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**

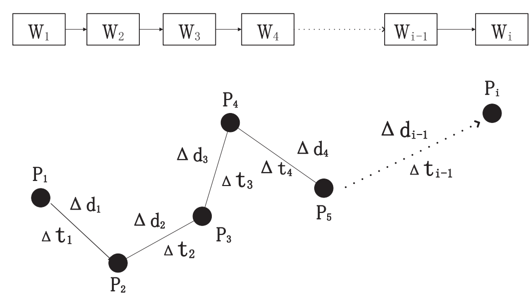

Fig. 1. Language sequence and trajectory sequence.

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 Δt and distance Δd 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 modelIII. 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 generalizes 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 $\mathbb{U} = \{u_1, u_1, \cdots, u_M\}$ and a set of locations $\mathbb{L} = \{l_1, l_2, \cdots, l_N\}$. 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 $r_k^{u_i} = (u_i, l_k, t_k, c_k)$, 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.

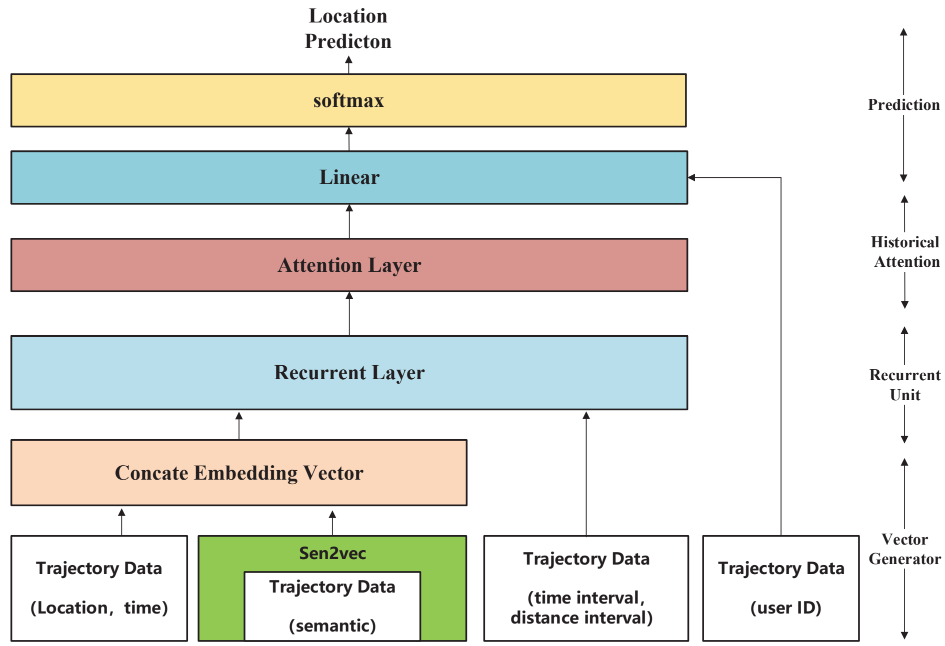

Fig. 2. Overall framework of the SASRM.

Definition 2: **(Semantic Trajectory)** A user's semantic trajectory is expressed as $\mathcal{T}^{u_i} = \{r_1^{u_i}, r_2^{u_i}, \cdots, r_K^{u_i}\}$, s.t. $\forall 1 \leq k < K$, $0 < t_{k+1} - t_k < \Delta t_g$. Δt_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 semantic trajectory $\mathcal{T}^{u_i} = \{r_1^{u_i}, r_2^{u_i}, \cdots, r_{k-1}^{u_i}\}$, the next location l_k is predicted from the set of locations $\mathbb{L}$.

B. LSTM

LSTM [8] is a classical variant of RNN, which is called long-term and short-term memory network. In order to solve the problem of gradient dispersion of RNN model, LSTM is proposed. Compared with RNN, it can capture long-term dependence in sequence information and has good scalability. We can modify LSTM to improve the performance of the model for different prediction tasks. The formula of original LSTM is shown on (1)-(6).

$$i_n = \sigma\left(x_n W_i + h_{n-1} V_i + b_i\right) \tag{1}$$

$$f_n = \sigma\left(x_n W_f + h_{n-1} V_f + b_f\right) \tag{2}$$

$$\tilde{c}_n = \tanh\left(x_n W_{\tilde{c}} + h_{n-1} V_{\tilde{c}} + b_{\tilde{c}}\right) \tag{3}$$

$$c_n = f_n \odot c_{n-1} + i_n \odot \tilde{c}_n \tag{4}$$

$$o_n = \sigma\left(x_n W_o + h_{n-1} V_o + b_o\right) \tag{5}$$

$$h_n = o_n \odot \tanh\left(c_n\right) \tag{6}$$

Here i_n represents the input gate, storing some information of the current time, f_n represents the forgetting gate, choosing to forget some information of the past, o_n represents the output gate. x_n and h_n represent input vector and hidden output vector in $n{-}th$, respectively. σ indicates the activation function $sigmod$, which is the threshold function of the neural network. It maps values between 0 and 1. $\tanh$ is a hyperbolic tangent function that maps values between -1 and 1. W_i, W_f, W_c and W_o are the weight vector of the input vector x_n. V_i, V_f, V_c and V_o are the weight vector of the hidden input vector h_{n-1}. b_i, b_f, b_c and b_o are the bias of the gate vector. $\odot$ represents the dot product (Hadamard product) of two matrices. c_n represents the state of cell and is used to update the hidden state h_n. The updating of c_n consists of two parts, one is the dot product of f_n and c_{n-1}, the other is the dot product of i_n and $\tilde{c}_n$.

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 show in section.III-A. We use adopt embedding technology [23] to generate vectors, which is similar like nature language processing. The part of a semantic location l_k, t_k and u_i can be directly encoded to the embedding vectors b_{l_k}, b_{t_k} and b_{u_i} by the embedding weight matrices B_l, B_t and B_u, here three matrices are initialized randomly, shown in (7)-(9).

$$b_{l_k} = l_k \cdot B_l \tag{7}$$

$$b_{t_k} = t_k \cdot B_t \tag{8}$$

$$b_{u_i} = u_i \cdot B_u \tag{9}$$

Inspired by Arora et al. [24], we adopt a semantic information vector embedding method called sen2vec to encode semantic vectors. For one text information c_k contain semantics, we use the pre-trained word vector v_{g_k} (here we use Glove [22]) to represent each word w. Firstly, all word vectors v_{g_k} are weighted by a and $p\left(w\right)$, Here a is pre-define parameter and default setting is 0.0001. $p\left(w\right)$ is the frequency of the word appear in the corpus. Then average sum to vector v_{s_k} in (10). If $p\left(w\right)$ is larger (the word appears a lot of times), the proportion of the corresponding v_{g_k} in vector v_{s_k} 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 $\tilde{u}$ of the $\tilde{S}$ in (11) and compute v_{c_k} in (12). $\tilde{S}$ is the set of intermediate vectors v_{s_k}. Next, we obtain the embedding vector b_{c_k} by linear transformation of v_{c_k}. It is worth noting that we finally get that vector b_{c_k} will maintain the same dimension as b_{l_k}, b_{t_k} and b_{c_k}. The custom dimension is good for training and contain rich semantic information.

$$v_{s_k}=\frac{1}{|c_k|}\sum\nolimits_{w\in c_k}\frac{a}{a+p\left(w\right)}v_{g_k} \tag{10}$$

$$\tilde{u}=PCA\left(\tilde{S}\right) \tag{11}$$

$$v_{c_k}=v_{s_k}-\tilde{u}\tilde{u}^T v_{s_k} \tag{12}$$

$$b_{c_k} = v_{c_k} \cdot B_c \tag{13}$$

Finally, b_{l_k}, b_{t_k} and b_{c_k} are concatenated as the input vector of the recurrent layer in (14).

$$x_k^S = concat\left(b_{l_k}, b_{t_k}, b_{c_k}\right) \tag{14}$$

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 Δt is equal to $t_{n+1}-t_n$. The distance interval Δd is equal to $d\left(l_{n+1}, l_n\right)$, $d\left(\cdot,\cdot\right)$ denotes the geographic distance between the grid centers of two locations. Therefore, the recurrent unit used in the SASRM is shown in (15)-(23):

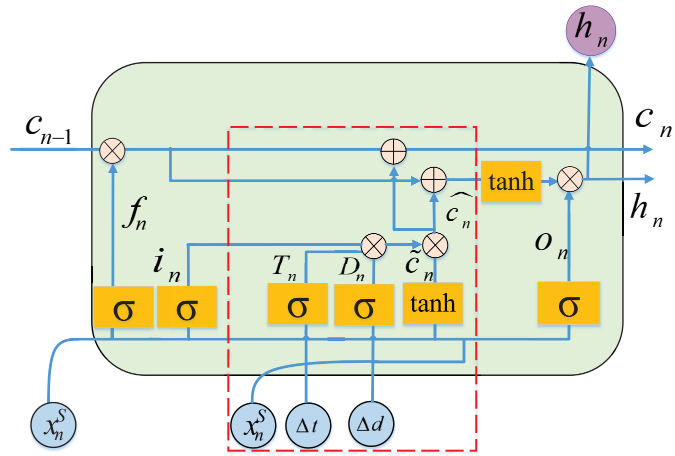

Fig. 3. The re-designed recurrent unit.

$$i_n=\sigma\left(x_n^S W_i + h_{n-1}V_i + b_i\right) \tag{15}$$

$$f_n=\sigma\left(x_n^S W_f + h_{n-1}V_f + b_f\right) \tag{16}$$

$$\tilde{c}_n=\tanh\left(x_n^S W_{\tilde{c}} + h_{n-1}V_{\tilde{c}} + b_{\tilde{c}}\right) \tag{17}$$

$$T_n=\sigma\left(x_n^S W_{xt} + \sigma(\Delta t_n W_t) + b_t\right) \tag{18}$$

$$D_n=\sigma\left(x_n^S W_{xd} + \sigma(\Delta d_n W_d) + b_d\right) \tag{19}$$

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

$$\widehat{c}_n = f_n \odot c_{n-1} + i_n \odot T_n \odot D_n \odot \tilde{c}_n \tag{21}$$

$$o_n=\sigma\left(x_n^S W_o + h_{n-1}V_o + \Delta t_n W_{to} + \Delta d_n W_{do} + b_o\right) \tag{22}$$

$$h_n = o_n \odot \tanh\left(\widehat{c}_n\right) \tag{23}$$

Here i_n is the input gate, f_n is the forgotting gate, o_n is the input gate same as basic LSTM unit. W_i, W_f, $W_{\tilde{c}}$, W_o, W_{xt} and W_{xd} are the weight vectors of the input vector x_n^S. V_i, V_f, $V_{\tilde{c}}$ and V_o are the weight vectors of the hidden input vector h_{n-1}. W_t, W_{to}, W_d and W_{do} are the weight vectors of time interval and distance interval, respectively. b_i, b_f, $b_{\tilde{c}}$, 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 $\widehat{c}_n$.

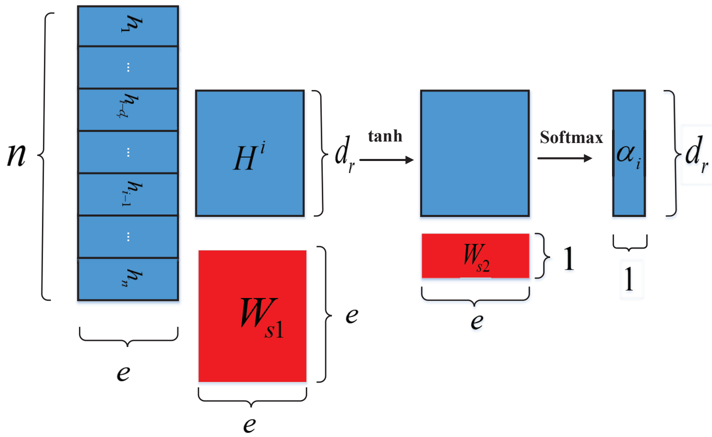

Fig. 4. Calculate the alignment weight vector.

Equation (18) use x_n^S and Δt compute the time gate T_n. Equation (19) use x_n^S and Δd compute the distance gate D_n. The memory state c_n contained Δt_n and Δd_n dependencies can be passed to c_{n+1}, c_{n+2}, ... in (20). The $\widehat{c}_n$ is used to compute the hidden state h_n in (21). At the same time, the Δt_n and Δd_n participated in the calculation of o_n in (22). These gates contain Δt_n and Δd_n 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 $W_{s1} \in \mathrm{R}^{e*e}$, $W_{s2} \in R^{e*1}$ 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 α_i.

$$H^i = [h_{i-d_r}, \cdots, h_{i-1}], n > i \geq d_r \tag{24}$$

$$\alpha_i = softmax\left(\tanh\left(H^i W_{s1}\right) W_{s2}\right) \tag{25}$$

$$\overline{h_i} = \begin{cases} h_i, i < d_r \\ {\alpha_i}^T \cdot H^i, n \geq i \geq d_r \end{cases} \tag{26}$$

According to alignment weight vector α_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_i}. 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_i} as the input of $softmax$ to obtain vector y_k in (28).

$$o_k = \overline{h_i} \cdot B_N + \mathrm{b}_N \tag{27}$$

$$y_k = softmax\left(o_k + b_{u_i}\right) \tag{28}$$

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.

$$J = -\sum_{k=1}^{K} l_{k+1} \log\left(y_k\right) \tag{29}$$

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 $\{B_l, B_t, B_u, B_c, B_N, W_i, W_f, W_{\tilde{c}}, W_o, W_{xt}, W_{xd}, W_t, W_d, W_{to}, W_{do}, V_i, V_f, V_{\tilde{c}}, V_o, W_{s1}, W_{s2}, b_i, b_f, b_{\tilde{c}}, b_o, b_t, b_d, b_N\}$.

TABLE I
STATISTICS OF TWO DATASETS

Dataset	NY	LA
City	New York	Los Angeles
Duration	1 year	4 months
Records (raw)	293558	1186852
Locations (raw)	7583	96237
Users (processed)	235	466
Trajectories (processed)	3107	8691
Tarj_max_len (processed)	84	78
Tarj./User	13	18

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

TABLE II
PERFORMANCE EVALUATION FOR DIFFERENT METHODS

Dataset	Method	Acc@1	Acc@5	Acc@10	Acc@15	Acc@20	δ_d/m
NY	MF	0.1450	0.3078	0.3762	0.4004	0.4124	2237
	LSTM	0.1710	0.4225	0.5433	0.6016	0.6539	1582
	SERM	0.2012	0.4004	0.4869	**0.6197**	**0.6539**	1519
	STGN	0.1972	0.4064	0.5231	0.5855	0.6338	1626
	SASRM*	0.1972	0.4346	0.5513	0.6056	0.6499	1533
	SASRM	**0.2052**	**0.4487**	**0.5533**	0.6036	0.6318	**1419**
LA	MF	0.2774	0.4161	0.4603	0.4707	0.4783	3959
	LSTM	0.2478	0.4865	0.5700	0.6163	0.6494	2425
	SERM	0.2698	0.4976	0.5721	0.6142	0.6446	2497
	STGN	0.2664	0.4893	0.5728	0.6135	0.6473	2508
	SASRM*	0.2816	0.5148	0.5942	0.6370	0.6618	2352
	SASRM	**0.2905**	**0.5204**	**0.5949**	**0.6398**	**0.6625**	**2341**

as dividing the NY into 100×100 grids. In terms of data pre-processing, we removed users with less than 50 check-in 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 $\left(-\frac{1}{e}, \frac{1}{e}\right)$, 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_{t_k}, b_{c_k} and e to be 50, attention depth d_r= 3, and grid size is 100×100 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 δ_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 δ_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

TABLE III
THE INFLUENCE OF DIFFERENT HIDDEN SIZE ON ACCURACY

Dataset	HiddenSize e	Acc@1	Acc@5	Acc@10
NY	50	0.2052	0.4487	0.5532
	100	0.2032	0.4447	0.5633
	150	0.2032	0.4507	0.5674
	200	0.2052	0.4527	0.5694
	250	0.2072	0.4547	0.5815
Dataset	**HiddenSize e**	**Acc@1**	**Acc@5**	**Acc@10**
LA	50	0.2905	0.5204	0.5949
	100	0.2899	0.5176	0.5963
	150	0.2905	0.5210	0.6052
	200	0.2940	0.5218	0.6004
	250	0.2960	0.5200	0.6011

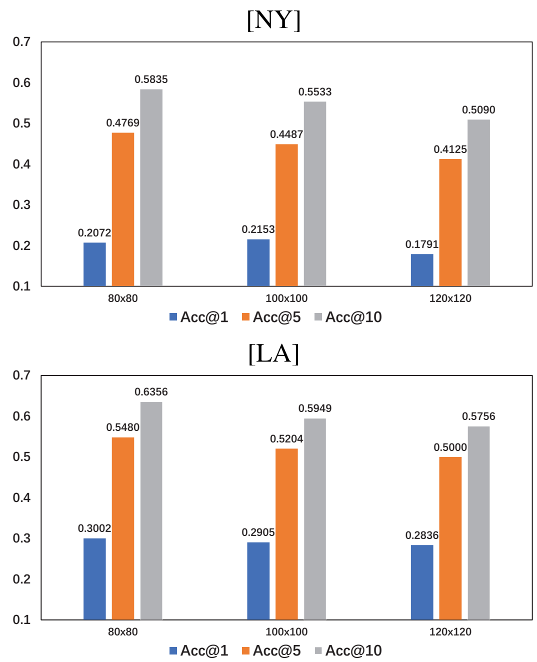

Fig. 5. Accuracy changes with grid size on two datasets.

general, the accuracy of SASRM increases on the both datasets with increases of hidden size e. For the larger hidden size provides more model parameters when the training data is adequate, which results in higher performance. Moreover, we also investigate effects of different grid sizes on the model performance. The size of grid is searched in $[80 \times 80, 100 \times 100, 120 \times 120]$. When the grid size is 100×100, the grid length and width of the NY dataset is about 500×500 meters, and the length and width of the LA dataset is about 600×600 meters. As it is shown in Fig.5, the accuracy of SASRM declines on the two datasets with increase of grid size. For the larger size of grid indicates the more candidates prediction location chosen, which leads to the poor prediction of next location. The experimental results of these two parameters show that the grid size and the dimension of hidden layer vector should be adjusted according to the needs of the actual prediction task, so as to bring appropriate prediction performance.

VI. CONCLUSIONS

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 (cstc2014kjrcqnrc40002).

REFERENCES

[1] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 29, 2015.

[2] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[3] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new poi recommendation," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[4] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015, pp. 543–554.

[5] L. Deng, J. Gao, and C. Vuppalapati, "Building a big data analytics service framework for mobile advertising and marketing," in *2015 IEEE First International Conference on Big Data Computing Service and Applications*. IEEE, 2015, pp. 256–266.

[6] P. S. Castro, D. Zhang, and S. Li, "Urban traffic modelling and prediction using large scale taxi gps traces," in *International Conference on Pervasive Computing*. Springer, 2012, pp. 57–72.

[7] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: quantifying influence on twitter," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 65–74.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 5528–5531.

[10] K. Cho, A. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1875–1886, 2015.

[11] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[12] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[13] C. Hori, T. Hori, T.-Y. Lee, Z. Zhang, B. Harsham, J. R. Hershey, T. K. Marks, and K. Sumi, "Attention-based multimodal fusion for video description," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4193–4202.

[14] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

[15] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye, "Mining periodic behaviors for moving objects," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1099–1108.

[16] C. Zhang, J. Han, L. Shou, J. Lu, and T. La Porta, "Splitter: Mining fine-grained sequential patterns in semantic trajectories," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 769–780, 2014.

[17] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden markov models," in *Proceedings of the 2012 ACM conference on ubiquitous computing*. ACM, 2012, pp. 911–918.

[18] N. Duong-Trung, N. Schilling, and L. Schmidt-Thieme, "Near real-time geolocation prediction in twitter streams via matrix factorization based regression," in *Proceedings of the 25th ACM international on conference on information and knowledge management*. ACM, 2016, pp. 1973–1976.

[19] 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 *IJCAI*, 2017, pp. 3602–3608.

[20] P. Zhao, H. Zhu, Y. Liu, J. Xu, Z. Li, F. Zhuang, V. S. Sheng, and X. Zhou, "Where to go next: A spatio-temporal gated network for next poi recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5877–5884.

[21] D. Yao, C. Zhang, J. Huang, and J. Bi, "Serm: A recurrent model for next location prediction in semantic trajectories," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 2411–2414.

[22] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[24] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *Proceedings of the International Conference on Learning Representations*, 2017.

[25] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[26] M. C. Mukkamala and M. Hein, "Variants of rmsprop and adagrad with logarithmic regret bounds," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2545–2553.

[27] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han, "Gmove: Group-level mobility modeling using geo-tagged social media," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1305–1314.