b_{ck}$. The custom dimension is good for training and contain rich semantic information.

$$v_{sk} = \frac{1}{|c_k|} \sum_{w \in c_k} \frac{a}{a + p(w)} v_{gk}$$  \hspace{1cm} (10)

$$\hat{u} = \text{PCA}(S)$$ \hspace{1cm} (11)

$$v_{ck} = v_{sk} - \hat{u}^T v_{sk}$$  \hspace{1cm} (12)

$$b_{ck} = v_{ck} \cdot B_c$$ \hspace{1cm} (13)

Finally, $b_{li}, b_{lk}$ and $b_{ck}$ are concatenated as the input vector of the recurrent layer in (14).

$$x^S_k = \text{concat}(b_{li}, b_{lk}, b_{ck})$$ \hspace{1cm} (14)

B. Recurrent Unit

As a variant of RNN, LSTM is capable of learning short-term and long-term dependencies. It has become an effective and scalable model for sequential prediction problems in the trajectory mining area and many improvements have been made to the original LSTM architecture. In this section, inspired by STGN [20], we re-designed the basic LSTM unit which utilizes time and distance intervals to receive time interval and distance interval information simultaneously. Unlike STGN, which has multiple gates to capture spatio-temporal dependencies, we only set a time gate and a distance gate to obtain the main spatio-temporal dependencies. In experiments, we explored using multiple gates (such as STGN) or not using gates, but the performance was poor. So we only use separate gates to capture dependencies. The time interval $\Delta t$ is equal to $t_{n+1} - t_n$. The distance interval $\Delta d$ is equal to $d(t_{n+1}, t_n)$. $d(\cdot, \cdot)$ denotes the geographic distance between the grid centers of two locations. Therefore, the recurrent unit used in the SASRM is shown in (15)-(23):

$$i_n = \sigma(x^S_n W_i + h_{n-1} V_i + b_i)$$ \hspace{1cm} (15)

$$f_n = \sigma(x^S_n W_f + h_{n-1} V_f + b_f)$$ \hspace{1cm} (16)

$$\tilde{c}_n = \tanh(x^S_n W_t \hat{c} + h_{n-1} V_c + b_c)$$ \hspace{1cm} (17)

$$T_n = \sigma(x^S_n W_t + \sigma(\Delta t_n W_t) + b_t)$$ \hspace{1cm} (18)

$$D_n = \sigma(x^S_n W_d + \sigma(\Delta d_n W_d) + b_d)$$ \hspace{1cm} (19)

$$c_n = f_n \odot c_{n-1} + i_n \odot T_n \odot D_n \odot \tilde{c}_n$$ \hspace{1cm} (20)

$$\tilde{c}_n = f_n \odot c_{n-1} + i_n \odot T_n \odot D_n \odot \tilde{c}_n$$ \hspace{1cm} (21)

$$o_n = \sigma(x^S_n W_o + h_{n-1} V_o + \Delta t_n W_{to} + \Delta d_n W_{do} + b_o)$$ \hspace{1cm} (22)

$$h_n = o_n \odot \tanh(\tilde{c}_n)$$ \hspace{1cm} (23)

Here $i_n$ is the input gate, $f_n$ is the forgetting gate, $o_n$ is the input gate same as basic LSTM unit. $W_i, W_f, W_t, W_o, W_{xt}$ and $W_{zd}$ are the weight vectors of the input vector $x^S_n, V_i, V_f, V_t$ and $V_o$ are the weight vectors of the hidden input vector $h_{n-1}, W_i, W_{to}, W_{do}$ are the weight vectors of time interval and distance interval, respectively. $b_i, b_f, b_t, b_o, b_t$ and $b_d$ are the bias of the gate vectors. $\tilde{c}_n$ is the intermediate memory state which are computed memory state $c_n$ and $\tilde{c}_n$. 
The recurrent unit aims to capture the complicated sequential information capture the spatio-temporal dependencies in the user’s trajectories, which helps to better predict the user’s next location. The improved unit structure is shown in Fig. 3, in which a red rectangle contains the time gate and the distance gate.

**C. Historical Attention**

The recurrent unit aims to capture the complicated sequential information and spatio-temporal dependencies contained in the current trajectory. An attention layer is designed to capture mobility regularity from the historical records. It takes the historical hidden state as the input and outputs the most related context vector when queried by a query vector from the recurrent.

In our attention layer, the hidden state vectors are weighted by two additional attention weight matrix $W_{x1} \in \mathbb{R}^{c \times e}$, $W_{x2} \in \mathbb{R}^{d \times e}$ and a attention depth $d_r$ (the dimension of the hidden state $h_n$ is $e$). In (24), the $H^i$ represents the part of hidden vectors $[h_{i-d_r}, \cdots, h_{i-1}]$ that needs to be weighted. As shown in Fig. 4 and (25), the $H^i$ is used to calculate the alignment weight vector $\alpha_i$.

$$H^i = [h_{i-d_r}, \cdots, h_{i-1}], n > i > d_r$$  

$$\alpha_i = \text{softmax} \left( \tanh \left( H^i W_{x1} \right) W_{x2} \right)$$  

$$\overline{h}_i = \begin{cases} 
  h_i, & i < d_r \\
  \alpha_i^T \cdot H^i, & n \geq i \geq d_r
\end{cases}$$

According to alignment weight vector $\alpha_i$, we score the $H^i$ to find a new context vector $\overline{h}_i$. In (26), if $i$ is greater than the attention depth $d_r$, then $\overline{h}_i$ is equal to $\alpha_i^T \cdot H^i$, otherwise $\overline{h}_i$ equal to the original hidden vector $h_i$. The newly calculated $\overline{h}_i$ is related to the historical vector $[h_{i-d_r}, \cdots, h_{i-1}]$, thereby incorporating historical information into recurrent model training and enhancing the role of historical information.

**D. Prediction**

To reflect the user’s personalized information, we add the user ID vector $b_u$. Because the information does not change with time during the training of the whole model, it is not inputted into the recurrent structure for training but is added to the output of the linear layer. Equation (27) is a linear transformation to transform $\overline{h}_i$ to $N$ dimensional vector $o_k$ ($N$ is total locations of prediction). Finally, we add $o_k$ and $b_u$ as the input of softmax to obtain vector $y_k$ in (28).

$$o_k = \overline{h}_i \cdot b_N + b_N$$  

$$y_k = \text{softmax} (o_k + b_u)$$

We use cross entropy as the loss function to calculate the loss value of each last location $l_{k+1}$ with vector $y_k$ in (29), $J$ is the loss value.

$$J = - \sum_{k=1}^{K} l_{k+1} \log (y_k)$$

In training, we use RMSprop [26] (a variant of the stochastic gradient descent algorithm) and a time-backward propagation algorithm (BPTT) to update the parameters of the SASRM. The parameters that need to be updated include $\{B_1, B_t, B_u, B_c, B_N, W_1, W_f, W_c, W_o, W_{xt}, W_{xd}, W_t, W_d, W_to, W_{do}, V_t, V_f, V_c, V_o, W_{x1}, W_{x2}, b_1, b_f, b_c, b_o, b_t, b_d, b_N\}$.

**V. EXPERIMENTS**

**A. DataSets**

We conduct experiments on two real-life semantic trajectory datasets: New York City(NY) and Los Angeles(LA). The dataset NY [16] contains 0.3 million check-in data records at New York from Jan. 2011 to Jan. 2012. The dataset LA [27] included 1.2 million tweets sign-in data records between Aug. 2014 and Nov. 2014. These two data sets record the user’s check-in information in the form of coordinate points, and each record contains fields such as the user’s id, geographic coordinates, time and text description. We indicate the city into grids, respectively, each grid representing a location, such
as dividing the NY into $100 \times 100$ grids. In terms of data pre-processing, we removed users with less than 50 checkin records, truncate trajectories with a limit of 10 hours and each user needs to include at least 3 complete trajectories. Finally, on the NA dataset, we retained 3,107 trajectories for 235 users. On the LA dataset, we retained 8,691 trajectories for 466 users. More datasets details show in Table I.

**B. Baseline Methods**

Three state-of-the-art recent approaches are employed for comparisons in this paper. And we also evaluate with a semantic free model SASRM*, which is a variant of SASRM.

1) MF: The most frequent method, predicting the next location based on the user’s historical frequency.
2) LSTM [8]: The basic LSTM model, the basic LSTM unit are used, location and user vector as input.
3) SERM [21]: The rich semantic neural network model based on LSTM, using a bag of keywords method to encode semantic vectors.
4) STGN [20]: A Spatio-temporal Gate Network for predicting the user’s next location by adding time and distance gates based on LSTM unit.
5) SASRM*: A variant of the SASRM model that removes the semantic module (input vector reserved location and time) and all others are retained.

**C. Parameter Settings and Metrics**

All of the recurrent model use the uniform initialization, the initialization range $(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$, and $e$ is the dimension of the hidden vector $h_n$. Compared with other methods in Table II, we set the dimensions of the vectors $b_{l_k}, b_{l_s}, b_{c_k}$ and $e$ to be 50, attention depth $d_e=3$, and grid size is $100 \times 100$ in two datasets.

For each dataset, we take 80% of the data as trainset and 20% as testset. At the beginning of the training, the learning rate was set to 0.05, and then decreased to 0.0001 for fine-tuning. The number of max iterations is 500 and all methods get the excellent training results.

We adopt two metrics to evaluate our model. The first metric is the accuracy Acc@K. The Acc@K rate for the entire experiment is the proportion of all test data that successfully appeared in the predicted top-K set. The second metric is the predicted distance $\delta_d/m$, which represent the distance (the unit of measurement is meters) between the ground-truth location and the top-5 prediction.

**D. Performance Evaluation**

As it is shown in Table I, SASRM* and SASRM achieved the better accuracy on Acc@1, Acc@5, Acc@10 and $\delta_d/m$ in NY. Although the SERM get good results in Acc@15 and Acc@20, we are more concerned about top 1-10 in practice. In LA, SASRM* and SASRM achieved significant improvement in all metrics.

The MF method belongs to the traditional prediction method, which predicts next location only depending on the user’s historical location frequency. It has poor performance compared with neural network methods. The LSTM method uses the basic LSTM unit and it is a benchmark method in recurrent neural network model. Obviously, it does not contain user semantic information, and has a weak spatio-temporal dependencies. It performs generally. For SERM model, it contains rich semantic information but ignores spatio-temporal dependencies and its effect is slightly better than LSTM method. Although STGN captures the spatio-temporal relationship very strongly, ignores the rich semantic information contained in trajectory. It has the same effect on two datasets as SERM model and does not have the best ranking. SASRM*, thanks to the attention layer and recurrent units with distance gate and time gate, compared to SERM and STGN, its accuracy has increased.

SASRM*, our proposed complete model, considers spatio-temporal dependencies and semantic information, and uses an attention layer to capture the influence of the historical location. It achieved the best results on both datasets.

**E. Impact of Parameters**

In general, different sizes of hidden vector $h_n$ will have different effects on results. The accuracy of different hidden size $e$ of SASRM with experiments on the benchmark datasets of NY and LA is illustrated on Table III. The parameter $e$ for the size of the recurrent unit hidden state is searched in $[50, 100, 150, 200, 250]$. As it is shown in Table III, with the increases of hidden size, Acc@1 increases slightly, and the increase in Acc@5 and Acc@10 is relatively obvious. In
In this paper, we propose a Semantic and Attention Spatio-temporal Recurrent Model (SASRM) for next location prediction. In vector generator part, we adopt a sen2vec method for encoding semantic vectors and concatenate location, time and semantic vectors as the model input. About the recurrent unit part, we designed a variant recurrent unit based on LSTM to facilitate the capture of spatio-temporal dependencies. Then, we added an attention layer to incorporate historical information into recurrent model training. Experimental results on two real-life semantic trajectory datasets show that SASRM outperforms several state-of-the-art models in accuracy. In the future work, we hope to consider how to integrate the attention mechanism with user habits to further improve the prediction accuracy of the user’s next location.

ACKNOWLEDGMENT

This research is supported in part by National Key Research and Development Project (Grant No. 2017YFC0820502), the Director Foundation Project of National Engineering Laboratory for Public Safety Risk Perception and Control by Big Data (PSRPC), National Natural Science Foundation of China (41571401) and Chongqing Natural Science Foundation (cstc2014jrcqnrc40002).

REFERENCES


