Context-aware Deep Model for Joint Mobility and Time Prediction

Yile Chen
Nanyang Technological University
Singapore
yile001@e.ntu.edu.sg

Cheng Long
Nanyang Technological University
Singapore
c.long@ntu.edu.sg

Gao Cong
Nanyang Technological University
Singapore
gaocong@ntu.edu.sg

Chenliang Li
Wuhan University
China
cllee@whu.edu.cn

ABSTRACT

Mobility prediction, which is to predict where a user will arrive based on the user’s historical mobility records, has attracted much attention. We argue that it is more useful to know not only where but also when a user will arrive next in many scenarios such as targeted advertising and taxi service. In this paper, we propose a novel context-aware deep model called DeepJMT for jointly performing mobility prediction (to know where) and time prediction (to know when). The DeepJMT model consists of (1) a hierarchical recurrent neural network (RNN) based sequential dependency encoder, which is more capable of capturing a user’s mobility regularities and temporal patterns compared to vanilla RNN based models; (2) a spatial context extractor and a periodicity context extractor to extract location semantics and the user’s periodicity, respectively; and (3) a co-attention based social & temporal context extractor which could extract the mobility and temporal evidence from social relationships. Experiments conducted on three real-world datasets show that DeepJMT outperforms the state-of-the-art mobility prediction and time prediction methods.

CCS CONCEPTS

• **Information systems → Data mining**; **Location based services**; • **Human-centered computing →** *Ubiquitous and mobile computing design and evaluation methods.*

KEYWORDS

mobility prediction; user modeling; location based services; neural networks

ACM Reference Format:
Yile Chen, Cheng Long, Gao Cong, and Chenliang Li. 2020. Context-aware Deep Model for Joint Mobility and Time Prediction. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM’20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371837

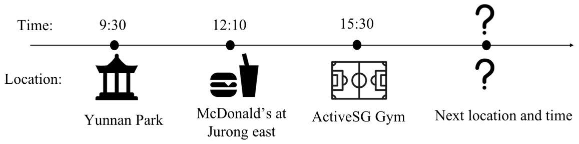


Figure 1: An example of joint mobility and time prediction

1 INTRODUCTION

With the prevalence of location-based services, a huge amount of mobile users’ footprint data is being generated. For example, location-based social networks (LBSNs) such as Foursquare have millions of users who check in at points-of-interest (POIs) in their daily lives. The generated footprint data provides a great opportunity of understanding human mobility behaviors. For example, researchers have proposed to leverage a user’s historical records to predict the user’s next location, i.e., where a user will arrive next.

There are many application scenarios where it is more desirable to know not only *where* but also *when* users will arrive next. The first example is targeted advertising, where an advertiser could push POI recommendations (based on predicted location) to mobile users at appropriate time (based on predicted time). The second example is taxi service, where a taxi company can route its taxis effectively to reduce taxi requesters’ waiting time if their travel plans of a destination and a targeted arrival time could be predicted. The third example is traffic congestion control, where a transportation authority could plan earlier if some potential crowd congestion is predicted based on where and when people would gather. All these applications require more than what the conventional mobility prediction task could support.

Motivated by these applications, we propose to predict for a user where and when he/she will arrive next, i.e., we jointly predict users’ mobility and time. One example is shown in Figure 1 for illustration. While the new task looks promising, there are three technical challenges. The first one is how to encode the mobility records of a user. A commonly used strategy is to use recurrent neural network (RNN) models to learn the latent representation of mobility records. However, this method suffers from its degraded performance when

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WSDM ’20, February 3–7, 2020, Houston, TX, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371837

modeling long sequences and cannot distinguish between users' mobility transitions within a short period (e.g., within a few hours) and those spanning a longer period (i.e., over several days). The second challenge is how to incorporate various contexts of mobility records. For example, existing methods have been proposed to capture some spatial context but they simply use geographical distances [9, 13] and are far from being capable of capturing complex interactions between different locations. Another type of context is that of periodicity since human mobility often manifests periodic patterns, such as daily, weekly and monthly routines [7] and it cannot be well captured by existing matrix factorization based models such as LRT [10]. The third challenge is that some users only have a limited number of mobility records, which makes it hard to learn their mobility regularities and temporal patterns.

To address these challenges, we propose a context-aware deep model named DeepJMT for joint mobility and time prediction. Specifically, for the first challenge, we propose to apply a hierarchical RNN based *sequential dependency encoder* which is more capable of dealing with long sequences of mobility records and also preserving the trajectory semantics. The hierarchical RNN has two levels, with the lower one for capturing mobility transitions within a trajectory of short periods and the higher one for modeling the transitions between two trajectories of long periods. For the second issue, we propose a *spatial context extractor* to extract the semantics of a user's current location from its spatial neighbors. In addition, we use a *periodicity extractor* to automatically select the highly correlated historical records to derive users' periodic patterns without using manually designed features. For the third challenge, we propose a novel co-attention based *social & temporal context extractor*, which extracts two contexts, namely a social context and a temporal context. The social context models the interactions between a user and his/her friends at different time slots and is used for augmenting the user's data for mobility prediction. The temporal context captures a user's preference on time slots for activities based on his/her friends' preferences and is used for time prediction. In summary, we make the following contributions:

- We propose a new problem of joint mobility and time prediction, which has applications in many scenarios such as targeted advertising, taxi service and crowd congestion control.
- We design a context-aware deep model called DeepJMT which consists of (1) a hierarchical RNN based sequential dependency encoder to capture a user's mobility regularities and temporal patterns; (2) a spatial context extractor and a periodicity context extractor which could extract a location's semantics and the user's periodicity, respectively; and (3) a co-attention based social & temporal context extractor which could extract the mobility and temporal evidence from social relationships to alleviate the data sparsity problem.
- We conduct extensive experiments on three real-world datasets and the experimental results show that our model outperforms the existing methods for both mobility prediction and time prediction tasks. For instance, our DeepJMT model outperforms the best baseline by 8.2% and 12.1% in terms of HR@5 and MAP, respectively, on the TKY dataset.

The rest of the paper is organized as follows. In Section 2, we discuss the related work on mobility prediction and time prediction. Then we present the problem definition and preliminaries in Section 3. In Section 4, we explain the details of the proposed DeepJMT model. The experimental results are presented in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

2.1 Mobility Prediction

Mobility prediction has attracted much attention partly due to the prominence of location based services. Different from general POI recommendation [14, 32] which predicts multiple potential locations that a user will visit in the future, mobility prediction only focuses on the next location a user will visit. Some pattern-based methods [18, 34] are proposed to extract explicit user mobility patterns from historical data with prior knowledge for next location prediction. However, they cannot capture undefined and complicated mobility regularities in the data. Some model-based methods [1, 3, 17, 24] are proposed to characterize the mobility patterns for mobility prediction. Recently, rather than merely modeling sequential patterns, many variants have been proposed for exploiting additional contextual information. Feng et al. [9] propose a metric embedding learning method to predict the next location, where geographical and temporal information is incorporated. Feng et al. [8] propose a method called POI2vec to learn low-dimensional latent representations of locations by considering both the sequential transitions and geographical influence and the learned representations are then used for mobility prediction. Liu et al. [13] propose an RNN based method called STRNN which incorporates some spatial and temporal context. Yao et al. [31] use additional textual information and its semantic vector as auxiliary information for mobility prediction. DeepMove [7] is the state-of-the-art method for mobility prediction. It uses RNN to embed time and user factors and applies attention mechanism to capture the periodical effect of mobility. However, none of these techniques are capable of predicting the time associated with the next location. In contrast, we propose a novel framework to achieve joint mobility and time prediction.

2.2 Time Prediction

Temporal point process [2], which models the time of a sequence of discrete random events (more background could be found in Section 3.2), is the most popular technique on temporal modeling for time prediction [5, 26, 27, 29]. Du et al. [5] study the marked temporal point process which is to predict next event type and time simultaneously. They propose to use recurrent neural networks for learning a conditional intensity function which characterizes a temporal point process, but it does not consider relevant signals in the context of human mobility. Xiao et al. [27] propose to use a kernel function rather than a conditional intensity function for characterizing a temporal point process. Xiao et al. [26] and Yan et al. [29] propose to incorporate Generative Adversarial Network (GAN) with temporal point process with the goal of generating point process instances that are as close as authentic temporal distribution. However, these methods only focus on temporal modelling but are not concerned about the prediction of event type.

3 PROBLEM FORMULATION AND PRELIMINARIES

In this section, we define the problem of joint mobility and time prediction. Then we provide some background knowledge of temporal point process.

3.1 Problem Formulation

Let $\mathcal{U} = \{u_1, u_2, ..., u_M\}$ be a set of users, $\mathcal{L} = \{l_1, l_2, ..., l_N\}$ be a set of locations with geographical coordinates, and $G = (\mathcal{U}, \mathcal{E})$ be a social relationship graph, where each $u \in \mathcal{U}$ is a user and $e \in \mathcal{E}$ is a social connection.

Given a user u, we denote his/her mobility records by M_u to be a sequence of spatial-temporal points, i.e., $M_u = \{p_1, p_2, ..., p_{|M_u|}\}$. Each point p_i has a location identification $l_i \in \mathcal{L}$ and a timestamp t_i, i.e., $p_i = (l_i, t_i)$. Considering that two consecutive spatial-temporal points with a large time gap in-between do not have strong correlation [31], i.e., the previous point has little influence on the next one, we split the whole mobility records of a user u into non-overlapping trajectories $S_u = \{s_1^u, s_2^u, ...\}$, where $s_i^u = \{p_1^i, p_2^i, ...p_n^i\}$ is a subsequence of M_u with the maximum time gap between two consecutive points at most σ_t (e.g., 8 hours).

Problem Statement: Given a user u's current trajectory $s_k^u = \{p_1^k, p_2^k, ..., p_m^k\}$ and his/her historical trajectories $s_1^u, s_2^u, ..., s_{k-1}^u$, the social relationship graph G and the location set $\mathcal{L}$ with geographical coordinates, the problem is to predict the next point p_{m+1}^k, i.e., (l_{m+1}, t_{m+1}).

3.2 Temporal Point Process

Temporal point process is a powerful tool for modeling the time of a sequence of random events, such as earthquakes and aftershocks [19] and information diffusion on graph networks [6].

We usually use a *conditional intensity function* $\lambda(t)$ to model the time for the next event given the history of all the previous events, and it could be expressed as $\lambda(t) = \frac{f(t|\mathcal{H}_{t_n})}{1-F(t|\mathcal{H}_{t_n})}$, where $\mathcal{H}_{t_n}$ denotes all the previous events $(t_1, ..., t_{n-1}, t_n)$, $f\left(t|\mathcal{H}_{t_n}\right)$ is the conditional density function for time $t > t_n$, and $F\left(t|\mathcal{H}_{t_n}\right)$ is the corresponding cumulative distribution function.

If we consider an infinitesimal interval $[t, t + \mathrm{d}t]$ in which the event can happen at most once, we have

$$\begin{aligned}\lambda(t)\mathrm{d}t &= \frac{f\left(t|\mathcal{H}_{t_n}\right)\mathrm{d}t}{1 - F\left(t|\mathcal{H}_{t_n}\right)}\\ &= \frac{\mathbb{P}\left(t_{n+1} \in [t, t + \mathrm{d}t], t_{n+1} \notin (t_n, t)\,|\mathcal{H}_{t_n}\right)}{\mathbb{P}\left(t_{n+1} \notin (t_n, t)\,|\mathcal{H}_{t_n}\right)}\\ &= \mathbb{P}\left(t_{n+1} \in [t, t + \mathrm{d}t]|t_{n+1} \notin (t_n, t), \mathcal{H}_{t_n}\right)\\ &= \mathbb{E}\left[N([t, t + \mathrm{d}t])|t_{n+1} \notin (t_n, t), \mathcal{H}_{t_n}\right]\end{aligned} \tag{1}$$

where $N([t, t + dt])$ denotes the number of points falling in an interval $[t, t + dt]$ (note that $N([t, t + dt])$ could be either 0 or 1). It can be shown that the conditional intensity function specifies the expected number of events happening during a time interval conditional on the past. Depending on how the conditional intensity function is specified, we get different models of temporal point process. For example, if we choose $\lambda(t)$ to be a constant, it corresponds to a homogeneous Poisson process and if we choose $\lambda(t)$ to be $\mu + \alpha \sum_{t_i < t} \exp\left(-(t - t_i)\right)$, where μ and α are parameters, it becomes the famous Hawkes process. In this paper, we follow [5] by modeling the conditional intensity function with a recurrent neural network, which has been shown to be capable of modeling a general non-linear dependency over historical events.

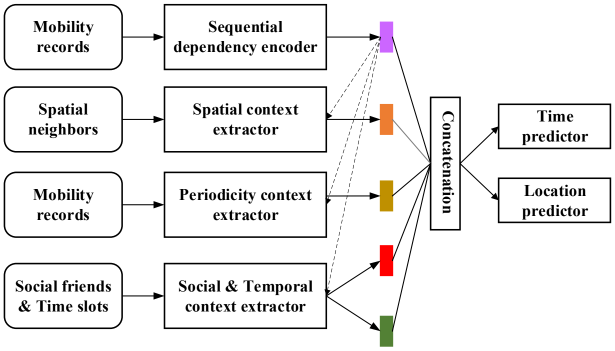

Mobility records
Sequential dependency encoder
Spatial neighbors
Spatial context extractor
Mobility records
Periodicity context extractor
Social friends & Time slots
Social & Temporal context extractor
Concatenation
Time predictor
Location predictor

Figure 2: Overview structure of DeepJTM. DeepJMT is composed of a sequential dependency encoder, a spatial context extractor, a periodicity context extractor, a social & temporal context extractor and two predictors for mobility prediction and time prediction respectively.

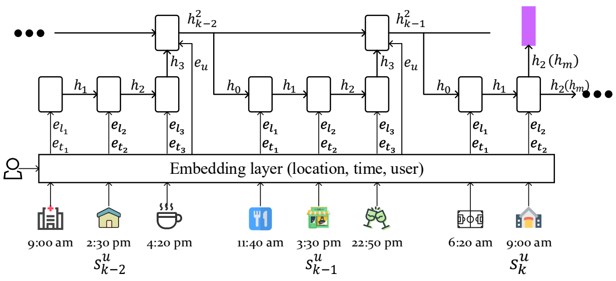


Figure 3: Structure of sequential dependency encoder. We show the latest three trajectories s_{k-2}^u, s_{k-1}^u, and s_k^u of user u till the current spatial-temporal point p_m^k.

4 PROPOSED MODEL

In this section, we introduce the details of the proposed DeepJMT model, namely those of its four components as shown in Figure 2.

4.1 Sequential Dependency Encoder

Sequential dependency encoder, as shown in Figure 3, corresponds to a hierarchical recurrent neural network (RNN), where GRU units [4] are used at both the low-level RNN and the high-level RNN.

Consider a trajectory of u $s_j^u = \{p_1^j, p_2^j, ..., p_n^j\}$ of length n where $p_i^j = (l_i, t_i)$. We embed user u and his/her spatial-temporal point l_i and t_i via an embedding layer to vectors e_u, e_{l_i}, and e_{t_i} and then feed these vectors as the input to the hierarchical RNN.

The hierarchical RNN involves two levels, namely the low-level RNN and the high-level RNN. The low-level RNN is for modeling transitions within a trajectory and the high-level RNN is for modeling transitions between trajectories. We feed the location and time embeddings e_{l_i} and e_{t_i} to low-level RNN and obtain the hidden

RIGHTSLINK

state

$$h_i = \text{GRU}_{low}(e_{l_i} \oplus e_{t_i}, h_{i-1}) \tag{2}$$

where $\oplus$ means the concatenation operation. The produced hidden state is regarded as the latent representation about the mobility status at this time step. Each spatial-temporal point is sent to low-level RNN iteratively and the last hidden state of the whole trajectory s^u_j, i.e., h_n, together with the user embedding e_u are sent to the high-level RNN. That is, at the end of each trajectory s^u_j, we obtain a hidden state

$$h^2_j = \text{GRU}_{high}(e_u \oplus h_n, h^2_{j-1}) \tag{3}$$

the hidden state h^2_j is then fed to be the initial hidden state h_0 of the low-level RNN similarly as Equation 2.

With the hierarchical RNN, users' long term sequential mobility patterns are better captured than non-hierarchical RNN structure for the following reasons. First, user's mobility records are modeled in a hierarchical way, where intra-transitions within a trajectories is modeled in low-level RNN and inter-transitions between two trajectories is modeled in high-level RNN. Thus the sequence length is greatly reduced for both two levels compared to non-hierarchical structure and the model does not suffer from the problem of a degraded performance for very long sequences of mobility records. Second, such hierarchical structure is able to distinguish the boundary of each trajectory with the hierarchical structure, so it preserves the trajectory information and is more capable of modeling mobility records.

4.2 Spatial Context Extractor

The spatial context extractor is designed to extract the semantics about a user's current location by leveraging its spatial neighbors (i.e., neighboring locations). The rationale is that a group of locations, when considered together, could provide some information about the functionality of the region in which these locations are located. For illustration, consider an example where there is a user at a restaurant. In the case that there are some clothing stores, supermarkets and a cinema nearby, it is very likely that the user is at a shopping mall and he/she is doing some shopping or entertainment. In another case that the locations nearby are mainly residences and parks, it is very likely that the user is doing some daily routine such as having a meal and the user's activity is less likely to be about shopping or entertainment. Intuitively, such semantics which infers the activity types would help with more accurate mobility prediction and we call this kind of semantics as spatial context.

Inspired by the recent advances of graph neural network (GNN) [11, 23], we model the spatial context of a location l, denoted by c_l, as the aggregation of all spatial neighbors as below.

$$c_l = g_l(\sum_{l_i \in C(l)} \alpha_{l_i} e_{l_i}) \tag{4}$$

where $C(l) = \{l_1, l_2, ..., l_n\}$ is the set containing spatial neighbors of location l, α_{l_i} is the corresponding weight for l_i, and $g_l(\cdot)$ is a feed-forward neural network. One simple method to derive the spatial context is to use mean aggregator for spatial neighbors, i.e., $\alpha_{l_i} = 1/|C(l)|$. Nevertheless, this method does not take into account the geographical distances between locations in $C(l)$ and location l or the dynamic influence between these locations and the user's current mobility status.

In this paper, we propose to use both the geographic distances and the dynamic influence for defining the weights. Specifically, according to the First Law of Geography, everything is related to everything else, but near things are more related than distant things [22]. To incorporate geographical distances properly, we adopt a Gaussian based kernel to distribute weights among spatial neighbors. The kernel function is defined as follows:

$$D(l_x, l_y) = \exp(-\frac{dist(l_x, l_y)}{\beta}) \tag{5}$$

where $dist()$ is the distance measure between two locations such as haversine distance and β is a scaling parameter.

To measure the dynamic influence of different spatial neighbors, we adopt the attention mechanism to calculate the influential strengths for different spatial neighbors. Specifically, given the latent representation h_m of a user's mobility status, the dynamic influence is defined using a bilinear operation [16] as follows:

$$q^s_{l_i} = h^\top_m W_l e_{l_i} \tag{6}$$

where $W_l \in \mathbb{R}^{d_h \times d_l}$ are learnable parameters. Note that the mechanism used in [23] can also be adopted to derive the influential strengths. Finally, based on both the geographical distances and the dynamic influence, we define the weights as follows:

$$\alpha_{l_i} = \frac{\exp\left(q^s_{l_i} \cdot D(l_i, l)\right)}{\sum_{l_\tau \in C(l)} \exp\left(q^s_{l_\tau} \cdot D(l_\tau, l)\right)} \tag{7}$$

4.3 Periodicity Context Extractor

According to some existing studies [3, 33], human mobility often manifests multi-level periodicity, including daily, weekly, and monthly routines. Since the periodicity phenomenon is reflected by not only the mobility that is close to the current step but also that of steps away. Following [7], we use an attention based RNN to extract periodical patterns from mobility records. Specifically, for each spatial-temporal point $p_i = (l_i, t_i)$ of mobility records in historical trajectories, we feed the embedded entities of location, time and user into a GRU denoted by GRU_{period}, and then obtain a sequence of hidden state $h'_1, h'_2, ...h'_k$ iteratively as follows.

$$h'_i = \text{GRU}_{period}\left(e_{l_i} \oplus e_{t_i} \oplus e_u, h'_{i-1}\right) \tag{8}$$

We calculate the correlation scores for previous mobility records by attention mechanism as in Equation 9. In this case, the records that are more related would be given higher scores regardless of how far they are from the current step:

$$q^h_{h'_i} = h^\top_m W_g h'_i \tag{9}$$

where $W_g \in \mathbb{R}^{d_h \times d_g}$ is a transformation matrix. We normalize the correlation scores and then compute the periodicity context which we denote by c_p as follows.

$$\alpha_{h'_i} = \frac{\exp\left(q^h_{h'_i}\right)}{\sum_{\tau=1}^{k} \exp\left(q^h_{h'_\tau}\right)}, c_p = \sum_{i=1}^{k} \alpha_{h'_i} h'_i \tag{10}$$

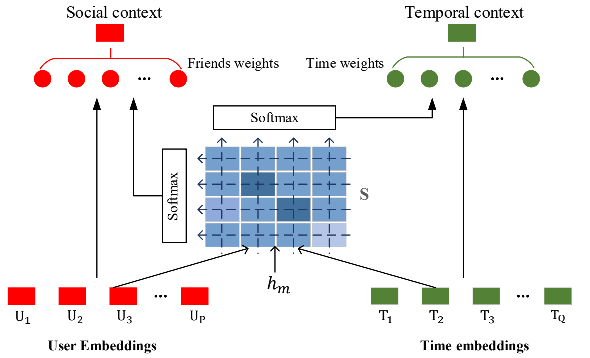


Figure 4: Structure of social & time context extractor

Note that we also tried to explicitly extract and store the periodic patterns by utilizing the existing strategies in [35], but we did not observe improvement in our preliminary experiments.

4.4 Social & Temporal Context Extractor

Human mobility data usually suffers from a data sparsity problem, i.e., some users' mobility records are quite limited [25], and as a result, it is difficult to capture their mobility regularities and temporal patterns. However, a user usually has similar behaviors and share similar interests with his/her friends [12, 20]. For example, if a user's friends have frequent mobility records at night, the user would probably be also active at night as a typical characteristic of temporal patterns. If the user's friends go to gyms frequently, the user would probably go to gyms regularly as a typical characteristic of mobility regularities. Motivated by this, we propose to leverage the social relationships to extract useful mobility and temporal evidence to facilitate both mobility and time prediction. Moreover, since a user may have different degrees of similarity for different friends during different time slots, e.g., a user has similar behaviors as his/her colleagues during workdays and those as his/her families during non-working days, we propose to use co-attention mechanism [15, 28] to jointly reason about the co-attention weights of different user-time pairs. We then make use of these attention weights to compute both a social context and a temporal context. The social context captures the aggregated influence on a user from his/her friends based on their similarity, and the temporal context captures a user's preference on time slots based on his/her friends' preferences. We call this process the social & temporal context extractor, which is presented in Figure 4.

Given the representation of mobility state h_m from the sequential dependency encoder, a list of user embeddings $U \in \mathbb{R}^{P \times d_u}$ from u's social friends where P denotes the number of u's friends, and time embeddings $T \in \mathbb{R}^{Q \times d_t}$, where Q is the number of time slots, similar to [21], we compute each entry of an affinity weight matrix $S \in \mathbb{R}^{P \times Q}$ as follows:

$$S_{ij} = [W_u h_m \oplus U_i]^\top W_s [W_t h_m \oplus T_j] \tag{11}$$

where W_u, W_t and W_s are parameter matrices, $U_i \in \mathbb{R}^{d_u}$ is the i-th row of U and $T_j \in \mathbb{R}^{d_t}$ is the j-th row of T. The purpose of this step is to calculate a score which indicates the influence of $u's$ i-th friend during time slot j considering user's current mobility status.

Then we compute a social context denoted by c_u and a temporal context denoted by c_t as follows.

$$c_u = g_u(softmax(pool_{row}(S))^\top \cdot U) \tag{12}$$

$$c_t = g_t(softmax(pool_{col}(S))^\top \cdot T) \tag{13}$$

where $pool_{row}(\cdot)$ and $pool_{col}(\cdot)$ are pooling functions on rows and columns, respectively, $softmax(\cdot)$ is the softmax function for normalizing the scores, and $g_u(\cdot)$ and $g_t(\cdot)$ are feed-forward neural networks. Here we choose to use the max-pooling function to select the most representative time slot for each friend and the most representative friend for each time slot.

4.5 Training and Inference

Based on the hidden state h_m, the spatial context c_l, the periodicity context c_p, and the social context c_u, we compute a probability distribution using a softmax function for mobility prediction as follows.

$$P(l_{m+1} = l_i | \mathcal{H}_{t_m}) = \frac{\exp\left(W_i^\top \theta_m^l\right)}{\sum_{\tau=1}^{N} \exp\left(W_\tau^\top \theta_m^l\right)} \tag{14}$$

where $\theta_m^l = h_m \oplus c_l \oplus c_p \oplus c_u$ is the concatenation of the hidden state, the spatial context, the periodicity context and the social context, and W_i is the i-th row of projection matrix W. Similar to [5], we model the conditional intensity function using neural networks. Specifically, the conditional intensity function is based on both the hidden state h_m and the temporal context c_t as follows.

$$\lambda(t) = \exp\left(v^\top \cdot \theta_m^t + w \cdot \xi_m + b\right) \tag{15}$$

where $\theta_m^t = h_m \oplus c_t$, $\xi_m = t - t_m$, t_m is the timestamp of last spatial-temporal point and t is the time variable, v and w are model parameters, and b is the bias term. Based on a conditional intensity function, we can derive the corresponding conditional density function $f(t)$:

$$f(t) = \lambda(t) \exp\left(-\int_{t_m}^{t} \lambda(\tau) d\tau\right) \tag{16}$$

By plugging in the conditional intensity function in Equation 16, we obtain the full expression of the conditional density function as follows.

$$\begin{aligned} f(t) = \exp\{&v^\top \cdot \theta_m^t + w \cdot \xi_m + b + \frac{1}{w} \exp\left(v^\top \cdot \theta_m^t + b\right) \\ &- \frac{1}{w} \exp\left(v^\top \cdot \theta_m^t + w \cdot \xi_m + b\right)\} \end{aligned} \tag{17}$$

We define the loss function as a combination of time prediction loss and mobility prediction loss. Specifically, we choose to optimize the negative log-likelihood loss defined by:

$$L = -\sum_{s_t^u \in S_u} \sum_{m=1}^{|s_t^u|-1} \left(\log P\left(l_{m+1} | \mathcal{H}_{t_m}\right) + \log f\left(t_{m+1}\right)\right) \tag{18}$$

During the training stage, model parameters as well as all the embeddings could be learned in an end-to-end manner through back propagation.

In the inference stage, we sort and pick top-K locations with the highest probabilities as the predicted locations for mobility prediction, where K can be chosen according to different application

Table 1: Statistics of three datasets

Dataset	#users	#locations	#trajectories	Avg. traj. for a user
NYC	1069	8,358	34,439	32.2
IST	7960	13,844	179,751	22.6
TKY	4662	11,747	156,982	33.7

requirements. For time prediction, the predicted time for the next location is calculated using the following integration to minimize the L2 loss.

$$\hat{t}_{m+1} = \mathbb{E}[\hat{t}_{m+1}|\mathcal{H}_{t_m}] = \int_0^\infty t \cdot f(t)dt \qquad (19)$$

The integration of the density distribution of the point process does not have a closed-form solution. We use numerical methods to approximate the integration.

5 EXPERIMENTS

In this section, we evaluate our DeepJMT model on three real-world datasets. We compare it to several state-of-the-art models, showing the superiority of our proposed model for both mobility and time prediction tasks.

5.1 Experimental Settings

5.1.1 Datasets. Our experiments are conducted on three real-world datasets, which are extracted from the public Foursquare check-in data [30] from April 2012 to January 2014. Specifically, we select 3 cities where users have a large number of mobility records: New York City (NYC), Istanbul (IST) and Tokyo (TKY). We filter out those locations which are visited by fewer than 10 times in each city and extract the mobility records for each user. We then segment the mobility records of a user into trajectories such as the the time gap between two consecutive points within a trajectory is at most 6 hours. We then filter out users with fewer than 5 trajectories. The statistics of the three processed datasets are summarized in Table 1.

We partition each dataset into training set, validation set and test set. Specifically, for each user, we use the earliest 70% trajectories as the training set, the most recent 20% trajectories as the test set and the remaining 10% trajectories as the validation set.

5.1.2 Evaluation Metrics and Parameter Setting. We use different metrics for evaluating mobility prediction and time prediction tasks. For the mobility prediction task, we use Hit Ratio@K (**HR@K**) and Mean Average Precision (**MAP**). HR@K is the percentage that a list of predictions with length K covers the ground truth location. MAP is a widely used metric for ranking tasks. Higher values of HR@N and MAP mean better performance. For the time prediction task, we use Mean Absolute Error (**MAE**) which measures the closeness between the predicted time and the real time. Lower values of MAE mean better performance.

In the experiments, the embedding sizes of user, time and location are set to be 100, 50 and 60, respectively and the dimension of the GRU hidden state is set to be 150. We discretize the number of time embeddings using 15 minutes long time bins. The model is optimized by Adam optimizer with a learning rate 0.0005, and the batch size is set to be 16 for the NYC dataset and 32 for the other two datasets.

5.1.3 Compared Methods. Since DeepJMT is able to perform mobility and time prediction simultaneously, we compare our model with three types of baselines. The first type can only perform mobility prediction The second type includes classical temporal point process models which can only perform time prediction. The third type of baselines can perform both mobility and time prediction.

Mobility prediction only

- PRME[9]: a metric embedding based method which embeds users and locations into the same latent space to capture user preference and sequential transition for mobility prediction. It only considers the geographical distance and time interval as the context information.
- GRU[4]: an RNN model with GRU unit. We use time, location and user embeddings as we do for DeepJMT and then send them to this model as inputs.
- STRNN[13]: an RNN-based model and incorporates spatial and temporal context into an RNN cell using linear combination of time-specific and distance-specific transition matrices as the context information.
- DeepMove[7]: the state-of-the-art model for mobility prediction. It models periodical patterns using attention mechanism and uses GRU to encode both recent and historical trajectories.

Time prediction only

- Hawkes Process: a classical temporal point process with the conditional intensity function as introduced in Section 3.2.
- Self-correcting (SC) Process: another type of temporal point process in which the conditional intensity function can be written as: $\lambda^*(t) = \exp\left(\mu t - \sum_{t_i<t}\alpha\right)$ with parameters λ and α.

Mobility and time prediction

- RMTPP[5]: an RNN based model designed for predicting the time and the type of next event given the types and timestamps of previous events. We adapt it for mobility prediction by regarding each location as an event type.
- IntensityRNN[27] : it is also able to predict the time and the type of next event. Instead of utilizing the intensity function as the indicator of time evolution, this model predicts the time directly and adopt a Gaussian penalty loss function.
- DeepJMT: our proposed method, which utilizes the spatial, periodicity and social-temporal contexts to perform mobility and time prediction in a unified way. To verify the effect of different components, we also conduct experiments on the degenerated DeepJMT models: (1) **Deep-So**: the variant of Deep JMT without the social & temporal context extractor, (2) **DeepJMT-Se**: the variant of DeepJMT without the sequential dependency encoder, (3) **DeepJMT-Sp**: the variant of Deep JMT without the spatial context extractor, and (4) **DeepJMT-Pe**: the variant of Deep JMT without the periodicity context extractor.

5.2 Performance Comparison

We first compare all methods in terms of the mobility and time prediction effectiveness. For the mobility prediction task, the results are shown in Table 2 and we have a few observations.

Table 2: Mobility prediction results in terms of HR@N and MAP on three datasets

Methods	NYC				TKY				IST			
	HR@5	HR@10	HR@20	MAP	HR@5	HR@10	HR@20	MAP	HR@5	HR@10	HR@20	MAP
PRME	0.2023	0.2696	0.3140	0.0679	0.2210	0.2738	0.3312	0.0411	0.0789	0.1325	0.1968	0.0121
GRU	0.2927	0.3416	0.3834	0.1242	0.2747	0.3433	0.4142	0.0879	0.1524	0.2108	0.2778	0.0581
STRNN	0.2771	0.3365	0.3789	0.1183	0.2808	0.3428	0.4085	0.0744	0.1481	0.2168	0.2766	0.0551
DeepMove	0.3847	0.4605	0.5271	0.1483	0.3642	0.4481	0.5379	0.1106	0.2386	0.3077	0.3726	0.0832
RMTPP	0.3532	0.4253	0.4871	0.1424	0.3452	0.4191	0.4869	0.1067	0.1829	0.2385	0.3028	0.0696
IntensityRNN	0.3552	0.4244	0.4832	0.1419	0.3412	0.4117	0.4789	0.1047	0.1740	0.2269	0.2900	0.0657
DeepJMT-So	0.4093	0.4882	0.5496	0.1595	0.4027	0.4843	0.5597	0.1224	0.2498	0.3184	0.3959	0.0913
DeepJMT-Se	0.3891	0.4418	0.5059	0.1375	0.3821	0.4677	0.5366	0.1103	0.2343	0.3010	0.3706	0.0764
DeepJMT-Sp	0.4021	0.4806	0.5381	0.1540	0.3843	0.4601	0.5306	0.1142	0.2345	0.3008	0.3678	0.0874
DeepJMT-Pe	0.4066	0.4853	0.5420	0.1602	0.4012	0.4838	0.5588	0.1220	0.2504	0.3201	0.3968	0.0903
DeepJMT	**0.4122**	**0.4924**	**0.5536**	**0.1623**	**0.4050**	**0.4876**	**0.5622**	**0.1240**	**0.2541**	**0.3226**	**0.3997**	**0.0928**

First, among the baseline models, PRME has the worst performance as it is a metric embedding method and less capable of modeling complex interactions between the spatial-temporal context and mobility records compared to the other neural network based methods.

Second, for all neural network based methods, RMTPP and IntensityRNN obtain much better results than STRNN despite that STRNN also incorporates spatial and temporal contexts. The major reason is that learning a unified framework for two highly correlated mobility prediction and time prediction tasks improves the model performance. Besides, STRNN even has a worse performance than GRU model because STRNN fuses the spatial-temporal information using the linear combination of two context matrices thus failing to model the nonlinearlity relationship of spatial and temporal information. Moreover, DeepMove achieves the best results among all the baseline methods. It is because other than being able to capture the sequential transitions with spatial-temporal context, it is also able to capture users’ periodic patterns.

Third, our DeepJMT model outperforms all the baseline methods on three datasets. Compared to DeepMove, DeepJMT uses a hierarchical RNN structure which is more capable of capturing mobility regularities and temporal patterns and incorporates more context information, and thus DeepJMT achieves better performance. DeepJMT still works when some types of context information are not available, but we observe that removal of each model component would lead to different degrees of performance degradation on all three datasets. This indicates that all the components of DeepJMT contribute to the model performance. Specially, we find that the performances of DeepJMT-Se and DeepJMT-Sp drop drastically compared to DeepJMT. This justifies our motivation that neighboring locations can reflect the semantics of a location and hierarchical RNN structures can capture the long term dependency while preserving the mobility semantics. Besides, DeepJMT-Pe and DeepJMT-So also result in a worse performance which illustrates the effectiveness of considering the periodicity and social relationship information.

The results for time prediction task for baseline models and DeepJMT variants are shown in Figure 5a and 5b respectively. We have the following observations. First, Hawkes and self-correcting process perform consistently worse than RNN based temporal process methods, which indicates that time evolution is very complicated and a manually specified conditional intensity function might not truly reflect the underlying time evolution process. Second, although RMTPP and IntensityRNN have relatively the same performance on mobility prediction, RMTPP outperforms IntensityRNN for time predition. This suggests that using intensity function is more suitable for modeling temporal process compared to directly predicting the time. Third, for all the model variants of DeepJMT, the performance doesn’t change quite much, which might be because the mobility prediction intrinsically rely on more context information that the time prediction task does.

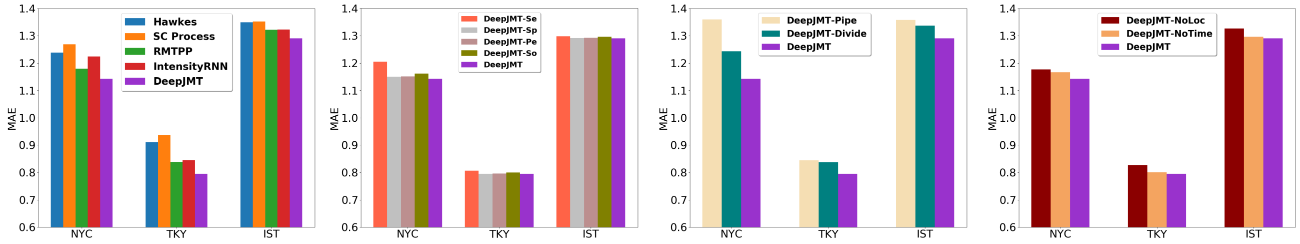
5.3 Model Analysis

5.3.1 Effect of Joint Learning Paradigm. In DeepJMT, the underlying strategy is to use a joint learning paradigm for mobility and time prediction, i.e., location predictor and time predictor operate in parallel, which we believe is better than to perform mobility and time prediction tasks in sequence. To evaluate the effectiveness of this strategy, we consider another two training paradigms of performing mobility prediction and time prediction:

- DeepJMT-Pipe: we first train the model only for mobility prediction task. After that, we use the trained model to get a predicted location with the highest probabilities. Then we use the predicted location to extract spatial and temporal features to train another regression model for time prediction. In other words, the mobility prediction and time prediction tasks are processed in a pipeline manner.
- DeepJMT-Divide: In this paradigm, location and time prediction are still trained in an end-to-end manner with the loss the same as Equation 18. The difference is that mobility prediction and time prediction are not performed in parallel. Specifically, location predictor first produces the predicted location, and then the predicted location and temporal information are sent to another RNN to model the intensity function and then perform time prediction.

The experimental results for mobility prediction and time prediction on the three datasets are shown in the first two rows in Table 3 and Figure 5c, respectively.

RIGHTSLINK



(a) baseline models **(b) model variants** **(c) joint learning paradigm** **(d) location and time information**

Figure 5: Time prediction results in terms of MAE on three datasets

Table 3: Effect of joint learning paradigm and time information on mobility prediction

Methods	NYC				TKY				IST			
	HR@5	HR@10	HR@20	MAP	HR@5	HR@10	HR@20	MAP	HR@5	HR@10	HR@20	MAP
DeepJMT-Pipe	0.4069	0.4901	0.5508	0.1580	0.4014	0.4822	0.5585	0.1220	0.2505	0.3187	0.3981	0.0905
DeepJMT-Divide	0.4020	0.4830	0.5443	0.1550	0.3985	0.4809	0.5560	0.1208	0.2450	0.3136	0.3917	0.0882
DeepJMT-NoTime	0.3924	0.4721	0.5346	0.1568	0.3983	0.4810	0.5567	0.1204	0.2456	0.3137	0.3920	0.0905
DeepJMT	**0.4122**	**0.4924**	**0.5536**	**0.1623**	**0.4050**	**0.4876**	**0.5622**	**0.1240**	**0.2541**	**0.3226**	**0.3997**	**0.0928**

We find that the performance of DeepJMT outperforms the other two models trained with different strategies for both mobility prediction and time prediction. Specially, there is a significant performance drop in DeepJMT-Pipe and DeepJMT-Divide models for the time prediction task, and DeepJMT also provides better performance for mobility prediction task. This is because in the joint learning paradigm, the sequential dependency encoder is trained in a multi-task learning framework. Since mobility prediction and time prediction are highly correlated, the multi-task learning framework would encourage the sequential dependency encoder to learn a better latent representation of a user's mobility records. In other words, separating these two predictors in sequence is not a good choice for two highly correlated tasks.

5.3.2 *Effect of Time and Location Information.* In the DeepJMT model, location information and time information are transformed into location and time embeddings, respectively, and they then are both sent as the input to the sequential dependency encoder. We believe that the interplay of the location information and the time information can complement and enhance each other in the sequential dependency encoder, which enables the model to better encode the mobility regularities and temporal patterns. To verify the effects of the time information and the location information, we evaluate the mobility prediction performance of DeepJMT trained without the time information (we denote this variant DeepJMT-NoTime) and also the time prediction performance of DeepJMT trained without the location information (we denote this variant DeepJMT-NoLoc). Specifically, the timestamp of each mobility record is not provided in DeepJTM-NoTime and the location identification is not provided in DeepJMT-NoLoc. Note that we do not evaluate mobility prediction performance of DeepJMT-NoLoc because it is unrealistic to predict the mobility without using any location information in historical records.

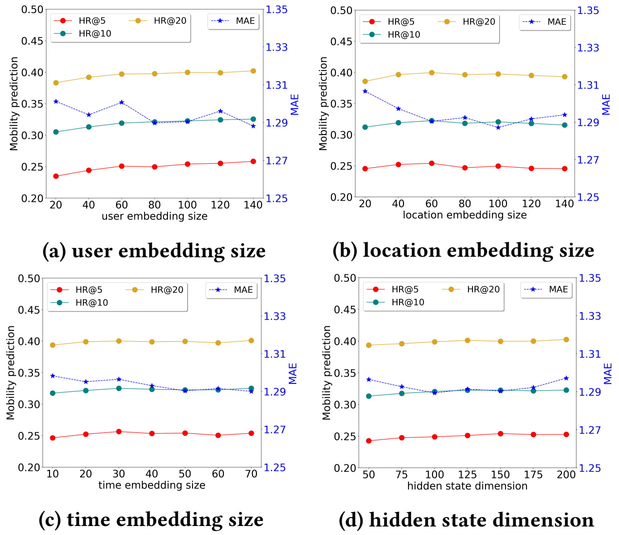
The results on the three datasets for mobility prediction and task prediction are shown in the third row in Table 3 and Figure 5d, respectively. We observe that DeepJMT, i.e., the model trained given both location and time information, achieves the best performance over all three datasets. With either type of information removed, the performance becomes worse for both tasks. It demonstrates that the time information and the location information can complement and enhance each other for mobility and time prediction.

5.4 Sensitivity of Parameters

We investigate the influence of different parameter settings of the DeepJMT model on mobility and time prediction performance. Specifically, we evaluate the sensitivity of four parameters: the location embedding size, the user embedding size, the time embedding size and the dimension of the hidden state in the sequential dependency encoder. We vary the user embedding size and the location embedding size with values in range {20, 40, 60, 80, 100, 120, 140}, time embedding size with values in range {10, 20, 30, 40, 50, 60, 70} and the dimension of the hidden state with values from range {50, 75, 100, 125, 150, 175, 200}. Except for the parameters being tested, we set the other parameters with their default values. Due to the space limitation, we only report the experimental results on the IST dataset and similar trends can be observed on the NYC and the TKY datasets.

Figure 6 presents the performance comparison when we vary the values of model parameters. In general, we find that using a larger embedding size would have more powerful representation ability and improve the performance. But it also increases the complexity of the model and makes it prone to overfit. In our experiments, we set the dimension as reported in Section 5.1.2 considering the trade-off between the effectiveness and the computational cost. In addition, we observe the four different parameters have a relatively small impact on the performance, which demonstrates the robustness of our proposed DeepJMT model.

RIGHTSLINK



(a) user embedding size

(b) location embedding size

(c) time embedding size

(d) hidden state dimension

Figure 6: Performance with varying parameters on IST

6 CONCLUSION

In this paper, we propose a novel context-aware deep model called DeepJMT for both mobility and time prediction. DeepJMT is composed of a sequential dependency encoder, a spatial context extractor, a periodicity context extractor and a social & temporal context extractor. In the model framework, we manage to utilize spatial neighbors and historical mobility records as context information and leverage the social relationship to learn more mobility and temporal evidence from friends. DeepJMT model is capable of capturing user mobility regularities and temporal patterns and achieves significant improvement over the state-of-the-art on real-world datasets. In the future, we plan to discover different fusion approaches for different context information and incorporate some other information such as user comments.

ACKNOWLEDGMENTS

This research was conducted in collaboration with Singapore Telecommunications Limited and supported by the Singapore Government through the Industry Alignment Fund - Industry Collaboration Projects Grant. Cheng Long's work is supported by MOE Tier 1 RG20/19 (S).

REFERENCES

[1] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation. In *IJCAI*. 2605–2611.

[2] Sung Chiu, Dietrich Stoyan, Wilfrid Kendall, and J Mecke. 2013. *Stochastic Geometry and Its Applications*.

[3] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *KDD*. 1082–1090.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. 1724–1734.

[5] 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 *KDD*. 1555–1564.

[6] Nan Du, Le Song, Manuel Gomez Rodriguez, and Hongyuan Zha. 2013. Scalable Influence Estimation in Continuous-Time Diffusion Networks. In *NIPS*. 3147–3155.

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

[8] Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. 2017. POI2Vec: Geographical Latent Representation for Predicting Future Visitors. In *AAAI*. 102–108.

[9] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI*. 2069–2075.

[10] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *RecSys*. 93–100.

[11] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[12] Huayu Li, Yong Ge, Richang Hong, and Hengshu Zhu. 2016. Point-of-Interest Recommendations: Learning Potential Check-ins from Friends. In *KDD*. 975–984.

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

[14] Yiding Liu, Tuan-Anh Pham, Gao Cong, and Quan Yuan. 2017. An Experimental Evaluation of Point-of-interest Recommendation in Location-based Social Networks. *Proceedings of VLDB* 10, 10 (2017), 1010–1021.

[15] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical Question-Image Co-Attention for Visual Question Answering. In *NIPS*. 289–297.

[16] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*. 1412–1421.

[17] Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. In *Ubicomp*. 911–918.

[18] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. WhereNext: a location predictor on trajectory pattern mining. In *KDD*. 637–646.

[19] Yosihiko Ogata. 1998. Space-Time Point-Process Models for Earthquake Occurrences. *Annals of the Institute of Statistical Mathematics* 50, 2 (1998), 379–402.

[20] Adam Sadilek, Henry Kautz, and Jeffrey P. Bigham. 2012. Finding Your Friends and Following Them to Where You Are. In *WSDM*. 723–732.

[21] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. In *KDD*. 2309–2318.

[22] W. R. Tobler. 1970. A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography* 46, sup1 (1970), 234–240.

[23] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[24] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. 2015. Regularity and Conformity: Location Prediction Using Heterogeneous Mobility Data. In *KDD*. 1275–1284.

[25] Fei Wu and Zhenhui Li. 2016. Where Did You Go: Personalized Annotation of Mobility Records. In *CIKM*. 589–598.

[26] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Xiaokang Yang, Le Song, and Hongyuan Zha. 2017. Wasserstein Learning of Deep Generative Point Process Models. In *NIPS*. 3250–3259.

[27] Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen M. Chu. 2017. Modeling the Intensity Function of Point Process Via Recurrent Neural Networks. In *AAAI*. 1597–1603.

[28] Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic Coattention Networks For Question Answering. In *ICLR*.

[29] Junchi Yan, Xin Liu, Liangliang Shi, Changsheng Li, and Hongyuan Zha. 2018. Improving Maximum Likelihood Estimation of Temporal Point Process via Discriminative and Adversarial Learning. In *IJCAI*. 2948–2954.

[30] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudré-Mauroux. 2019. Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach. In *WWW*. 2147–2157.

[31] Di Yao, Chao Zhang, Jian-Hui Huang, and Jingping Bi. 2017. SERM: A Recurrent Model for Next Location Prediction in Semantic Trajectories. In *CIKM*. 2411–2414.

[32] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. 2013. Time-aware point-of-interest recommendation. In *SIGIR*. 363–372.

[33] Quan Yuan, Wei Zhang, Chao Zhang, Xinhe Geng, Gao Cong, and Jiawei Han. 2017. PRED: Periodic Region Detection for Mobility Modeling of Social Media Users. In *WSDM*. 263–272.

[34] Chao Zhang, Jiawei Han, Lidan Shou, Jiajun Lu, and Thomas F. La Porta. 2014. Splitter: Mining Fine-Grained Sequential Patterns in Semantic Trajectories. *Proceedings of VLDB* 7, 9 (2014), 769–780.

[35] Ali Zonoozi, Jung-jae Kim, Xiao-Li Li, and Gao Cong. 2018. Periodic-CRN: A Convolutional Recurrent Model for Crowd Density Prediction with Recurring Periodic Patterns. In *IJCAI*. 3732–3738.