Recurrent Spatio-Temporal Point Process for Check-in Time Prediction

Guolei Yang
Facebook, Inc.
Menlo Park, California
yanggl@iastate.edu

Ying Cai
Iowa State University
Ames, Iowa
yingcai@iastate.edu

Chandan K. Reddy
Virginia Tech.
Arlington, Virginia
reddy@cs.vt.edu

ABSTRACT

We introduce a new problem, namely, *check-in time prediction* where the goal is to predict the time when a given user will check-in to a location of interest. We design a novel Recurrent Spatio-Temporal Point Process (RSTPP) model for check-in time prediction. RSTPP addresses two key challenges: 1) Data scarcity due to uneven distribution of check-ins among users/locations. 2) User trajectories contain valuable information that is ignored by standard temporal point process which only considers historical event times. RSTPP is designed to learn the latent dependencies of event times over both historical events and spatio-temporal information about locations a user visited before check-in to the location of interest. We evaluate RSTPP on several real-world datasets, and it significantly outperforms state-of-the-art event time predicting techniques. Our work derives a set of practical implications that can benefit a wide spectrum of applications.

CCS CONCEPTS

• **Information systems** → **Location based services**; **Data mining**; • **Computing methodologies** → *Machine learning*;

KEYWORDS

Check-in time prediction, recurrent spatio-temporal point process, deep learning, LSTM, location-based social networks.

ACM Reference Format:
Guolei Yang, Ying Cai, and Chandan K. Reddy. 2018. Recurrent Spatio-Temporal Point Process, for Check-in Time Prediction. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3269206.3272003

1 INTRODUCTION

Using GPS-enabled devices such as smartphones, people can conveniently share their locations in real-time. This trend of location-sharing has led to the prosperity of Location-Based Social Networks (LBSNs). Traditional social networks including Facebook and Google+ are endeavoring to add geo-spatial features to their services, e.g., allowing users to "geo-tag" posts and report their

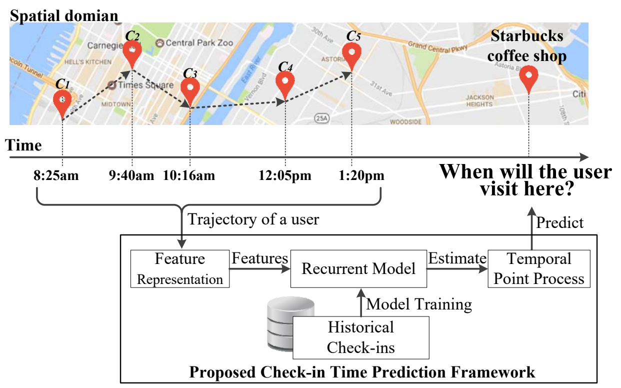

Figure 1: An illustration of check-in time prediction.

check-ins to Point-of-Interests (PoIs). Consequently, a huge amount of check-in data which records the moving trajectories of millions of individuals is being collected on a daily basis.

We propose to leverage massive amount of check-in data for the problem of **check-in time prediction**. Specifically, given a user and a location of interest (e.g., a PoI or a region), the goal is to predict the time when the user will check-in to the location, regardless of whether the user has visited the location before or not. Figure 1 illustrates the check-in time prediction problem with an example. A user trajectory consists of five successive check-ins $C = \{c_1, c_2, ..., c_5\}$. Using C as observation, the task is to predict when the user will visit a given shop based on the knowledge learned from historical check-ins to the shop.

As large-scale location data is becoming available from various applications and platforms, the problem of predicting movements of individuals has been gaining a lot of popularity. Existing works (e.g., [3, 9, 10, 16, 20, 26]) mainly aim to *predict the next location(s)* a user is going to visit given his/her current location. These techniques cannot be directly applied to check-in time prediction.

Our objectives of this work are two-fold — to complement existing research works towards the goal of thoroughly understanding human mobility patterns, and to derive practical implications for real-world applications. Check-in time prediction can serve as a building block for a wide spectrum of applications. For example, it can be used to identify the users who are most likely (or not likely) to visit a shop or an attraction within a designated time window. This information can be directly used in targeted marketing, tourism service, ride-sharing, etc. Moreover, the ability to precisely model a user's check-in time to any PoI enables new types of applications that go beyond the current scope of check-in prediction. For instance, given a collection of PoIs, our solution can order these PoIs by the time that a user will visit them, i.e., *predicting the user's itinerary*. Using this information, a recommendation system can recommend a series of PoIs/activities in certain orders that matches the user's itinerary, which is a new feature not offered by existing recommendation systems.

Since our goal is to predict event time, *Temporal Point Process* (TPP) can be used to tackle this problem. Standard TPP models assume that the time of an event occurrence conditionally depends on the time of its previous events. Therefore, TPP predicts a user's next check-in time to a location l based on his/her previous check-ins to the same location. However, there are two critical challenges that need to be addressed for TPP to be applied on this problem.

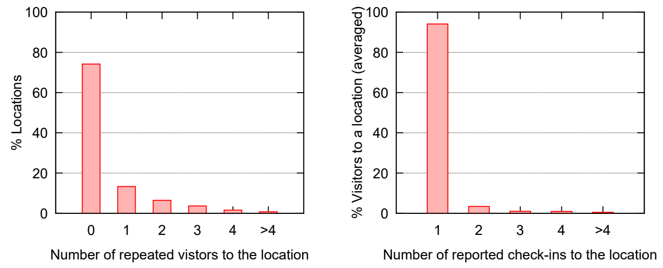

Figure 2: Check-in statistics of 4500 users and 965 locations in New York City randomly crawled from Foursquare.

(i) Check-in data scarcity: User check-ins are abundant but highly unevenly distributed among users and locations (Figure 2). For example, the Foursquare dataset [23] contains about 300 million check-ins but less than 10% of the PoIs have more than 10 check-ins [16]. As a result of the uneven distribution of check-ins, locations that have repeated visitors, i.e., users who visited the location more than once, are extremely rare. This is because check-ins are self-reported and hence are sparse and incomplete. If we model check-ins to a location as event sequences, a large proportion of the sequences will have only one event, making it very hard, if not impossible, to train a valid TPP model. Moreover, *for new visitors to a location, no historical event is available to make predictions.*

(ii) Drawback of standard temporal point process: Given a location of interest l, standard TPP models (e.g., Hawkes process) make predictions solely based on the user's historical check-in times to l. Locations a user visited before l, which may contain valuable information, are simply ignored. For example, if a user just had a meal at a restaurant, it is not likely that he/she will immediately visit another restaurant. Historical check-ins may reflect a user's preference for different locations, which also govern his/her future check-in behaviors. Such factors can hardly be captured by looking only at check-in times to the location of interest.

Our contributions: We address the above two challenges using a novel Recurrent Spatio-Temporal Point Process (**RSTPP**) model. We design a modified Long Short Term Memory (LSTM) network to learn the latent dependencies of next event time over diverse user and location information from both historical check-ins to l and other locations users visited before l. By exploring the long short-term memory mechanism, RSTPP is able to identify and remember most relevant check-ins while forgetting the influence of irrelevant check-ins. We summarize the contributions of this paper:

- Introduce the *check-in time prediction problem* which has practical value for a wide variety of applications. Given a user and a location, the goal is to predict the time when the user will visit the location.
- We propose a novel RSTPP model which combines long short-term memory network and temporal point process. Compared to standard TPP models, RSTPP has two unique features: 1) It can take advantage of relatively abundant precedent-location information on users' trajectories to improve accuracy. 2) Even if a user has not visited l before, our model is still able to make predictions using only his historical trajectory as observation.
- Our model is evaluated on real-world datasets. Our experimental results show that RSTPP outperforms state-of-the-art event time prediction techniques for various prediction tasks.

The rest of the paper is organized as follows: Section 2 formally defines the problem. Section 3 presents the proposed model. Experimental results are shown in Section 4. Section 5 summarizes related work. And Section 6 concludes the paper.

2 PRELIMINARIES

2.1 Problem Statement

We define the notions of check-in and trajectory used in this paper.

Definition 2.1 (Check-in). Let U denote a set of unique user identifiers, L denote a set of locations, and T denote the time domain. A check-in c is a triplet $(u, l, t) \in U \times L \times T$, which indicates the user u has visited l at time t.

Definition 2.2 (Trajectory). Let C be the collection of check-ins and $u \in U$ a user, then the set $C_u := \{u' = u | (u', l, t) \in C\}$ is the trajectory of u.

Here, a *location* can refer to a specific PoI, e.g., a hotel, or a user-defined spatial region. In LBSNs such as Foursquare, locations are usually associated with some descriptive information, such as its coordinates, category, user ratings, etc. We formally define our problem as follows:

Definition 2.3 (Check-in Time Prediction). Given a location of interest $l \in L$, a user of interest $u \in U$, and a sequence of historical check-ins $\{c_1, c_2, ..., c_k\} \subseteq C$ of u, predict the next time t when u will check-in to l.

2.2 Background

Temporal point process is a class of stochastic process that models the time of a sequence of events, denoted by $\{t_1, t_2, ..., t_n\}$ where $t_i \in \mathbb{R}^+$ is the time of the i^{th} event. In this paper, an event of interest is a user u check-in to a given location l. Note that the trajectory of u may contain many check-ins, but only those check-ins to l are considered to be events. Typical temporal point process assumes that the time of an event conditionally depends on the time of historical events. This dependency is described by the *conditional intensity function*, which has the following form:

$$\lambda(t)dt = P\{\text{an event occurs in } [t, t + dt)|H\} \tag{1}$$

Here, $\lambda(t)$ is the intensity function which can be considered as the instantaneous probability a new event will occur at time t. The probability is conditional on H, which is a set of historical events occurred before t. Let t_n denote the time of the last event in H, the probability that no event has occurred in $[t_n, t)$ is:

$$S(t|H) = exp\big(-\int_{t_n}^{t} \lambda(\tau)d\tau\big) \quad (2)$$

Hence, the conditional probability density that the next event will occur at time t given H is:

$$f(t|H) = \lambda(t)S(t|H) \quad (3)$$

Using the density function, we can compute the expected time of the next event as:

$$\hat{t}_{exp} = \int_{t_n}^{\infty} tf(t|H) \quad (4)$$

Equation (4) usually does not have an analytic solution [4]. Thus we obtain approximate numerical solutions to estimate the parameters of the intensity function. Specifically, we can maximize the joint log-likelihood of observing the historical events H:

$$l(H) = \sum_{i=1}^{n} \log(f(t_i|H)) \quad (5)$$

There is a rich family of TPP models, each with a unique form of intensity function. Instead of empirically constructing an intensity function for our problem, we design a long short term memory network to learn an intensity function using information from both user profile and his/her check-in data.

Long Short Term Memory [12] is a special type of recurrent neural network that is capable of learning both long-term and short-term dependencies between current event and historical events. The historical check-in data of a user may be accumulated from a relatively long time period (e.g., several weeks or even a year). Consequently, a new check-in event may reflect a user's habits and preferences that are only obvious when looking at long-term check-in behaviors. On the other hand, there might also be many historical check-ins that are irrelevent to a new check-in. As such, we want the model to be able to identify and remember relevant check-ins while forgeting the influence from irrelevant events. LSTM is found to be particularly suitable to model such complex dependencies in various applications [2, 19, 22].

Our goal is to allow the model to learn from the information of locations a user visited before an event of interest occurs. Such information may reflect the user's mobility patterns and are hence valuable to our problem. To this end, we modify a LSTM to take a user trajectory as input. A trajectory which may contain several events, only one event, or no event at all. We use diverse spatio-temporal information contained in the trajectory to make predictions for next event time.

3 RECURRENT SPATIO-TEMPORAL POINT PROCESS

Historical events are scarce in check-in data due to the lack of repeated check-ins to the same location by a user. This causes difficulties in both training and predicting stage of TPP models. We enrich the event sequences with *Precedent Check-ins*, which are check-ins a user reported before check-in to l, the location of interest. Figure 3 illustrates a normal sequence of events and the same sequence enriched with precedent check-ins.

Our intuition is that consecutive check-ins are correlated. If a user checks-in to a restaurant, this behaviour shall decrease the user's intensity to visit another restaurant at the moment. But check-in to a shop may not affect the intensity to visit a nearby

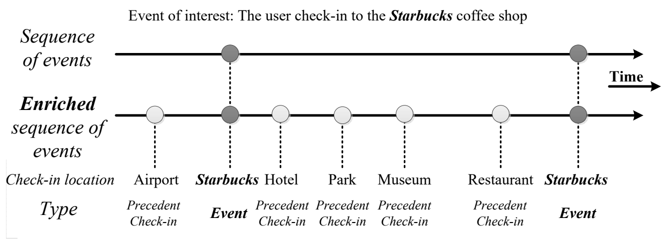

Figure 3: Example of an enriched event sequence.

shop of the same type. Similarly, moving towards or away from a location shall cause the intensity to visit the location to fluctuate accordingly. These examples are straightforward but the actual dependency could be much more complex. Our RSTPP model is designed to discover such dependencies.

3.1 Feature Representation

Each check-in in the sequence is represented by a feature vector which contains the information about the check-in in a more structured manner. The feature vector for a check-in c_i is denoted by $\boldsymbol{y_i}$, which consists of the following elements:

1. Localized event time $t_i \in \mathbb{R}$. Time of the earliest check-in to the location is set to 0 and the remaining times are set accordingly in the unit of hours or days.
2. Additional time-date information which indicates if the check-in occurred on a weekday or a weekend, and the time period of the day (e.g., morning, afternoon, or midnight). This is because humans demonstrate different check-in patterns at different time periods.
3. Euclidean distance between the check-in location to the location of interest l. The distance is computed using coordinates of the locations and normalized to the range of $[0, 1]$, where 1 indicates the distance is negligible, and 0 means that the distance is larger than a threshold denoted by θ_D.
4. Category of the check-in location, such as *Restaurant* or *Hotel*. We use a x-bit vector to represent the category where x is the total number of categories. The i^{th} bit is set to 1 if the location belongs to the i^{th} category, otherwise 0.
5. Number of users overlapping with the location of interest l. Let p denote the check-in location of c_i. We compute the proportion of users that have visited both p and l within a given time window. It reflects the inherent relevance of the two locations. The overlapping ratio is computed as follows:

$$OR = C_{l,p}^{v} \Big/ \sqrt{C_l^v C_p^v}, \quad (6)$$

where $C_{l,p}^{v}$ is the number of users who visited both l and p within a time window v. C_l^v and C_p^v are the number of visitors to l and p within the time window, respectively.
6. Previous check-ins by friends, which reflects the social aspect of check-in behaviors. The friend information can be collected from a user's social network. We calculate the minimal time interval from t_i to a check-in to the location of interest by a friend of the user. The value is normalized to $[0, 1]$ such that 1 indicates a friend just check-in to the location and 0 means the minimal time interval is larger than a threshold denoted by θ_T.

Each vector is also associated with a Boolean value indicating whether it is an event or a precedent check-in. Not all the above information are always available on LBSNs. e.g., some LBSNs do not provide location categories. In such cases, the unavailable features are ignored in generating the vector. The user feature vector p_u contains two values.

The first value is the user's preference to the location of interest l. We incorporate both explicit and implicit preferences. Explicit preference is the user's rating for l, which is usually not directly available for every location, but can be estimated using techniques such as matrix factorization [14]. As for implicit preference, we calculate the user's check-in frequency to l, i.e., the total number of check-ins to l on the user's trajectory, divided by the length (time between the first and the last check-in) of the trajectory.

The second value is the normalized distance between the location of interest and the user's home. The explicit home address of a user is usually unknown. Here we use the method introduced in [25] to estimate a user's home using his historical check-ins.

3.2 Model Architecture

Figure 4 illustrates the architecture of the proposed RSTPP model. The core of RSTPP is a set of hidden network layers and transformation functions (inside the dash-line box). The model takes an enriched event sequence through the input layer. Then, the hidden network layers update the internal status of the model by considering both current input and its previous status. As such, the model learns a compressed representation of all relevant historical check-ins which is stored in its internal status. Finally, the output layer uses the internal status to construct a conditional intensity function, which is used to make prediction of the next event time.

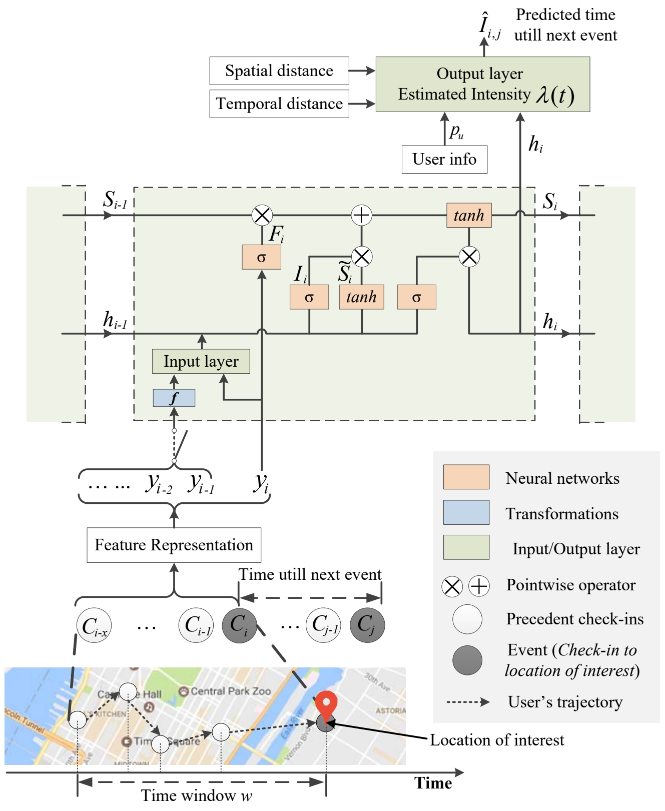

Figure 4: The proposed RSTPP model architecture.

Model Input: The input to the model is a sequence of events enriched with precedent check-ins, denoted by $\{c_1, c_2, ..., c_n\}$ where c_i $(1 \leq i \leq n-1)$ can be either an event or a precedent check-in. Using these check-ins, the goal of the model is to predict the next event time after c_n. When fed into the model, a check-in c_i is represented by a feature vector y_i. The model then iterates through the input sequence in n steps, and updates its internal status at each step. Comparing with standard LSTM, the input to our RSTPP at each step can be a single check-in, or a series of check-ins within a time window w. In the latter case, the accumulated information from these check-ins will be used as input by the model.

Internal Status Update: The internal status of the model at the i^{th} step can be represented as a vector h_i. In each step, the vector is updated according to the previous network status and current inputs. The information of check-ins is incorporated into the status vector during this updating process. Specifically, the status updating of our LSTM network is determined by the type of current input (i.e., if is a precedent check-in or an event).

The first step of status updating is to forget information of irrelevant historical check-ins. This is implemented by the forget gate layer F_i, which is computed as follows in our model:

$$F_i = \begin{cases} g(\boldsymbol{M_F}[h_{i-1}, y_i] + b_F), & \text{if } c_i \text{ is an event} \\ 1 \text{ (i.e., do not forget)} & \text{otherwise} \end{cases} \tag{7}$$

Here, $\boldsymbol{M_F}$ is the weight matrix of the forget gate layer, b_F is a constant, and $g(\cdot)$ is an activation function, for which we choose the sigmoid function $g(x) = exp(1/1 + e^{-x})$. $[\cdot, \cdot]$ denotes the concatenation of two vectors. Note that the forget step will be triggered only for events. As for precedent check-ins, all information should be kept because it will still be unclear which check-in(s) might influence the next event.

Then, the model decides which new information shall be remembered. This is implemented with two different hidden layers, the input gate layer I_i and the candidate cell status layer $\tilde{S}_i$. Depending on the type of input, they are updated as follows:

$$I_i = \begin{cases} g(\boldsymbol{M_I}[h_{i-1}, Y_i] + b_I), & \text{if } c_i \text{ is an event} \\ g(\boldsymbol{M'_I}[h_{i-1}, y_i] + b'_I), & \text{otherwise} \end{cases} \tag{8}$$

$$\tilde{S}_i = \begin{cases} tanh(\boldsymbol{M_S}[h_{i-1}, Y_i] + b_S), & \text{if } c_i \text{is an event} \\ tanh(\boldsymbol{M'_S}[h_{i-1}, y_i] + b'_S), & \text{otherwise} \end{cases} \tag{9}$$

In the above equations, $\boldsymbol{M_I}, \boldsymbol{M'_I}, \boldsymbol{M_S}, \boldsymbol{M'_S}$ are weight matrices and b_I, b'_I, b_S, b'_S are constant parameters. We use *tanh* as activation function since here we want the output value to be in the range of $[-1, 1]$. Finally, Y_i represents the *accumulated check-in information* from a series of successive check-ins occurred in a certain time window w from check-in time of y_i, which is defined as follows:

$$Y_i = W'_I y_i + \textstyle\sum_{(0<(t_i - t_j)<w)} f(t_i - t_j) W_I * y_j \tag{10}$$

where the condition $(t_i - t_j) < w$ defines a time window that includes all the precedent check-ins and events within w from the time of c_i. W_I and W_I' are the input transformation vectors, and $f(\Delta t)$ is the *time-sensitive impact-tuning function*. Specifically, it is a monotonically non-increasing function with an output range of $[0, 1]$ and $f(0) = 1$. $f(\Delta t)$ is meant to explicitly decrease the impact of "old" check-ins while having little influence on most recent check-ins. This is because temporally close check-ins are more likely to be relevant to each other.

The next step is to update the cell status, denoted by S_i. It is updated by "forget" the information marked by the forget gate layer while "remember" new information provided by the input gate and candidate status layer:

$$S_i = F_i S_{i-1} + I_i \tilde{S}_i \tag{11}$$

After status updating, the cell status now records the information that should be passed to the next step. And finally, we can update the vector representation of the internal network status:

$$h_i = \begin{cases} g(\boldsymbol{M_H}[h_{i-1}, Y_i] + b_H) tanh(S_i), & \text{if } c_i \text{ is an event} \\ g(\boldsymbol{M'_H}[h_{i-1}, y_i] + b'_H) tanh(S_i), & \text{otherwise} \end{cases} \tag{12}$$

Note that both S_i and h_i are passed to the next step of the model through the recurrent edges.

Model Output: Intuitively, the status vector h_i can be seen as a latent representation of knowledge learned from all previous check-ins by the model. We can now compute the conditional density function of next event time using h_i as follows:

$$\lambda(t) = exp\Big(W^T h_i + \beta_t (t - t_i) + \beta_d D + B^T p_u + b\Big) \tag{13}$$

The computation involves several factors: $W^T h_i$ computes the influence of previous check-ins, $\beta_t(t - t_i)$ and $\beta_d D$ represent current temporal and spatial impact, respectively, where D is the Euclidean distance between current location and the location of interest. p_u is the aforementioned user vector. Finally, b is a constant base intensity for the location. Note that the exponential function guarantees the intensity is positive. The intensity function is then used to predict the time interval between current check-in c_i and the next event c_j, defined as: $\hat{I}_{i,j} = \hat{t}_j - t_i$. Here $\hat{t}_j$ is estimated using Equation (4).

Note that the RSTPP model may generate multiple predictions for the time of the same event. This is because if an event has precedent check-ins, each of the check-ins will generate an output but they have the same next event. In this case, we will use the output of the most recent precedent check-in as the final prediction for the event time. There are two reasons behind this: 1) temporally close check-ins are usually more relevant to each other. 2) The output of the most recent precedent check-in is supposed to incorporate information of all the precedent check-ins and historical events, thus shall contain more complete information. If an event has no direct precedent check-ins, i.e., it follows another event with no other check-in in between, then the event will have a single predicted time. This design guarantees that a valid prediction can be made even for users who have never visited the location of interest. In that case, all his/her check-ins will be treated as precedent check-ins, since no event appears in the sequence.

Model Training: For a given location of interest, the training dataset is a collection of historical user trajectories. Each trajectory contains at least one check-in to the location as the ground truth. We use the Back-Propagation Through Time algorithm [18] to train the proposed RSTPP model. The loss function we used is the negative log-likelihood of observing the training instances (given in Equation (5)). During each iteration of the training process, output of the model is fed into the loss function and its parameters are updated accordingly in the back propagation stage using stochastic gradient descent until it achieves convergence. Due to space constraints, we will not elaborate on these classical algorithms.

4 EXPERIMENTS

In this section, we perform a rigorous set of experiments using different datasets and compare the performance of the proposed RSTPP model with several state-of-the-art methods.

4.1 Dataset Description

We use several check-in datasets collected by various location-based social networks and web services.

Foursquare[1] **:** This dataset was collected from Foursquare [23]. We use the 227,428 check-ins in NYC and 573,703 in Tokyo. Each check-in contains user ID, location ID, coordinates, and time. Using the user ID or location ID, we retrieve the user's friend list and location category on Foursquare.

Gowalla[2] **:** This most widely used data set was collected by the authors of [3] from the LBSN Gowalla. This dataset contains 6,442,890 check-ins reported by 196,591 users globally. Location category is not available in this dataset. And since Gowalla is closed now, we cannot collect such information online. Thus we will ignore location category for our experiments on the dataset.

Brightkite[3] **:** The dataset is collected from another (now closed) LBSN but smaller in scale compared to the above two data sources. The dataset is also collected by the authors of [3]. It contains 4,491,143 check-ins from 58,228 users. Similar to the Gowalla dataset, location category is missing in this dataset.

Yelp[4] **:** Besides check-ins, this dataset provides information such as user-location ratings, user information, and location information. However, similar to Foursquare, the social network of users is not included in the dataset and it is not available on Yelp.com. We do not use social information for our experiments on this dataset.

4.2 Comparison Methods

Baseline models: We compare the proposed model to a set of generic time-to-event modeling techniques, including the state-of-the-art temporal point process-based techniques.

- **Most popular time (Popular)**. This method returns the most popular check-in time in a day for a given location as the predicted check-in time. To find the most popular check-in time, we fit a mixture of two normal distributions over 24 hours of a day for the location using all its check-in times. This is because

[1] https://sites.google.com/site/yangdingqi/home/foursquare-dataset
[2] https://snap.stanford.edu/data/loc-gowalla.html
[3] https://snap.stanford.edu/data/loc-brightkite.html
[4] https://www.yelp.com/dataset_challenge

Table 1: Performance of predicting check-in time to a PoI , showing RMSE and (standard deviation), unit: day.

Model	Repeated visitors only				New visitors only				All check-ins a model can use			
	Foursquare	*Gowalla*	*Brightkite*	*Yelp*	*Foursquare*	*Gowalla*	*Brightkite*	*Yelp*	*Foursquare*	*Gowalla*	*Brightkite*	*Yelp*
Popular	6.3751	6.5258	8.7713	7.3505	7.5762	6.9939	9.0177	8.3011	6.6520	6.7785	8.7888	7.5633
	(1.9014)	(1.8835)	(2.2176)	(2.0114)	(1.6359)	(1.8510)	(2.0228)	(1.8144)	(1.7135)	(1.6202)	(1.9947)	(1.9761)
Linear	2.1018	1.6662	3.3810	2.5159	2.2218	2.0100	2.9923	2.6006	1.9970	1.8541	2.4510	2.5013
	(0.5931)	(0.4858)	(0.5219)	(0.6220)	(0.5217)	(0.4344)	(0.5561)	(0.5482)	(0.5120)	(0.4036)	(0.5333)	(0.5585)
Average	4.1157	3.0893	5.2489	4.9782					4.4336	3.7117	5.2210	4.6527
	(1.3816)	(1.3034)	(1.5948)	(1.4229)					(1.4481)	(1.3920)	(1.5537)	(1.5871)
Hawkes	1.8703	1.4839	1.5051	1.6592		*Cannot Support*			2.2139	2.1788	2.2938	2.1929
	(0.4712)	(0.4359)	(0.4007)	(0.4251)					(0.4849)	(0.4937)	(0.4150)	(0.4672)
SC	1.5834	1.4779	1.4025	1.6350					2.0345	2.0978	2.1960	2.2935
	(0.3226)	(0.3666)	(0.3583)	(0.3711)					(0.4014)	(0.3715)	(0.3903)	(0.3677)
ACD	1.7793	1.4745	1.3345	1.7001					2.2400	2.3333	2.3519	2.2744
	(0.3920)	(0.3812)	(0.3710)	(0.3449)					(0.4019)	(0.3610)	(0.3229)	(0.3905)
RMTPP	1.4947	1.4309	1.4032	1.6312					2.1353	2.0994	2.0375	2.2017
	(0.3621)	(0.3400)	(0.3511)	(0.3379)					(0.3977)	(0.3430)	(0.3317)	(0.3708)
RSTPP	**1.1202**	**1.1008**	**1.1328**	**1.3010**	**1.1403**	**1.2011**	**1.3329**	**1.2915**	**1.1019**	**1.1575**	**1.2452**	**1.2793**
	(0.3151)	**(0.3411)**	**(0.3559)**	**(0.3317)**	**(0.3560)**	**(0.3305)**	**(0.3295)**	**(0.3403)**	**(0.3527)**	**(0.3419)**	**(0.3231)**	**(0.3445)**

the most popular visit time of PoIs, such as restaurants, usually demonstrate a clear two-peak pattern.

- **Linear regression (Linear)**. We use standard linear regression to fit a model where the input is the feature vector generated for the current check-in (the same as the one used in RSTPP), and the predicted variable is the time till the next event.
- **Average inter-event time (Average)**. This method returns the sum of previous event time and the average inter-event times (time between two successive check-ins to the location of interest) of all visitors to the location as prediction.
- **Hawkes process (Hawkes)** [11]. It has the following intensity function: $\lambda(t|H) = \gamma_0 + \alpha \sum_{t_j<t} \gamma(t, t_j)$, where γ_0 is the base intensity and $\gamma(t, t_j)$ a kernel function selected by the user. In our experiments, we use the exponential kernel.
- **Self-Correcting process (SC)** [13]. The intensity function is: $\lambda(t|H) = exp\big(\mu t - \sum_{t_i<t} \alpha\big)$ with two parameters μ and α to be learned.
- **Second-order Autoregressive Conditional Duration (ACD)** [6]. It has the following intensity function: $\lambda(t|H) = \gamma_0 + \sum_{j=0}^{m} \alpha_j d_{i-j}$, where d_{i-j} is the duration between the i^{th} and j^{th} events.
- **Recurrent Marked Temporal Point Process (RMTPP)** [4]: The state-of-the-art TPP model. Note that the event marker is not a necessary component in our problem. Thus we will feed a dummy marker to RMTPP in the training process and only use the predicted next event time. The size of hidden layers of RMTPP is set to be the same value as the ones used in our model.

We use the implementations provided by the authors of [4] for the TPP-based models. The other simple baselines are implemented using standard *R* library. Since the goal is to predict the next check-in time *to a specific location*, the TPP-based models can only be trained on sequences that contain repeated check-ins to the same location. Otherwise, they cannot learn the dependency between event times since there may only be one event per visitor.

RSTPP Model settings: [5] We select the hyper-parameters as follows. The check-in time window w is set to 12 hours, because check-ins that are more than 12 hours apart are usually less relevant to each other. As for the time-sensitive impact-tuning function, we use $f(\Delta t) = b/\log(e + \Delta t)$ where b is a parameter to be learned. Each check-in has a label that indicates its time period of the day: Morning (5:00am - 10:00am), Noon (10:00am - 2:00pm), Afternoon (2:00pm - 6:00pm), Night (6:00pm - 11:00pm), LateNight (11:00pm - 5:00am next day). We use these time periods because they usually reflect different check-in patterns. The time window v used to compute visitor overlapping ratio of two locations is set to $v = \{\infty, 24hr, 8hr, 4hr, 1hr\}$, as such, five overlapping ratios will be generated and added to each feature vector using different time windows. The distance threshold θ_D used to generate feature vectors is set to 10km and the time threshold θ_T is $12hr$. The size of hidden layer is 512. The learning rate is set to 0.01 with a momentum of 0.9.

[5]Upon acceptance of this paper, the implementation of the proposed model will be hosted on Github repository for public use.

4.3 Results and Analysis

For a comprehensive performance evaluation, we design four prediction tasks. Each task targets certain types of real-world application scenarios.

Task I: Predict check-in time to a PoI:

First, we select a subset of locations having at least one repeated visitor, so that the baseline models can be trained. The four datasets contain 8519, 21273, 4495, and 3970 check-in sequences after the selection, respectively. On average, each check-in sequence contains 2.37 events and has a duration of 14 days. The proposed RSTPP model is trained and tested on the same sequences but enriched with precedent check-ins. For all the models, 80% of randomly selected user trajectories are used for training while the rest are used for testing. If a user has visited a location for k times, the first $k-1$ check-ins are used as observations to predict his k^{th} check-in time. We report both mean and the standard deviation of the Root-Mean-Square-Error (RMSE) between the predicted and actual check-in times (in days).

Second, we train RSTPP model using only locations without repeated visitors. The baseline models, except for the popular and linear model, cannot support such scenarios and hence cannot be compared against. During testing time, we make predictions only

Table 2: Performance of predicting check-in time to an area, showing RMSE and (standard deviation), unit: day.

Model	Repeated visitors only				New visitors only				All check-ins a model can use			
	Foursquare	*Gowalla*	*Brightkite*	*Yelp*	*Foursquare*	*Gowalla*	*Brightkite*	*Yelp*	*Foursquare*	*Gowalla*	*Brightkite*	*Yelp*
Popular	6.2890 (1.8109)	5.9712 (1.7993)	8.2970 (2.1114)	6.9935 (2.0531)	6.8512 (1.5739)	6.2001 (1.6655)	8.5109 (1.9351)	7.2324 (1.8080)	6.4883 (1.6811)	6.0974 (1.5405)	8.2996 (2.0150)	7.1953 (1.9935)
Linear	2.0410 (0.6227)	1.6075 (0.4153)	2.9750 (0.5771)	2.1417 (0.6015)	2.1958 (0.4610)	1.7795 (0.4088)	2.8895 (0.5183)	2.4535 (0.5479)	2.2518 (0.5051)	1.8694 (0.4100)	2.4510 (0.5212)	2.3944 (0.5510)
Average	3.7520 (1.3143)	2.8950 (1.1175)	4.9777 (1.6156)	4.5212 (1.3319)	*Cannot Support*				4.0033 (1.2508)	3.3105 (1.2995)	4.8528 (1.5460)	4.6048 (1.4561)
Hawkes	1.5328 (0.4320)	1.5002 (0.4400)	1.4339 (0.3620)	1.5209 (0.4188)					1.5729 (0.4179)	1.5493 (0.4228)	1.4665 (0.3750)	1.5139 (0.4093)
SC	1.3550 (0.3375)	1.2298 (0.3145)	1.3570 (0.3277)	1.5120 (0.3169)					1.4866 (0.3351)	1.2585 (0.2982)	1.4480 (0.3328)	1.4765 (0.3210)
ACD	1.4079 (0.3885)	1.5358 (0.3531)	1.4295 (0.3203)	1.5333 (0.3071)					1.5625 (0.3623)	1.5917 (0.3255)	1.5680 (0.3017)	1.6232 (0.3456)
RMTPP	1.3015 (0.3431)	1.2078 (0.3057)	1.4700 (0.3219)	1.4955 (0.3273)					1.4812 (0.3339)	1.3995 (0.3333)	1.5030 (0.3204)	1.5184 (0.3177)
RSTPP	**1.0943 (0.3100)**	**0.9766 (0.3107)**	**1.1010 (0.3157)**	**1.1507 (0.3115)**	**1.1430 (0.3298)**	**1.1933 (0.3207)**	**1.2052 (0.3109)**	**1.1533 (0.3150)**	**1.0924 (0.3077)**	**1.0529 (0.3300)**	**1.2101 (0.3123)**	**1.1336 (0.3005)**

for new visitors, and hence only precedent check-ins can be used as observations. Similarly, 80% of the trajectories are used for training and the remaining are used for testing.

Finally, we let each model use as much data as it possibly can. RSTPP, popular, and linear model are trained using all check-ins. The other models are trained using locations with repeated visitors. We make predictions for both repeated and new visitors. For the TPP-based baseline models, a prediction is made using only the base intensity value for new visitors.

Table 3: Prediction error w.r.t. number of PoIs per area.

Model	#PoI per area			
	5	10	15	20
Popular	6.233	6.180	6.025	5.418
Linear	2.425	2.204	2.053	1.970
Average	5.121	5.060	4.579	4.334
Hawkes	1.562	1.469	1.463	1.447
SC	1.534	1.438	1.427	1.425
ACD	1.507	1.554	1.537	1.492
RMTPP	1.473	1.455	1.448	1.432
RSTPP	**1.093**	**1.067**	**0.949**	**0.934**

The results for this task are shown in Table 1. The proposed RSTPP outperforms all the baseline models in terms of RMSE, which shows that incorporating location information of precedent check-ins can significantly improve the prediction performance. RSTPP shows better performance on the Foursquare dataset compared to other datasets. It can be explained by the fact that the Foursquare dataset provides most complete information about locations and users (i.e., location categories and user social connections), which is not available in other LBSNs. We hypothesize that the proposed model can be further improved if more complete user trajectories were available to the model.

RSTPP significantly outperforms the baselines in predicting check-in time for all types of visitors. This is due to the dominating majority (>90%) of check-ins in the datasets were made by new visitors. When making predictions for such new visitors, the TPP-based models can only use the constant base intensity, resulting in an obvious drop on the average performance. In contrast, RSTPP can use precedent check-ins of new visitors as observations.

Task II: Predict check-in time to an area:

A location can also be a pre-defined spatial area. In this task, we generate an area as follows: First, we randomly select a PoI and find its k-nearest neighbors. The k value is randomly selected in [5, 20]. Then, these PoIs are treated as an area, and check-ins to any of these PoIs are counted as check-ins to the area. We then generate 1,000 such areas in this manner for each dataset. The TPP models benefit from the generated check-ins since they boost the number of repeated visitors to a location by 5.5 times on average.

Table 4: Prediction error w.r.t. number of locations to order.

Model	#Locations to order			
	2	3	4	5
Popular	0.508	0.511	0.511	0.513
Linear	0.376	0.391	0.403	0.403
Average	0.510	0.541	0.512	0.509
Hawkes	0.257	0.262	0.300	0.309
SC	0.254	0.271	0.295	0.310
ACD	0.268	0.274	0.304	0.310
RMTPP	0.255	0.268	0.292	0.309
RSTPP	**0.182**	**0.190**	**0.232**	**0.239**

Similarly, we train three sets of models using repeated visitors only, using new visitors only, and using all check-ins a model can use. The comparison results are shown in Table 2. Note that even with the event sequences of a much larger number of repeated visitors as training instances, the baseline methods are outperformed by the proposed RSTPP model for all the test cases.

We also analyze the performance in areas with different size (the number of PoIs in the area)(Table 3). In general, all models have better performance in larger area compared to smaller ones. The reason is that larger areas are likely to have more repeated visitors and more events per visitor, which benefits the training process of all the models. The proposed model steadily outperforms all the baseline models in different areas.

Task III: Predict the check-in order to multiple locations: Finally, we evaluate the ability of our model to predict a user's check-in order to several given locations. That is, given k different locations, sort them by the predicted check-in time of the user. For this task, we use the Kendall's Tau as performance metric, which is commonly used to evaluate ranking results.

Kendall's Tau (τ) is the number of *inversions* in the predicted ordering of locations compared to the actual order of check-ins. A pair of locations is called an "inversion" if their actual order of a check-in pair is inverted in the prediction result. As such, $\tau = 0$ means the ordering is completely correct. Note that the Kendall's Tau for ordering k locations can be in $[0, 2^k]$. Therefore, we use the *normalized* Kendall's Tau (in range $[0, 1]$, the lower the better) to make it suitable for fair comparisons.

The models are trained with all the check-ins they can use as described in Task II. For test instances, we select only users who reported at least 10 check-ins. If a user has n check-ins, we use the first $n-k$ check-ins as observation, to predict the check-in times for the last k locations and sort them accordingly. The results (averaged over all the test sequences for each model) are shown in Table 4 with respect to $k = \{2, 3, 4, 5\}$. Predicting the check-in order of a large number of locations appears to be more challenging due to the much larger prediction space. But our RSTPP model shows the best overall performance with various values of k.

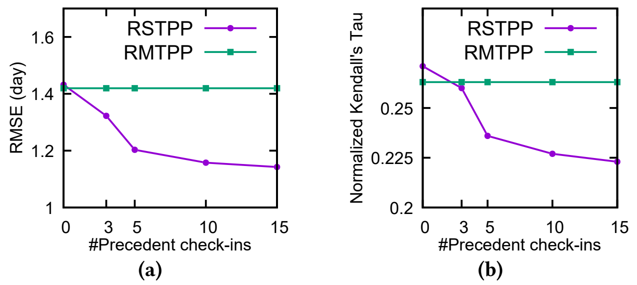

Figure 5: Prediction error w.r.t. the number of precedent check-ins used.

Task IV: Using different number of precedent check-ins: This task aims to directly evaluate the advantage of using precedent check-ins in our model. Specifically, we select a set of trajectories from the four datasets, such that each trajectory contains repeated visits to some randomly selected locations of interest. Then, we adjust the number of precedent check-ins that can be used on each trajectory from 0 to 15, where 0 means the trajectory contains only events (i.e., all precedent check-ins are removed) while 15 corresponds to at most 15 precedent check-ins can be used by a model. At each test point, we train a proposed model and compare its performance with RMTPP, which is the best performer among the baselines. The two models are used to predict the check-in time to a PoI and the check-in order to 3 locations. The results are shown in Figure 5 (a) and Figure 5 (b). Note that the performance of RMTPP is shown as a straight line since it does not use precedent check-ins. In contrast, RSTPP shows significant improvements on prediction error as the number of used precedent check-ins increases.

4.4 Practical Implications

- Data scarcity is a prevalent problem in check-in data mining [10, 16, 26]. We show that this problem is tractable by exploring possible ways to "enrich" the data. For check-in time prediction specifically, it is crucial to consider the entire trajectory of a user. A large number of users' check-ins demonstrate patterns in both spatial and temporal dimension. Hence, consecutive check-ins within a short time period are usually correlated.
- The aforementioned new applications are made feasible with the proposed model. PoI owners can use our model to find future visitors, as well as their check-in time, to their PoIs within the next 1 to 2 days. Our model also achieves good accuracy in the itinerary prediction, which can be used to recommend locations in an order that best matches a user's itinerary.
- Dense and complete check-in data is the key to check-in time modeling. Such data are generated by highly active users. Models trained on such data constantly outperform the ones trained using sparse trajectories, even when the total number of trajectories/users used are the same. It is worth studying how to better motivate LBSN users to report their complete trajectories.
- In check-in time modeling, information derived from a user's profile (e.g., social connections) are of less importance compared to their locations. It appears that a user's historical check-ins can tell much more about the user's preference and behaviour patterns than what he/she is willing to share in their profile.

5 RELATED WORK

5.1 Human Movement Prediction

The first large-scale study of human mobility patterns using spatio-temporal data collected by LBSNs on the Internet was done by Cho *et al.* [3]. By analyzing the data, the authors identified a set of factors that affect a user's moving patterns. Based on these factors, they designed a set of predictive models which can reproduce a user's movement. Several techniques have since been proposed to predict the next location a user is most likely going to, given his current trajectory. Noulas *et al.* [16] proposed to extract spatio-temporal features from users check-in data and then train a model on these features for the next location prediction. Unlike plain coordinates, check-in data contains the *exact place* (e.g., a store, a coffee shop) a user visited, which allows a model to learn from features such as location type. A few works also explore the "temporal" aspect of next location prediction. The *NextPlace* framework proposed in [20] can predict the next check-in location, the arrival time, and duration of the stay. The work in [17] proposed a novel model to predict a user's preference of a sequence of locations for the purpose of tour recommendation. In a very recent work [9], the authors proposed a generic predictive model termed '*TribeFlow*' which not only mines and predicts user trajectories but can also be used for next product recommendation. Recently, a Spatial Temporal Recurrent Neural Network (ST-RNN) was introduced in [15] for next location prediction. In this previous work [24], we introduced a Survival Analysis-based check-in time prediction framework which employs recurrent neural network to learn the intensity function of the occurrence of check-in events. In contrast, the proposed RSTPP in this work explores temporal point process for this application which has the ability to handle new visitors.

5.2 Temporal Point Process

Temporal Point Process (TPP) models a sequence of events. Typically, TPP models assume the time of an event conditionally depends on the time of previous events. Based on the specific form of dependency, TPP has several variants. For instance, Hawkes process [11] assumes each occurrence of a historical event will increase the probability of the occurrence of a new event. In contrast, Self-correcting process [13] assumes that the probability of a new event grows over time, but every time a new event occurs, it will drop by a certain amount. Autoregressive Conditional Duration process [6] models the dependency between inter-event durations.

TPP is adopted in applications such as disastrous event analysis [11], criminal networks modeling [21], and financial prediction [1]. TPP also demonstrated remarkable performance in predicting recurrent user activities [7, 8]. In [5], Du *et al.* introduced Low Rank Hawkes process to predict the next returning time of a user to a service. This model can be generalized by explicitly incorporating spatial information into its intensity function. However, none of these works discussed any mechanisms for incorporating spatio-temporal features, not to mention handling check-in data scarcity.

A closely related technique is Recurrent Marked Temporal Point Process (RMTPP) [4]. RMTPP applies recurrent neural network to *Marked* TPP to predict the time of next event occurrence and its marker. RMTPP can be applied to predict check-in time by treating it as a standard TPP problem, hence it also suffers from the aforementioned drawbacks. This is because RMTPP cannot take advantage of successive check-ins on user trajectories before check-in to a location of interest, which is a unique feature offered by our RSTPP model. As such, our model is able to make predictions for users who have not even visited the given location before. In contrast, RMTPP and existing TPP models can only use base intensity to make prediction for all the users who have no historical events.

6 CONCLUSION

Human movement prediction is a long standing research topic with enormous application potential. Check-in time prediction is a relatively new problem in this field which was not fully investigated. In this paper, we proposed Recurrent Spatio-Temporal Point Process (RSTPP), a flexible and adaptive model designed for check-in time prediction. RSTPP is a novel combination of Long Short-Term Memory (LSTM) network and temporal point process. The LSTM component is used to learn a latent representation of a sequence of check-ins. This representation is then used to construct the intensity function of a temporal point process. Finally, the model uses this intensity function to predict the user's next check-in time to a location of interest. The unique structure of RSTPP allows it to take advantage of precedent check-ins on a user's trajectory, which alleviates the critical data scarcity problem. As a result, RSTPP is able to outperform state-of-the-art event-time prediction techniques that use standard point process models.

Acknowledgements

This work was supported in part by the National Science Foundation grants IIS-1619028, IIS-1707498 and IIS-1838730.

REFERENCES

[1] Emmanuel Bacry, Adrian Iuga, Matthieu Lasnier, and Charles-Albert Lehalle. 2015. Market impacts and the life cycle of investors orders. *Market Microstructure and Liquidity* 1, 02 (2015), 1550009.

[2] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733* (2016).

[3] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *17th ACM SIGKDD conference on knowledge discovery and data mining*. ACM, 1082–1090.

[4] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1555–1564.

[5] Nan Du, Yichen Wang, Niao He, Jimeng Sun, and Le Song. 2015. Time-sensitive recommendation from recurrent user activities. In *Advances in Neural Information Processing Systems*. 3492–3500.

[6] Robert F Engle and Jeffrey R Russell. 1998. Autoregressive conditional duration: a new model for irregularly spaced transaction data. *Econometrica* (1998), 1127–1162.

[7] Mehrdad Farajtabar, Nan Du, Manuel Gomez Rodriguez, Isabel Valera, Hongyuan Zha, and Le Song. 2014. Shaping social activity by incentivizing users. In *Advances in neural information processing systems*. 2474–2482.

[8] Alceu Ferraz Costa, Yuto Yamaguchi, Agma Juci Machado Traina, Caetano Traina Jr, and Christos Faloutsos. 2015. Rsc: Mining and modeling temporal activity in social media. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 269–278.

[9] Flavio Figueiredo, Bruno Ribeiro, Jussara M Almeida, and Christos Faloutsos. 2016. TribeFlow: mining & predicting user trajectories. In *Proceedings of the 25th International Conference on World Wide Web (WWW'16)*. 695–706.

[10] Huiji Gao, Jiliang Tang, and Huan Liu. 2012. Mobile location prediction in spatio-temporal context. In *Nokia mobile data challenge workshop*, Vol. 41. 44.

[11] Alan G Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* 58, 1 (1971), 83–90.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Valerie Isham and Mark Westcott. 1979. A self-correcting point process. *Stochastic Processes and Their Applications* 8, 3 (1979), 335–347.

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[15] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts.. In *AAAI Conference on Artificial Intelligence*. 194–200.

[16] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. 2012. Mining user mobility features for next place prediction in location-based services. In *Data mining, IEEE 12th international conference on*. IEEE, 1038–1043.

[17] Vineeth Rakesh, Niranjan Jadhav, Alexander Kotov, and Chandan K Reddy. 2017. Probabilistic Social Sequential Model for Tour Recommendation. In *Tenth ACM International Conference on Web Search and Data Mining*. ACM, 631–640.

[18] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, and others. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.

[19] Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

[20] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T Campbell. 2011. NextPlace: a spatio-temporal prediction framework for pervasive systems. In *International Conference on Pervasive Computing*. Springer, 152–169.

[21] MB Short, GO Mohler, P Jeffrey Brantingham, and GE Tita. 2014. Gang Rivalry Dynamics via Coupled Point Process Networks. *Discrete & Continuous Dynamical Systems-Series B* 19, 5 (2014).

[22] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

[23] Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu. 2015. Nation-Telescope: Monitoring and visualizing large-scale collective behavior in LBSNs. *Journal of Network and Computer Applications* 55 (2015), 170–180.

[24] Guolei Yang, Ying Cai, and Chandan K Reddy. 2018. Spatio-Temporal Check-in Time Prediction with Recurrent Neural Network based Survival Analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2976–2983.

[25] Guolei Yang and Andreas Züfle. 2016. Spatio-Temporal Site Recommendation. In *Data Mining Workshops, IEEE 16th International Conference on*. IEEE, 1173–1178.

[26] Jihang Ye, Zhe Zhu, and Hong Cheng. 2013. What's your next move: User activity prediction in location-based social networks. In *SIAM International Conference on Data Mining*. SIAM, 171–179.