A Learning-Based POI Recommendation With Spatiotemporal Context Awareness

Yi-Cheng Chen, Tipajin Thaipisutikul, and Timothy K. Shih

***Abstract*—Due to the great advances in mobility techniques, an increasing number of point-of-interest (POI)-related services have emerged, which could help users to navigate or predict POIs that may be interesting. Obviously, predicting POIs is a challenging task, mainly because of the complicated sequential transition regularities, and the heterogeneity and sparsity of the collected trajectory data. Most prior studies on successive POI recommendation mainly focused on modeling the correlation among POIs based on users' check-in data. However, given a user's check-in sequence, generally, the relationship between two consecutive POIs is usually both time and distance subtle. In this article, we propose a novel POI recommendation system to capture and learn the complicated sequential transitions by incorporating time and distance irregularity. In addition, we propose a feasible way to dynamically weight the decay values into the model learning process. The learned awareness weights offer an easy-to-interpret way to translate how much each context is emphasized in the prediction process. The performance evaluations are conducted on real mobility datasets to demonstrate the effectiveness and practicability of the POI recommendations. The experimental results show that the proposed methods significantly outperform the state-of-the-art models in all metrics.**

***Index Terms*—Human mobility, machine learning, point-of-interest (POI) recommendation, recurrent neural network (RNN), spatiotemporal data.**

I. Introduction

THESE DAYS, due to the popularity of location-based social networks (LBSNs), people can easily share their thoughts, comments, pictures, etc., along with the location with their friends. This rapid rise in online check-in data provides a great opportunity to understand the mobility behaviors of people from their historical traces, and to foresee their future footprints. Point-of-interest (POI) prediction is one essential technique for trajectory analytics which has widespread applications, such as smart transportation, urban planning, and tourism recommendation, to name a few.

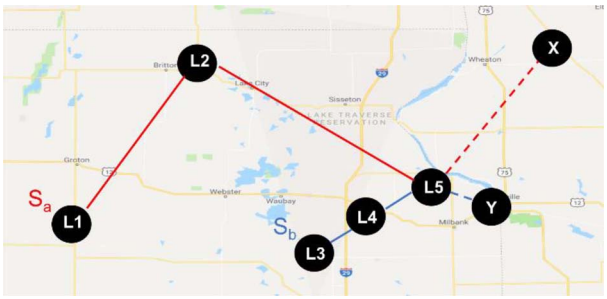


Fig. 1. Example of recommendation considered distance difference between two consecutive locations.

In general, obtaining revealing insights from trajectory check-in data is not an easy task since information is noisy, heterogeneous, and dynamic in time and distance asymmetry. The prior research focused on this domain mainly utilizes pattern discovery [17], [28], [30], [33], [44] to find some common characteristics of users, and then predicts the future locations based on the analysis results. Such methods usually suffer from the one-sided nature of the predefined patterns and ignore the user's preferences. Therefore, model-based methods [16], [29], [52] were recently proposed to tackle a mobility prediction problem. They utilize sequential models [e.g., the Markov chain and recurrent neural networks (RNNs)] to learn the transition regularities and the model parameters from the given training corpus. However, these models usually do not consider the irregularity of the spatiotemporal changes between two consecutive check-ins.

In real applications, the spatial and temporal differences are important; in a check-in trajectory, different intervals between two locations may reveal different information. By including this essential factor, we could make more precise predictions. For example, in Fig. 1, suppose there are two hot spots X and Y, and the check-in sequences of users U_a and U_b are $S_a : L_1 \rightarrow L_2 \rightarrow L_5$ and $S_b : L_3 \rightarrow L_4 \rightarrow L_5$, respectively. When predicting the next interesting spot, a system may prefer recommending X for U_a since the distance difference between the consecutive locations in S_a is long. This might indicate that U_a drives a car during his or her itinerary. In contrast, due to the distance between consecutive locations being close in S_b, U_b may visit every tourist spot by walking. Hence, Y is more suitable for recommendation to U_b.

Manuscript received January 12, 2020; revised April 17, 2020; accepted June 4, 2020. This work was supported in part by the Ministry of Science and Technology, Taiwan, under Project MOST 108-2221-E-008-063-MY3 and Project MOST 109-2634-F-008-008. This article was recommended by Associate Editor B. Ribeiro. *(Corresponding author: Yi-Cheng Chen.)*

Yi-Cheng Chen is with the Department of Information Management, National Central University, Taoyuan 320, Taiwan (e-mail: ycchen@mgt.ncu.edu.tw).

Tipajin Thaipisutikul and Timothy K. Shih are with the Department of Computer Science and Information Engineering, National Central University, Taoyuan 320, Taiwan (e-mail: 106582601@cc.ncu.edu.tw; timothyk-shih@gmail.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2020.3000733

Currently, many RNN-based learning models, such as long short-term memory (LSTM) and gated recurrent unit (GRU), are widely applied to capture structures in sequential data such as POI prediction tasks, since the relationships and hidden dependencies among locations could be trained and discovered. However, their components are designed to process sequential data with regular elapsed units between successive elements in sequences that are not suitable for POI data; the elapsed time and distance of the check-in sequences can normally vary from minutes to months and from a few to many kilometers. Some prior studies [2], [11], [24] embed time and location as input features in the model learning process. However, the divided discrete bins from the semantics of time and location may not properly capture the continuous time and distance intervals. Also, originally, the models were designed for only short-term interests and are not well designed for integrating long-term interests into the prediction tasks. A gate-learning model [50] equips LSTM with additional time gates to model time intervals which could help capture both users' short- and long-term interests more easily. However, the spatial contexts are not included in the training process, which may thus fail to properly model users' movement relations.

The aspect of both spatial and temporal differences should be considered in the model learning, since the prediction of the next move of users is mainly based on how long it takes and how far it is from the current position to the destination. Hence, in this article, we propose a novel POI recommendation system, deep navigator (DeNavi), to handle time and distance irregular intervals in longitudinal check-in trajectories. We incorporate the irregularities of time and distance between the consecutive elements in a sequence into the memory unit to boost the performance improvement of the standard recurrent networks. Intuitively, the memory cell should decay in a way that the greater the elapsed time and distance, the smaller the effect of the previous memory on the current output. Based on this assumption, the elapsed time and distance are formulated into a proper weight exploitation. Also, the prediction is not only based on the users' short-term interests but also on their long-term interests. There are three learning models in DeNavi: 1) DeNavi-LSTM considers the time and distance awareness based on LSTM and utilizes a subspace decomposition of the memory cell and hidden state by a decay function to discount the historical information according to interval differences; 2) DeNavi-GRU, likewise, also considers the time and distance awareness and utilizes lightweight learning for the hidden state by decay function; and 3) DeNavi-Alpha dynamically includes the time and distance awareness based on LSTM and utilizes the dynamic exponential weight moving average (EWMA) to adjust the importance of each context differently.

The main contributions of this article are listed as follows.

1) A novel learning model, DeNavi, is proposed to incorporate time and distance irregularities from check-in sequences. The time and distance differences are considered and included in the learning process for model training.
2) By carrying a decomposition of the memory cell out into short- and long-term effects, in the proposed model, the short-term counterpart is modified by the discounted weight and is then combined with the long-term counterpart before entering the sequential learning model.
3) We propose a feasible way to dynamically weight the decay values by integrating the EWMA into the model learning process. The learned awareness weights offer a convenient way to translate how much each context is emphasized in the prediction process.
4) We perform extensive experiments on two real-life mobility datasets. Our results demonstrate that DeNavi outperforms state-of-the-art mobility prediction models. It exhibits an outstanding generalization ability and is robust across trajectory datasets of different natures.

The remainder of this article is organized as follows. Sections II and III present related works and preliminaries, respectively. Section IV provides the proposed models. Section V details the experiments and the performance study, while Section VI presents the conclusion.

II. Related Work

POI recommendation, as a natural extension of conventional recommendation, has recently been proposed and has captivated great research interest. In this section, we provide an overview of our related works as two types: 1) traditional POI recommendation and 2) deep learning POI recommendation.

A. Traditional POI Recommendation

The hidden Markov model (HMM) [6], [16] is a method to model future movements by constructing a transition matrix probability between locations based on past trajectories. The Markov chain is often exploited for POI recommendations in LBSNs to model the sequence pattern. Cheng *et al.* [9] considered two main properties in the check-in sequence: 1) a personalized Markov chain and 2) region localization. They then proposed a novel matrix factorization method for POI recommendation that incorporates users' movement constraint and the personalized Markov chain. He *et al.* [18] modeled successive check-in behaviors by proposing a third-rank tensor that blends a personalized Markov chain with a latent pattern. LORE [45] utilizes sequential patterns to predict the probability of a user visiting a location by exploiting a dynamic location transition graph via an additive Markov chain. Note that sequential influence with geographical influence and social influence could also be fused into the recommendation framework. The authors also propose a gravity model that integrates the spatiotemporal, social, and popularity influences by estimating a power of law distribution.

Matrix factorization [30], [38], [42] is a method to discover sequential mobility signals from the trajectory and then aims to forecast mobility based on these popular patterns. The basic idea is to factor the user-item matrix into two latent matrices which represent the user and item characteristics. Xiong *et al.* [39] produced latent vectors of users, items, and time bins via factorization for POI recommendation. Cheng *et al.* [8] further employed geographic influence and social influence in a matrix factorization method for performance improvement. Cheng *et al.* [9] solved the problem

of successive POI by the factoring personalized Markov chain (FPMC) and included the geographic influence for predicting the POIs that the user may be interested in [32].

Several latent probabilistic generative models [14], [23], [25], [31], [35], [36], [40], [41] could effectively mimic users' decision-making process by utilizing the multiplication of latent variables. Fu *et al.* [14] developed a generative probabilistic model for restaurant recommendation using multiple information fusion called GC-BCoNMF. This article exploits the latent features of each aspect in multirating, such as cuisine, environment, and cost to predict ratings. The WWO [23] probabilistic graphical model integrates the user interests and the evolving sequential preference with temporal interval assessment. Liu *et al.* [25] proposed a general geographical probabilistic factor model, Geo-PFM, to capture the geographical influences. Since the probability of a user visiting a POI is inversely proportional to the geographic distance between them, Geo-PFM could identify POIs in the same region that shares similar features. Qian *et al.* [31] proposed a spatiotemporal context-aware and translation-based recommendation framework (STA) to model the third-order relationship users, POIs, and spatiotemporal contexts for large-scale POI recommendation. TRM [35] utilizes the latent topic-region variable to effectively fuse the sequential influence and cyclic pattern with personal interest to the latent space. Wang *et al.* [36] introduced a novel probabilistic-generative model, SPORE, to capture the user decision-making process for choosing spatial items based on the latent variable topic region. Yin *et al.* [40] proposed a spatial-aware user preference modeling method based on latent factors to include the influence of spatial dynamics of personal preferences. The authors leverage the spatial-aware crowd's preferences by exploiting the geographic autocorrelation of personal preferences with geographically hierarchical additive representations. Yin *et al.* [41] extended TRM to concurrently determine the semantic, temporal, and spatial patterns of check-in activities. The authors also developed a novel clustering-based branch and bound algorithm (CBB) to trim the POI search space.

B. Deep Learning POI Recommendation

With the impressive achievement of deep neural-network (DNN) models in many domains [11], [12], [22], [51], various approaches [5], [27] that leverage DNN for recommendation systems have been proposed including the POI prediction task. In order to foresee the future footprint, the latest visited POI needs to be considered. Thus, among several DNN techniques, the RNN models have been widely explored to encapsulate users' short- and long-term dependencies from the user's sequence of the observed records [4], [13], [24], [26], [49]. Cheng *et al.* [5] presented a jointly wide and deep learning method to combine the benefits of memorization and generalization for recommender systems. The deep recurrent collaborative filtering framework (DRCF) [27] utilized multilayer perception and RNN architectures to capture user–venue interactions in a CF manner from sequences of observed feedback. T-LSTM [4] was proposed for the patient subtyping task by decomposing the memory cell, and enables a time decay function on the short-term content. Then, T-LSTM feeds the output from the recurrent unit into an autoencoder to learn a powerful single representation for sequential records of patients. DeepMove [13] captured the complicated sequential transitions and the multilevel periodicity by including a multimodal embedding RNN and a historical attention unit into a traditional model. CARA [26] captured the users' dynamic preferences by leveraging both sequences of feedback and contextual information. It is equipped with two additional gates, which are the contextual attention gate that controls the impact of the ordinary context on the users' contextual preferences and a time-geo-based gate that controls the impact of the hidden state from the previous check-in based on the transition context. GT-SEER [49] learned user preferences via a pairwise ranking model under the sequential constraint and incorporated temporal influence and geographical influence to enhance the POI recommendation system. Zhu *et al.* [50] proposed a time-LSTM model which equips LSTM with time gates to model next item recommendation based on the assumption that the user's short-term and long-term interests are both important in reaching the best POI recommendation [20]. Furthermore, ST-RNN [24] modeled temporal and spatial contexts for the next POI recommendation by dividing check-ins into a time and distance window. STF-RNN [2] and SERM* [11] similarly modeled the embedding of input features, including time, location, and user in an RNN unified framework.

Generally speaking, directly applying traditional RNN-based models to capture the user's dynamic preferences is intuitive, but not efficient since it cannot integrate the contextual information correlated with the check-ins, such as time and geographical intervals. In addition, RNN, in practice, cannot properly capture the irregularity of short- and long-term users' interests. Even though there are recently proposed solutions to cope with the above challenges, the existing solutions are mainly based on including additional context embedding features or providing additional gates to handle the transition of contexts.

In this article, the proposed DeNavi model contrasts significantly with the previous literature in several aspects. Most prior works exploit a latent probabilistic generative model which utilizes the latent variables to calculate the prediction results. In order to cover many aspects, more information is necessary for the latent variables, such as user, POI, region, topic region, general public preference, and rating, to name a few. On the contrary, the proposed method, which is based on the RNN (LSTM and GRU), only needs the check-in history, that is, user sequences, to learn the characteristic of user behaviors. Differing from traditional RNNs, DeNavi decomposes the long-term memory cell into short- and long-term effects for capturing both long-term and short-term interests from the lengthy POI sequences with the gate units. With this concept, we could properly model temporal and spatial intervals between consecutive check-ins as a key factor to vary the short-term interests. The short-term effect is adjusted by the discounted weight before entering the sequential learning

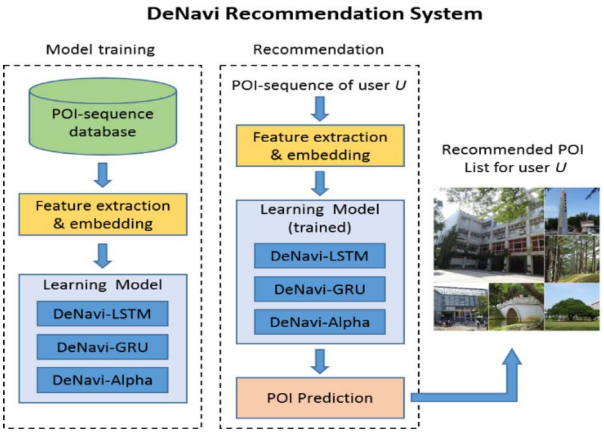


Fig. 2. Architecture of DeNavi.

model. Besides, DeNavi provides a feasible way to dynamically learn the decay values by integrating the EWMA into the model learning process.

III. Preliminary

Let $U = \{u_1, u_2, \ldots, u_n\}$ be a set of users and $L = \{X_1, X_2, \ldots, X_m\}$ be a set of locations (or POIs). A check-in record is a pair (X, t) where $X \in L$ and t is the timestamp of the check-in. Note that each POI is associated with its coordinate {latitude, longitude}. For a user $u \in U$, the trajectory check-in sequence $S_u = \langle (X_1, t_1), (X_2, t_2), \ldots \rangle$ represents the check-in history of u. The elapsed time and distance between (X_i, t_i) and (X_{i+1}, t_{i+1}) are denoted as Δt and Δd, respectively.

Problem Definition: For a user $u \in U$, given the trajectory $S_u = \langle (X_1, t_1), (X_2, t_2), \ldots, (X_{i-1}, t_{i-1}) \rangle$ and his current check-in record (X_i, t_i), the POI recommendation is to predict possible top-k locations that user u will visit at the next time step. We give the top-k locations to the user by ranking the prediction score of each $X_{i+1} \in L$ at timestamp t_{i+1}.

IV. Proposed Recommendation System: DeNavi

Fig. 2 presents the architecture of DeNavi which consists of three major components: 1) feature extracting and embedding; 2) learning model and training; and 3) prediction module.

A. Feature Extraction and Embedding

For preprocessing, the trajectory is transformed into a unique latent vector which has a lower dimension and can better capture the precise semantic spatiotemporal relationship. Due to the problem of sparsity in POI data, many research works now prefer to use embedding techniques instead of one-hot-encoding to model consecutive changes among POIs. One of the most popular embedding techniques is word2vec [12], [30], [34], which allows the user to minimize the computation overhead by decreasing data dimensionality to an acceptable size. Therefore, in this article, we extend the idea of the word2vec technique to model the successive transitions among POIs, since the dense vector representation is more suitable for representing numerous POI inputs than one-hot-encoding. DeNavi first provides a technique to encode the POIs based on historical check-in records. As a result, each POI is mapped to a unique latent vector which can represent the relation among POIs in longitudinal check-ins. The motivation behind the proposed approach is to represent a POI by a vector based on its context. This means that POIs appearing in similar contexts will be similarly embedded to vectors. Hence, for training the embedding model, we utilize all users' POI sequences which contain the contextual relationships of consecutively visited POIs. This embedding process can transform each POI into a latent vector which retains the sequential relations among POIs.

B. Learning Model and Training

Latent vectors of the trajectory are processed and trained by the recurrent model for the learning module in DeNavi. In addition, differences in time and distance between successive trajectory check-ins are calculated and used as the spatiotemporal contexts in our model. The elapsed distance is calculated by Vincenty's formula to calculate the distance between two points on the surface of a spheroid, developed in [19]. After that, these two contexts are passed to either the decay function or EWMA to dynamically weight the importance of each context based on the unique characteristics of the processed data.

In this article, three models, DeNavi-LSTM, DeNavi-GRU, and DeNavi-Alpha, were developed to capture the complicated sequential information or long-range dependencies contained in the check-in trajectory. The recurrent layer takes the sequence of the POI latent vector embedded by the previous module together with the spatiotemporal intervals as input. Then, it outputs the hidden state step by step. These hidden states are called the current mobility status. During this processing, the memory cell will capture the user's long-term interest by memorizing not only the order of the user's historical visited POIs but also the time and distance interval information between neighboring POIs.

Modeling distance intervals can help capture the user's general spatial interest while modeling time intervals help capture the user's periodical visiting behavior. Normally, a more recently visited POI with a shorter interval distance or time should have a larger influence on choosing the next POI. In each time step, we try to adjust the model parameters to reduce the error cost, according to our objective function. We detail three learning models as follows.

1) DeNavi-LSTM: We utilize the spatiotemporal information, including the time and distance intervals to model the user's short-term interest and long-term interest simultaneously. The idea of DeNavi-LSTM is extended from the LSTM model, which consists of one cell state and three controlled gates to keep and update the cell memory. Intuitively, a POI visited a long time ago and a long distance away has little influence on the next POI, and vice versa. DeNavi-LSTM is designed to handle dynamic check-in interval irregularities by incorporating the elapsed spatiotemporal information into the standard LSTM architecture.

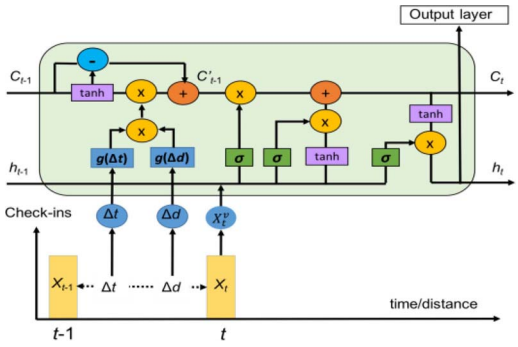


Fig. 3. Concept of the DeNavi-LSTM learning model.

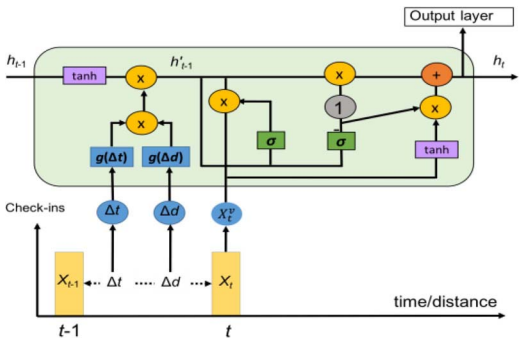


Fig. 4. Concept of the DeNavi-GRU learning model.

The concept of DeNavi-LSTM is given as Fig. 3. When using DeNavi-LSTM for the next POI recommendation, X_t^v represents the latent vector of the user's last visited POI, which can be exploited to learn the user's short-term interest. The memory cell C_{t-1} contains information about the user's historical visited POIs, which reflects the user's long-term interest. h_{t-1} stores the previous output result. However, how much the short-term interest determines where to go next heavily depends on the time interval and the geographical distance between the last POI and the current POI.

The objective functions of DeNavi-LSTM are listed as follows:

$$f_t = \sigma\left(W_f X_t^v + U_f h_{t-1} + b_f\right) \tag{1}$$
$$i_t = \sigma\left(W_i X_t^v + U_i h_{t-1} + b_i\right) \tag{2}$$
$$o_t = \sigma\left(W_o X_t^v + U_o h_{t-1} + b_o\right) \tag{3}$$
$$C_{t-1}^s = \tanh(W_s C_{t-1} + b_s) \tag{4}$$
$$C_{t-1}^l = C_{t-1} - C_{t-1}^s \tag{5}$$
$$C_{t-1}^{s'} = C_{t-1}^s \times (g(\Delta t) \circ g(\Delta d)) \tag{6}$$
$$C'_{t-1} = C_{t-1}^l + C_{t-1}^{s'} \tag{7}$$
$$Z_t = \tanh\left(W_Z X_t^v + U_Z h_{t-1} + b_Z\right) \tag{8}$$
$$C_t = f_t \circ C'_{t-1} + i_t \circ Z_t \tag{9}$$
$$h_t = o_t \circ \tanh(C_t). \tag{10}$$

The inputs are X_t^v: current location latent vector, C_{t-1}: memory cell of POI history, and h_{t-1}: previous output result. We first derive the forget gate f_t, input gate i_t, and output gate o_t, deciding how much information will be forgotten, input, and output by (1)–(3), respectively. s represents a sigmoid function to map the values between 0 and 1, where 0 represents completely ignoring the content and 1 represents completely keeping this content. W and U are the learning weights matrices, and b is the bias vector of each gate. Note that the operation $\circ$ is the Hadamard elementwise product.

Then, the memory cell is decomposed into short- and long-term effects by (4) and (5), and the short-term effect is decayed by (6). We use an interval-aware weight utility function of the elapsed time $g(\Delta t)$ and distance $g(\Delta d)$, which converts the time and distance lapse into an appropriate weight. Δt and Δd are the time and distance differences between X_{t-1} and X_t . The function $g(x) = 1/\log(e + x)$ is a heuristic decaying function such that the larger the value of x (Δt or Δd), the less the effect of the memory. The subspace decomposition is a main component of the DeNavi architecture. It decomposes the memory of the previous time step into two counterparts: 1) short-term effects and 2) long-term effects; therefore, we still keep all of the long-term effects for prediction. However, the short-term memory should be calibrated properly based on the amount of time and distance spans between the records at time step $t-1$ and t. For instance, if the elapsed gap is large, it infers that there is no updated check-in recorded for a period of time. Hence, the short-term content should not play an important role in the prediction on the current output.

The subspace decomposition of the long-term effect is merged with the discounted short-term effect to compose the adjusted previous memory as C'_{t-1} in (7). Z_t is learning from the input X_t and previous result h_{t-1} in (8). Finally, in (9) and (10), we derive the new memory cell C_t and output h_t. Note that with the data-driven concept, the parameters of the network learning (i.e., W, U, and b) are all adjusted concurrently with the rest of the network parameters by backpropagation.

2) DeNavi-GRU: DeNavi-GRU is a lightweight version of DeNavi-LSTM, which also utilizes gating information in different ways to overcome the vanishing gradient problem. The main difference between DeNavi-GRU and DeNavi-LSTM is that the unit controls the flow of information and exposes the full hidden content without using the memory unit. Although there is no memory cell in the GRU model, we also implement subspace decomposition of history from previous results. The concept of DeNavi-GRU is shown in Fig. 4. The objective function is as follows:

$$h_{t-1}^p = \tanh\left(U_p h_{t-1} + b_p\right) \tag{11}$$
$$h'_{t-1} = h_{t-1}^p \times (g(\Delta t) \circ g(\Delta d)) \tag{12}$$
$$z_t = \sigma\left(W_z X_t^v + U_z h'_{t-1} + b_z\right) \tag{13}$$
$$r_t = \sigma\left(W_r X_t^v + U_r h'_{t-1} + b_r\right) \tag{14}$$
$$\tilde{h}_t = \tanh\left(W_h X_t^v + U_h\left[r_t \circ h'_{t-1}\right] + b_h\right) \tag{15}$$
$$h_t = (1 - z_t) \circ h'_{t-1} + z_t \circ \tilde{h}_t. \tag{16}$$

The inputs are X_t^v: current location latent vector and h_{t-1}: previous output result. Since GRU does not have a memory cell, we first apply the tanh function to extract the potential memory h_{t-1}^p from the previous result as it carries concise information about the history sequence as shown in (11). Then, potential memory is discounted by the decay function of time

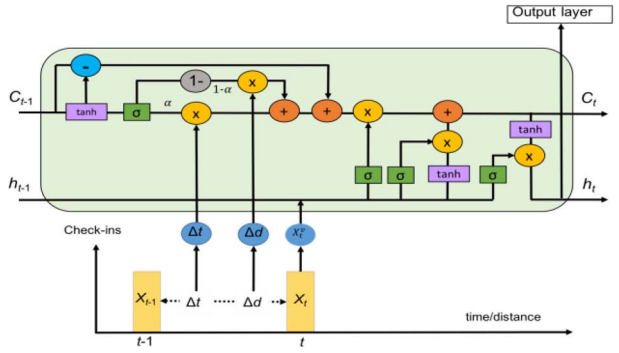


Fig. 5. Concept of the DeNavi-Alpha learning model.

and distance differences as in (12). Note that the decomposition parameters, that is, U_p and b_p, are updated simultaneously by backpropagation with the standard network.

In contrast to DeNavi-LSTM, DeNavi-GRU only uses the update gate and reset gate to decide what information should be passed to the output. In (13), the update gate z_t learns to help the model to decide how large a portion of the past content needs to be included for the future. This helps the DeNavi-GRU model to determine the important information from the past while reducing the chance of the vanishing gradient problem. The reset gate r_t learns to decide how much the earlier information should be eliminated in (14). In (15), $\tilde{h}_t$ learns the current memory content which is influenced by the reset gate r_t to determine what relevant information to remove and to store from the previous time steps. In the last step, in (16), the final result in the current time step h_t is calculated to determine which information should be taken from the current memory content $\tilde{h}_t$ and from the previous steps h_{t-1}. The same as the traditional GRU model, generally, the performance of DeNavi-GRU compared to DeNavi-LSTM in some kind of applications is computationally more efficient due to a less complex structure.

3) DeNavi-Alpha: Since each LBSN dataset captures different users' mobility preferences, using a decay function to weight the impact of contexts for prediction is effective. However, when facing multiple contexts as the input, how to tune the significance of each context is a critical issue. The motivation of DeNavi is based on the assumption that the longer the elapsed time between consecutive check-ins, the less impact on the next prediction, and vice versa. However, in some applications, this assumption is not always held for both time and distance contexts for POI recommendation. Time and distance contexts do not always have a direct variation with each other. For example, Mary may decide to visit a new restaurant R_2 near the restaurant R_1, where she had dinner one month ago. In this case, although the time interval from the previous check-in is long (i.e., more than 24 h), she still decides to visit a new restaurant R_2 since it is very close (i.e., less than 1 km) to restaurant R_1. Another similar case may be described as follows. After dinner, Mary may decide not to visit a popular pub near restaurant R_2, although the distance is very close. Because the traffic around the pub is terrible, it takes a very long time for her to obtain there. In addition,

Algorithm 1 Model Training

Input: $S = \{S_{ull}, S_{u2}, \ldots, S_{un}\}$: A set of users' check-in POI-sequences;
Z: batch, size;
Max_E: Maximum number of epoch.
Output: θ_2: The well-traiued learning model

```
For each S_ui ∈ S do
  Sort each POI in S_ui in chronological time order;
  For each X_j ∈ S_ui do
    Calculate the time difference Δt_j between X_j and X_{j+1};
    Calculate the distance difference Δd_j between X_j and X_{j+i};
    Embed X_j to latent vector X_j^v from θ_1 learned by Eq. (1);
  End
  Transform S_ui to S_ui^v=⟨(X_1^v, Δt_1, Δd_1), (X_2^v, Δt_2, Δd_2), ...⟩;
  Insert S_ui^v into training set T;
End
Calculate batch index m = |T|/Z;
Randomly select training instances from T to construct BS = {Batch_1, Batch_2, ..., Batch_m};
Initialize model θ_2 with random parameter setting;
While (epoch ≤ Max_E)
  For each Botch_i ∈ BS do
    Update θ_2 based on minimizing the prediction error derived from Eq. (20);
  End
End
Output θ_2;
```

Mary may decide to visit a distant place where she can quickly drive, leading to a short time interval between successive POIs. For tackling the above issues, DeNavi-Alpha appropriately combines each context with the exponential moving weight average (EWMA) technique to dynamically learn the impact of different contexts. We could adjust the weight of different contexts while the model is learning. For effectively capturing unique mobility patterns from datasets, DeNavi-Alpha extends from the DeNavi-LSTM model by properly utilizing contexts in POI prediction. The concept of DeNavi-Alpha is given in Fig. 5.

The objective functions are similar to DeNavi-LSTM [i.e., (1)–(10)], but replaces (6) with

$$\alpha = \sigma\left(W_\alpha C^s_{t-1} + b_\alpha\right) \tag{17}$$

$$C^{s'}_{t-1} = C^s_{t-1} \times (\alpha \times g(\Delta t) + (1-\alpha) g(\Delta d)). \tag{18}$$

The same as DeNavi-LSTM, in (4) and (5), we decompose the memory C_{t-1} of the previous time step into two counterparts: 1) short-term effects C^s_{t-1} and 2) long-term effects C^l_{t-1}. Then, the context learning weight is trained as represented in (17). This learning weight is used to discount the short-term effect C^s_{t-1} as shown in (18). After that, the complementary subspace of the long-term effect is combined with the discounted short-term effect to compose the adjusted previous memory as C'_{t-1} in (7). Finally, Z_t is learned from the input X_t and the previous result h_{t-1} in (8). Finally, in (9) and (10), we derive the new memory cell C_t and output h_t.

4) Training Algorithm: Algorithm 1 outlines the training process of the DeNavi system. First, we construct the input of the proposed learning model for training. For user u's POI-sequence S_u in the training dataset, we calculate the time and

distance differences (Δt and Δd) between the current location X_i and the previous location X_{i-1}. Then, we transform each X_i to an embedded latent vector X_i^v, and reconstruct $S_u^v = \langle (X_1^v, \Delta t_1, \Delta d_1), (X_2^v, \Delta t_2, \Delta d_2), \dots \rangle$ and insert S_u^v into the training instance set T. Then, we leverage the minibatch learning method to train the model on each user's existing history until convergence. The model output is a latent embedding vector denoted as h_i, that is, (10) and (16), in each time step. h_i could also be considered as the prediction latent vector for the next timestamp. Hence, the error function is calculated by root-mean-square error (RMSE) between predicted the embedding vector h_i and the actual embedding vector X_{i+1}^v

$$E = \sum_{S_u \in Z} \sum_{i=1}^{|S_u|} \left(h_i - X_{i+1}^v\right)^2. \tag{19}$$

During the training process, in each epoch, all of the user's sequences are randomly selected from T for each minibatch Z (size is specified by user). Each training batch consists of various lengths of the user's visited POIs for different time steps (however, the minimum length of S_u is set to 80) as described in (19). We take a gradient step to minimize the error based on h_i and X_{i+1}^v. We use Adam [21], a variant of gradient descent, to optimize the parameters in DeNavi. The Adam optimization adapts the learning rate for each parameter by performing a little update for frequent parameters and a heavy update for infrequent parameters. For the most optimized settings of DeNavi, the cell size and the hidden state size are set as 312 in our experiments. Also, the number of epochs is set as 15 and the batch size is set as 64 for our proposed model.

C. Prediction Module of DeNavi

The prediction model is the final component that combines the context from different modules to complete the prediction task. As shown in Fig. 2, when recommending POIs to user u, we first input u's trajectory (i.e., the POI history) $S_u = \langle (X_1, t_1), (X_2, t_2), \dots, (X_i, t_i) \rangle$ into the feature extraction and embedding model to transform each POI into the corresponding latent vector. Then, the DeNavi system will feed the latent vector sequence, time differences, and distance differences into the well-trained learning model (DeNavi-LSTM, DeNavi-$_{\text{GRU}}$, or DeNavi-Alpha) to derive a prediction vector X_{i+1}^v. Finally, we compare all POI latent vectors to recommend the top-N similar locations to user u. The similarity function is defined in

$$S\left(X_{i+1}^v, Y^v\right) = \frac{X_{i+1}^v \cdot Y^v}{\left\|X_{i+1}^v\right\| \|Y^v\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^m x_i^2} \sqrt{\sum_{i=1}^m y_i^2}} \tag{20}$$

where prediction vector $X_{i+1}^v = \langle x_1, \dots, x_m \rangle$ and the embedding vectors of a location $Y^v = \langle y_1, \dots, y_m \rangle$ are in the same latent space. This allows us to effectively compute the semantic similarity between two POIs, and to find the POIs most similar to an output embedding. Consequently, the most top-N similar POIs are returned as the final prediction results.

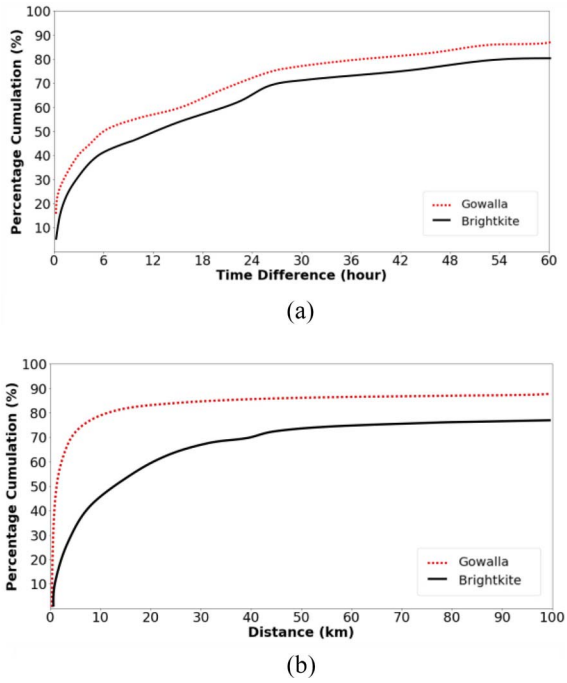


Fig. 6. Cumulative distribution of time and distance differences in the Gowalla and BrightKite datasets. (a) CDF of the time differences between two POIs. (b) CDF of the distance differences between two POIs.

V. Performance Evaluation

In this section, we evaluate the effectiveness and robustness of the proposed methods compared with the state-of-the-art models. For designing the experiments, we discuss the following research questions.

1) *Q1:* Can we strengthen the efficiency of the traditional recurrent architecture by utilizing the check-in ordinary and spatiotemporal transition contexts correlated with the sequence of check-ins?
2) *Q2:* Can we enhance the performance of the traditional recurrent model by incorporating the transition of time and distance between consecutive check-ins into the subspace decomposition of the memory cell in LSTM and GRU?
3) *Q3:* Can DeNavi-Alpha, which leverages the dynamic learning weight α for each context, such as the temporal differences and geographical differences between consecutive check-ins, outperform the traditional decay function in DeNavi-LSTM?

In the following sections, we demonstrate the performance of the developed DeNavi recommendation system.

A. Experimental Setup

We use two public LBSN datasets that contain interactions between users and POIs and are called Gowalla and BrightKite [7]. The check-in data were collected from February 2009 to October 2010 and April 2008 to October 2010, and the total number of check-ins is 6.4 million with 107 092 users and 4.5 million with 58 228 users for Gowalla and BrightKite, respectively. Every record in the data consists of user ID, timestamp, POI ID, POI latitude,

TABLE I
GOWALLA AND BRIGHTKITE [7] DATASETS

dataset	no. of users	no. of POIs	no. of check-ins
Gowalla	51,985	101,865	3,301,557
BrightKite	18,754	46,745	3,071,500

and POI longitude. Fig. 6(a) shows the cumulative distribution functions of the time differences between two successive check-in records in the Gowalla and BrightKite datasets. In the Gowalla dataset, more than 50% of successive check-ins happened within 6 h and over 30% of the data happened more than 24 h. In the BrightKite dataset, about 35% of successive check-ins happened within 6 h and over 50% of the data happened more than 24 h. Fig. 6(b) shows the distance differences distribution between check-in records within 6 h. From the result, around 90% and 45% of successive check-ins happened within 15 km in the Gowalla and BrightKite datasets, respectively. It is obvious that the BrightKite dataset is sparser in the temporal and spatial domains than the Gowalla dataset.

Since the experimental datasets are sparse, the length of records for one user ranges from 1 up to 2000. The noise may severely affect the performance of recommendation; thus, we filter out the POIs with less than five records and the users with less than ten check-ins to include only active users. After preprocessing, we have the dataset description as shown in Table I. Each user's POI sequence is treated as an instance for training or testing. We take the 80% dataset as the training set and the remaining 20% for the test set. The experiments are repeated at least $K = 3$ times with different random seeds, and the results reported in the following section are averaged from K iterations.

To evaluate the effectiveness of our proposed method, we use three well-known metrics as follows: 1) accuracy at N; 2) precision at N; and 3) recall at N (denoted by ACC@N, P@N, and R@N, respectively), where N is the number of recommendation results. In this article, we choose $N = 1, 5, 10$, and 20 to illustrate the different results of metrics at N. Note that for an instance in the testing set, ACC@N is 1 if the visited POI appears in the set of top-N recommendation POIs, and 0 otherwise. The overall ACC@N is calculated as the average value of all testing instances. Suppose, for user u_i, $Test\_visit_{ui}$ denotes the set of corresponding visited POIs in the testing data, and $Recom\_set_{ui}$ denotes the set of recommended POIs. The definitions of P@N, R@N, and F@N are described in the following equations:

$$\mathrm{P@N} = \frac{1}{|U|}\sum_{u_i \in U}\frac{\left|Recom\_set_{u_i} \bigcap Test\_visit_{u_i}\right|}{\left|Recom\_set_{u_i}\right|} \tag{21}$$

$$\mathrm{R@N} = \frac{1}{|U|}\sum_{u_i \in U}\frac{\left|Recom\_set_{u_i} \bigcap Test\_visit_{u_i}\right|}{\left|Test\_visit_{u_i}\right|} \tag{22}$$

$$\mathrm{F@N} = 2 * \left(\frac{\mathrm{P@N} * \mathrm{R@N}}{\mathrm{P@N} + \mathrm{R@N}}\right). \tag{23}$$

We demonstrate the performance of the proposed De-Navi models compared with several prior methods, including the state-of-the-art RNN architectures and other factorization-based approaches. All models are implemented on Tensorflow and Theano, including the following.

1) *MF:* A traditional matrix factorization method [3] that aims to accurately predict the users' check-ins by learning the latent factors via regressing over the existing user-item ratings.
2) *FPMC:* Rendle *et al.* [32] embeded users' preferences and built a personalized Markov chain into the model for next basket item recommendation.
3) *RNN:* A traditional recurrent architecture [15], [46] that only considers the sequence of POIs in its hidden unit while ignoring additional contextual information associated with the longitudinal check-ins.
4) *LSTM:* The variant RNN model contains a memory cell and three multiplicative gates to allow short- and long-term dependency learning.
5) *GRU:* The variant RNN model is equipped with two gates to control the information flow.
6) *ST-RNN:* The method [24] replaces the single transition matrix in RNN to model the spatiotemporal contexts by including time-specific and distance-specific transition matrices during model learning.
7) *STF-RNN:* The method [2] includes space in terms of location centroid ID and time in terms of time bins into the model learning as input features. The space and time are mapped to the embedded semantic vector via a lookup table layer before being incorporated with the recurrent structure to discover long-term dependencies.
8) *SERM*:* The method [11] jointly learns the embedding of multiple semantic factors, including user, location, time, and the transition parameters of RNN simultaneously.
9) *Time-LSTM:* The method [50] equips LSTM with time gates to model the time intervals. These time gates help to update the long- and short-term interests in the later recommendation.

We evaluate our DeNavi models on two mobility datasets to access the performance. In general, the evaluation results of the two mobility datasets demonstrate the superiority and generalization of the DeNavi recommendation system. We rank the candidate locations by the probabilities generated from the embedding model, then check whether the ground-truth location appears in the top-N candidate locations. More experimental details are provided in the following sections.

B. ACC@N Performance

First, to respond to the three research questions above, we provide the result of ACC@N with $N = \{1, 5, 10, 20\}$ in Table II. Comparing the other methods, the traditional RNN-based method performs significantly better than MF and FPMC because of its powerful sequence modeling ability. We can observe that the prediction ACC@10 of RNN has an improvement of about 70% and 40% in Gowalla and BrightKite compared to traditional approaches, respectively. This indicates that human mobility contains some periodical

TABLE II
ACC@N PERFORMANCE OF DIFFERENT MODELS ON TWO DATASETS

Model	Gowalla				BrightKite			
	ACC@1	ACC@5	ACC@10	ACC@20	ACC@1	ACC@5	ACC@10	ACC@20
MF	0.0103	0.0257	0.0432	0.0699	0.1031	0.1763	0.2305	0.2851
FPMC	0.0123	0.0253	0.0528	0.0701	0.1352	0.2246	0.2752	0.3137
RNN	0.0297	0.0483	0.0752	0.1154	0.2128	0.3542	0.4010	0.4321
GRU	0.0386	0.0814	0.1359	0.1540	0.3103	0.4007	0.4453	0.4857
LSTM	0.0571	0.0893	0.1306	0.1553	0.3527	0.4134	0.4504	0.4803
ST-RNN	0.0603	0.1142	0.1614	0.1897	0.3672	0.4321	0.4798	0.5023
STF-RNN	0.0594	0.0813	0.1235	0.1403	0.2931	0.3849	0.4124	0.4652
SERM*	0.0611	0.1163	0.1523	0.1801	0.3778	0.4212	0.4670	0.5021
Time-LSTM	0.0701	0.1227	0.1789	0.1926	0.4233	0.4521	0.4976	0.5481
DeNavi-LSTM	0.0732	0.1397	0.1801	0.2127	0.4342	0.4778	0.5300	0.5679
DeNavi-GRU	0.0518	0.0912	0.1586	0.1728	0.3320	0.4296	0.4593	0.4993
DeNavi-Alpha	**0.0764**	**0.1419**	**0.2097**	**0.2460**	**0.4412**	**0.4813**	**0.5388**	**0.5831**

regularities, which helps to improve prediction accuracy. Since the complex sequential transition from the current trajectory can be captured by recurrent parts in RNN, our model can also apply this concept.

Second, it is not surprising that there is a performance improvement of LSTM and GRU over RNN since LSTM and GRU have more capability of modeling complex and longer sequences. Also, we notice that there is a slight improvement of ST-RNN, STF-RNN, and SERM* compared to the standard LSTM and GRU methods. From this we can infer that time and distance intervals are crucial factors for performance improvement. Third, time-LSTM, which inserts time gates into the recurrent units, performs better than other state-of-the-art baselines. The improvement of time-LSTM shows that it can model users' short- and long-term interests with temporal and spatial contexts well.

Then, we observe that there is a gradual improvement of DeNavi-GRU over GRU with an average of about 2%–6%. This indicates that modeling temporal and spatial contexts in DeNavi-GRU can improve the performance when compared with GRU. Finally, we find that the proposed DeNavi-LSTM architecture, which leverages the irregularity of time and distance of the transition context, is more effective than other baselines in terms of capturing the movement effects between consecutive check-ins. We can see that DeNavi-LSTM has a marginal improvement of 16%–17% as compared to LSTM in general.

In particular, the proposed DeNavi-Alpha enables the memory discount by employing the elapsed time and elapsed distance between sequential elements to weight the short-term memory content. With this concept, unlike previous works, we still gain performance enhancement as we do not need to add additional current inputs or add additional gates that can have an effect on the current output prediction. Since the nature of the LBSN data is sparse in terms of time and distance between check-ins, the same contexts (elapsed time and distance) may have different impacts on different datasets. Hence, we integrate the dynamic learning context weight for time and distance contexts instead of using the static weight decay function. As the results in Table II show, we can find that with dynamic learning context weight by DeNavi-Alpha, we can further improve the accuracy of the results as compared to the traditional LSTM by up to 30% and 18% in Gowalla and BrightKite, respectively. This indicates that the proposed DeNavi-Alpha architecture is able to leverage the dynamic learning context weight of time and distance transition contexts to enhance accuracy performance.

C. Precision@N, Recall@N, and F-measure@N Performance

In this section, we evaluate the ranking performance of the proposed models compared with the traditional recurrent architecture. As stated in an earlier section, after data preprocessing, the minimum check-in sequence length is 80. Therefore, we manually fix the first 40 check-ins in our test dataset as the input for the prediction model. We employ precision (P@N), recall (R@N), and F-measure (F@N) to evaluate the ranking tasks. The evaluation score for our experiment is computed according to whether the next selected location appears in the ranked list as shown in (21)–(23). Since the model comparison results show a similar trend with different values of N (1, 5, 10, and 20), we discuss the representative results of P@N, R@N, and F@N with $N = 10$ and 20 in our experiments as shown in Figs. 7 and 8.

We discover that the proposed DeNavi-Alpha model achieves better performance than all the baselines. When compared with the traditional RNN model, there is more than 100% improvement in both precision and recall. Specifically, DeNavi has an improvement on F-measure of 24% and 14% compared with ST-RNN and time-LSTM on BrightKite, and of 30% and 12% on the Gowalla dataset. In addition, when considering the F-measure metric, we also found that on average,

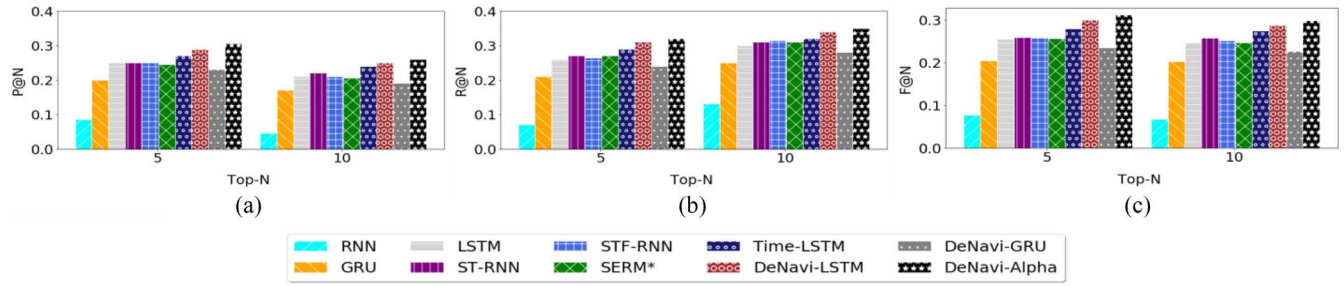


Fig. 7. Comparison of precision, recall, and F-measure on the BrightKite dataset.

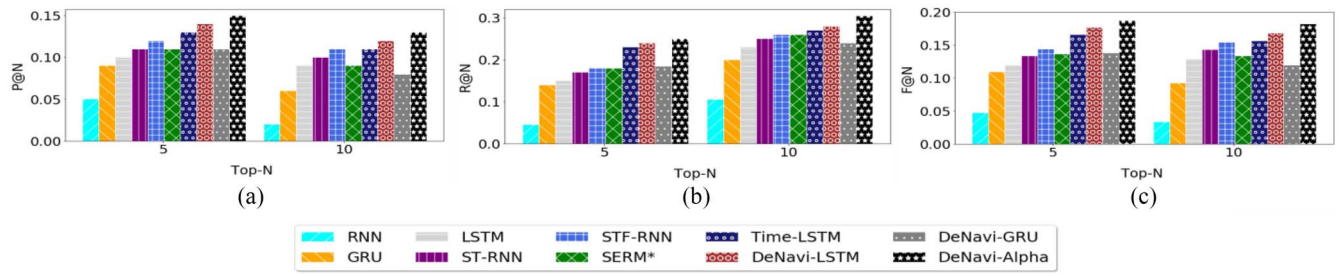


Fig. 8. Comparison of precision, recall, and F-measure on the Gowalla dataset.

our proposed model outperforms other state-of-the-art models. This result verifies the effectiveness of our spatiotemporal sequence modeling that incorporates temporal and geographical transition influence into the decomposition of the short- and long-term memory cells of the traditional LSTM model.

In addition, we observe that the DeNavi-Alpha model performs better on BrightKite than Gowalla for precision and recall. The possible reason lies in the fact that the number of locations in Gowalla is greater than in BrightKite. As shown in Table I, there are double the number of POIs in Gowalla than in BrightKite. According to the metrics in (21)–(23), the result seems reasonable. Also, there is low density in the Gowalla datasets, which normally results in relatively low precision and recall values [43], [47].

D. Effectiveness of Time and Distance Transition Contexts

As we can see from the above experimental results, DeNavi-Alpha outperforms all existing recurrent models. In this section, we analyze how much improvement in incorporating time and distance differences there is between successive check-ins in DeNavi-LSTM and DeNavi-GRU. In DeNavi-Alpha, there are two decompositions of memory cells representing long-term and short-term effects. We capture user long-term effects by memorizing not only the order of the user's historical visited POIs but also the time and distance interval information between neighboring POIs. Meanwhile, we dynamically learn to give the proper weight to each context in short-term effects. Normally, the short-term effect determines where to go next, and heavily depends on the time interval and the geographical distance between the last and the next POI. Intuitively, a POI visited a long time ago and a long distance away has little influence on the next POI, and vice versa.

However, different users may have different preferences regarding these contexts. For example, user U usually visits POI P five days a week, although the distance from his house to POI P is very close, it will take a very long time to obtain there since the traffic is terrible. In this case, the transition of distance may have more impact on the next move. This motivates this article to let the recurrent model learn the dynamic context weights from the dataset since modeling distance intervals can help capture users' general spatial interest while modeling time intervals helps capture their periodical visiting behavior.

In this section, we first investigate the effectiveness of integrating either time transition context or distance transition context. In order to explore the improvement of one context per experiment, we set either time elapsed or distance elapsed to 1, which means that we disable the highlighted context on discounted short-term memory. As shown in Fig. 9(a) and (b), we can observe that including the transition of time (time-DeNavi-LSTM) and distance (distance-DeNavi-LSTM) results in significant improvement in ACC@10 on average by 10%–30% on the two datasets as compared with LSTM. Moreover, we notice that on both datasets, the impact of elapsed time moderately influences the improvement of prediction more than the impact of elapsed distance. In addition, we can observe that DeNavi-Alpha outperforms the traditional LSTM, time-DeNavi-LSTM, and distance-DeNavi-LSTM, which means that both spatiotemporal contexts are critical for improving the recommendation performances.

According to [37], we could explain the difference in performance improvement on the two datasets in more detail. In the Gowalla dataset, the consecutive transition plays a crucial role in the prediction since there is a trip recommender system that guides users to follow the trip advice. In the BrightKite dataset, either elapsed time or elapsed distance between two consecutive trajectories are generally longer than those in the Gowalla dataset. The reason behind this

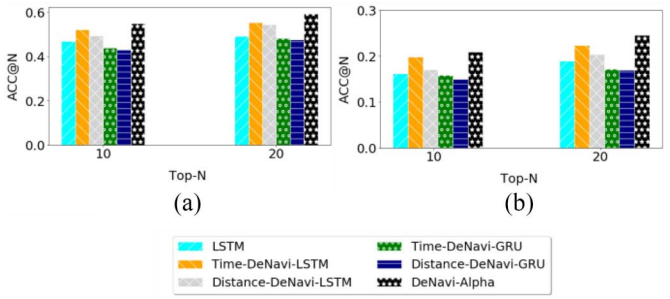


Fig. 9. Evaluation of spatiotemporal contexts in the DeNavi-Alpha learning model on the BrightKite and Gowalla datasets.

circumstance is that the BrightKite system lets a user free to check-in at any POI, even if it is impossible for the user to visit the POI within a short time period in reality. This mechanism makes user preference the key factor in the BrightKite dataset since a user normally has regular visits in a set of spatial locations. Also, user movement is usually influenced by the relationships of the social network (i.e., friends, family members, and colleagues, to name a few), especially for visiting long-distance POIs. The user preference in this context means the general preference or long-term behaviors, such as certain types of locations. For example, one user is likely to check-in at the restaurants near his workplace and his friends' area which is located at a distance from his home. According to [7], the mobility in BrightKite is more distant than in Gowalla due to the probability of visiting friends. In other words, one reason for the longer Δt and Δd in BrightKite is the influence of friends on an individual's mobility. In addition, similar to the results of [7], we observe that 52% of all check-ins in BrightKite (30% in Gowalla) have been previously visited by the same user. This means that if a user checks into a place for the first time, there is more than a 50% chance that he might return and check-in again. This is the main reason why the performance of our proposed models in BrightKite is better than in the Gowalla dataset.

E. Scalability Discussion

In addition, we discuss the scalability of the proposed algorithms. As shown in Fig. 10(a) and (b), with the increase in the size of the training set, all accuracy curves as a whole tend to rise. The change is more obvious when the training data size reaches 70%. From the observation, for accuracy, we can notice that the ST-RNN and STF-RNN models converge around 70% training size, while our proposed models require more data of up to 80% training size to learn the complexity of the POI sequences.

We also discuss the scalability for the precision and recall rates of the proposed algorithms. As shown in Fig. 11(c) and (d), all models could converge at around 80% training data; however, the proposed DeNavi-Alpha and DeNavi-LSTM have higher P@10 than other learning models. Likewise, as in Fig. 11(e) and (f), almost all models could reach a higher recall rate when using 80% training data. We can notice that although the MF and FPMC models converged

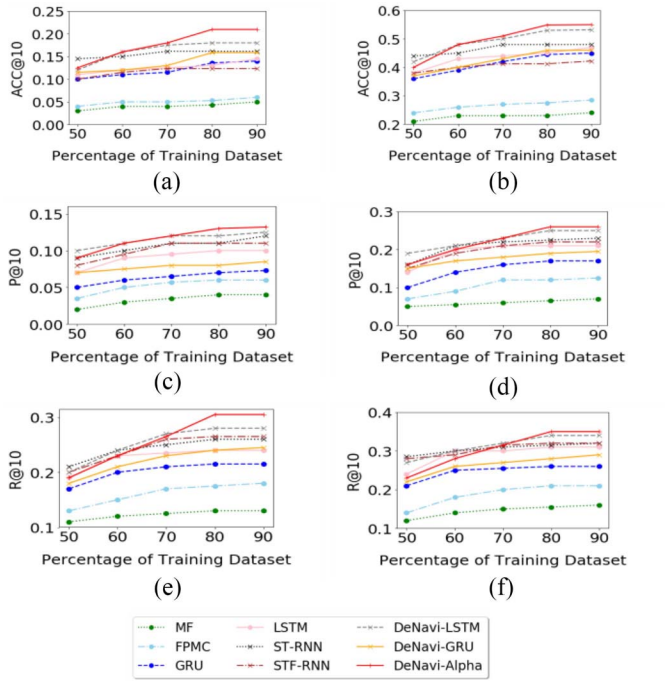


Fig. 10. Scalability on accuracy, recall, and precision of varying the training percentage on the Gowalla and BrightKite datasets. (a) Scalability of ACC@10 on Gowalla. (b) Scalability of ACC@10 on BrightKite. (c) Scalability of P@10 on Gowalla. (d) Scalability of P@10 on BrightKite. (e) Scalability of R@10 on Gowalla. (f) Scalability of R@10 on BrightKite.

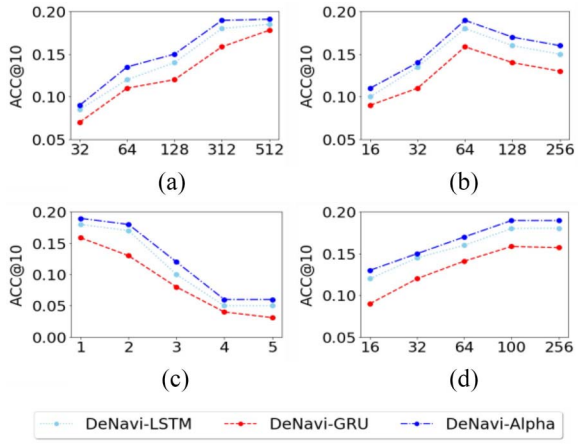


Fig. 11. Performance comparison of different parameter settings on the Gowalla dataset. (a) Different cell size. (b) Different batch size. (c) Different layer size. (d) Different embedding size.

curve is smoother, the precision and recall rates are significantly lower than those of other learning models, especially the proposed DeNavi-Alpha and DeNavi-LSTM.

F. Discussion of Parameter Settings

We also investigate the impact of the DeNavi-Alpha model parameters since it has the best performance results. Obviously, different parameter settings may lead to different performance. Hence, we perform more experiments by varying the cell sizes, batch sizes, the number of layers, and POI latent vector sizes during training. In Fig. 11(a), we show

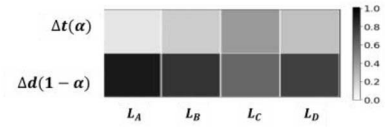
how increasing the cell size improves the performance of the proposed DeNavi-Alpha model in terms of the ACC@10 metric. Definitely, the cell size determines the model complexity. The cell with a larger size fits the data better.

We also present the impact of batch size in Fig. 11(b). We could clearly find the optimal result when the batch size is set to 64. One possible explanation why the small and large batch size (i.e., 16 and 256) have ACC@10 less than the optimal one (i.e., 64) is that a small batch size may lead to local optimum, while a big one may lead to insufficient updating of parameters in the model. As shown in Fig. 11(c), increasing DeNavi layers will add more computation burden during training rather than improving performance. In addition, we also found that stacking or adding a peephole mechanism to the DeNavi modules does not help increase model performance. Finally, in Fig. 11(d), we present the impact of the POI embedding vector size on ACC@10. The result shows that the optimal size of POI embedding is 100 since the performance starts to converge at this number as compared to other values.

We conclude our tuning model settings as follows: we set the pretrained embedding vector size as 100, while learning rate = 0.01, gradient clip = 5.00, the number of model layers = 1, batch size = 64, cell size = 312, and training percentage = 80 for the model learning process. We conclude the evaluation section by stating that the DeNavi-Alpha model significantly outperforms all the baseline methods on two different sets of mobility data in terms of prediction accuracy, precision, and recall.

Info	The last four POIs of S_u ($X_{t-3} \rightarrow X_{t-2} \rightarrow X_{t-1} \rightarrow X_t$)	GroundTruth (X_{t+1})
POI	...→ L_A (22 New Road) → L_B (15 Khartoum Road) → L_C (20 Station) → L_D (347 High Street)	Sun Pier
Δt	...→ L_A (40sec) → L_B (18sec) → L_C (15sec) → L_D (10sec)	1min
Δd	...→ L_A (1.2km) → L_B (2.6km) → L_C (4.7km) → L_D (2.2km)	900m

(a)



(b)

Fig. 12. Visualization of the time and distance context weights and recommendation of DeNavi-Alpha for userID 39 934 in the Gowalla Data. (a) Information of the last four POIs in S_u and recommendations. (b) Visualization of how DeNavi-Alpha works.

G. Interpretability Discussion

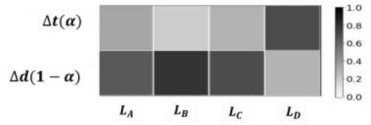
In this section, we discuss in more detail the model interpretability. For the interpretation of the proposed model, we discuss: 1) what we learn for recommendation and 2) why we do the recommendation. Without any doubt, from traditional deep learning approaches, we cannot easily explain what the model has learned internally. However, in this article, we introduce an alpha gate unit into DeNavi-Alpha to selectively learn what is the important context during interference. We use the pretrained DeNavi-Alpha to inspect various cases in the testing data. We randomly select two cases where ACC@N is 1 (i.e., correct recommendation) on DeNavi-Alpha for our inspection and print out four alpha values along the input sequence for demonstration.

1) *Case 1 (Δd Is More Significant Than Δt):* We know that the proposed DeDavi-Alpha model decomposes the long-term memory into long-term and short-term effects, and then utilize the α value to control the contribution of the decay. We take the sequence of userID 39934 in the Gowalla dataset as an example. Clearly, unlike the GPS cellular data, LSBN allows users to check-in anywhere at any time. Under this circumstance, learning only temporal intervals between POIs may not reflect the actual relationship between the visiting time. For example, as shown in Fig. 12(a), user 39 934 in Gowalla checks in at L_A (22 New Road), L_B (15 Khartoum Road), L_C (20 Station), and L_D (347 High Street) at once (i.e., $\Delta t \sim 1$ min). In this case, Δt may have less impact than Δd. As a result, the proposed DeNavi-Alpha could correctly recommend the ground truth POI (Sun Pier) by the proposed inference process as shown in Fig. 13(a). The first three predictions of DeNavi-Alpha of this user are returned as $\hat{R}_1$ (Sun Pier), $\hat{R}_2$ (Sail and Color Loft), and $\hat{R}_3$ (21 Crow Lane). The possible main reason is that these $\hat{R}_n$ have similar Δd distribution of approximately 1–5 km for this user sequence. Also, L_A, L_B, L_C, and L_D and $\widehat{R_n}$ are normally checked-in together. Since DeNavi-Alpha has the capability to capture the sequential patterns learned by the LSTM unit and Δd and Δt distributions learned by the additional alpha gate, it can provide the right recommendation under the complex trajectories. Fig. 13(b) shows how DeNavi-Alpha works during the inference process of the last four POIs ($L_A \rightarrow L_B \rightarrow L_C \rightarrow L_D$) before the prediction. Note that more dark color in the heap map means that the context is more important. We can observe that the model correctly emphasizes the context Δd and decreases the contribution of Δt in each time step.

2) *Case 2 (Δd and Δt Are Both Important):* Generally, the POI trajectories contain the irregularity of both spatial and temporal intervals. In common cases, the spatial difference and temporal difference both play significant roles for the next POI recommendation. We take the sequence of userID 1827 in the Gowalla dataset as an example. As shown in Fig. 13(a), DeNavi-Alpha could correctly recommend the first three predictions of DeNavi-Alpha of this user, returned as $\hat{R}_1$ (Architecture School), $\hat{R}_2$ (801 Kaheka St.), and $\hat{R}_3$ (500 Pohukaina).

Info	The last four POIs of S_u ($X_{i-3} \rightarrow X_{i-2} \rightarrow X_{i-1} \rightarrow X_i$)	GroundTruth(X_{i+1})
POI	...→ L_A (801 Kaheka St.)→ L_B (1223 N School St.)→ L_C (Art Bldg.)→ L_D (Music Bldg Complex)	500 Pohukaina
Δt	...→ L_A (11.30hrs)→ L_B (4hrs)→ L_C (4hrs)→ L_D (25hrs)	2hrs
Δd	...→ L_A (7km)→ L_B (5.6km)→ L_C (3km)→ L_D (650m)	5.2km

(a)



(b)

Fig. 13. Visualization of the time and distance context weights and recommendation of DeNavi-Alpha for userID 1827 in the Gowalla Data. (a) Information of the last four POIs in S_u and recommendations. (b) Visualization of how DeNavi-Alpha works.

Based on the assumption that the greater the elapsed time or distance, the smaller the effect of the recent POIs on the current decision, intuitively, POIs visited a long time ago and a long distance away have little influence on the next POI. We can observe that in Fig. 13(b), the model gives more weight to Δd during the check-in at L_A (801 Kaheka St.), L_B (1223 N School St.), and L_C (Art Bldg.), while Δt puts more emphasis on L_D (Music Bldg Complex). The reason is that the check-in of L_A, L_B, and L_C happened on the same day while L_D happened the following day. This indicates that during the same day, DeNavi-Alpha learned to capture the sequential pattern and the distribution of Δd. However, when there is a large gap of Δt, DeNavi-Alpha will pay more attention to Δt. As a result, the ground-truth POI (500 Pohukaina) is recommended correctly in the inference process.

VI. Conclusion

In this article, we investigated the challenges of POI mobility prediction from sparse and lengthy POI trajectories. We proposed a novel POI recommendation system DeNavi to predict the next move. Including the time and distance intervals between POI check-ins in the memory unit, three learning models: 1) DeNavi-LSTM; 2) DeNavi-GRU; and 3) DeNavi-Alpha were developed to enhance the performance of the standard recurrent networks. Specifically, by integrating the EWMA into the model learning process, DeNavi-Alpha enables a practical approach to dynamically weighting the spatial and temporal decay values. As a consequence, DeNavi-Alpha can capture how much each context should be emphasized in the prediction process. The detailed experiments on two real-life mobility datasets demonstrate that DeNavi significantly outperforms all the baselines in all metrics. In particular, the experimental results also show that DeNavi-Alpha performed better than the state-of-the-art methods since it is able to effectively capture meaningful contexts for mobility by dynamically integrating the weight of spatial and temporal decay values. For our future works, we will consider introducing the attention mechanism, user profile, and supplementary contexts into our model for better recommendation accuracy.

References

[1] C. C. Aggarwal, "Context-sensitive recommender systems," in *Recommender Systems*. Cham, Switzerland: Springer, 2016, pp. 255–281.

[2] A. Al-Molegi, M. Jabreel, and B. Ghaleb, "STF-RNN: Space time features-based recurrent neural network for predicting people next location," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, Athens, Greece, 2016, pp. 1–7.

[3] M. Abdi, G. Okeyo, and R. Mwangi, "Matrix factorization techniques for context-aware collaborative filtering recommender systems: A survey," *Comput. Inf. Sci.*, vol. 11, no. 2, p. 1, 2018.

[4] I. Baytas, C. Xiao, X. Zhang, F. Wang, A. Jain, and J. Zhou, "Patient subtyping via time-aware LSTM networks," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2017, pp. 65–74.

[5] H. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommend. Syst. (DLRS)*, 2016, pp. 7–10.

[6] M. Chen, Y. Liu, and X. Yu, "NLPMM: A next location predictor with Markov modeling," in *Proc. Adv. Knowl. Disc. Data Min.*, 2014, pp. 186–197.

[7] E. Cho, S. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2011, pp. 1082–1090.

[8] C. Cheng, H. Yang, I. King, and M. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," in *Proc. 26th AAAI Conf. Artif. Intell. (AAAI)*, 2012, pp. 17–23.

[9] C. Cheng, H. Yang, M. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," in *Proc. 23rd Int. Joint Conf. Artif. Intell. (IJCAI)*, 2013, pp. 2605–2611.

[10] H. Dyvik, "Exploiting structural similarities in machine translation," *Comput. Human.*, vol. 28, nos. 4–5, pp. 225–234, 1994.

[11] D. Yao, C. Zhang, J. Huang, and J. Bi, "SERM: A recurrent model for next location prediction in semantic trajectories," in *Proc. ACM Conf. Inf. Knowl. Manag. (CIKM)*, 2017, pp. 2411–2414.

[12] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2016, pp. 1555–1564.

[13] J. Feng *et al.*, "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proc. World Wide Web Conf. (WWW)*, 2018, pp. 1459–1468.

[14] Y. Fu, B. Liu, Y. Ge, Z. Yao, and H. Xiong, "User preference learning with multiple information fusion for restaurant recommendation," in *Proc. SIAM Int. Conf. Data Min.*, 2014, pp. 470–478.

[15] A. Graves, "Supervised sequence labelling," in *Studies in Computational Intelligence Supervised Sequence Labelling With Recurrent Neural Networks*. Heidelberg, Germany: Springer, 2012, pp. 5–13.

[16] S. Gambs, and M. Killijian, and M. Cortez, "Next place prediction using mobility Markov chains," in *Proc. 1st Workshop Meas. Privacy Mobility (MPM)*, vol. 3, 2012, pp. 1–6.

[17] D. Graur, R.-A. Mariş R. Potolea, M. Dînşoreanu, and C. Lemnaru, "Complex localization in the multiple instance learning context," in *New Frontiers in Mining Complex Patterns* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2018, pp. 93–106.

[18] J. He, X. Li, L. Liao, D. Song, and K. Cheung, "Inferring a personalized next point-of-interest recommendation model with latent behavior patterns," in *Proc. 13th AAAI Conf. Artif. Intell. (AAAI)*, 2016, pp. 137–143.

[19] Y. Hu, S. Ravada, R. Anderson, and B. Bamba, "Distance queries for complex spatial objects in oracle spatial," in *Proc. 22nd ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. (SIGSPATIAL)*, 2014, pp. 291–300.

[20] D. Jannach, L. Lerche, and M. Jugovac, "Adaptation and evaluation of recommendations for short-term shopping goals," in *Proc. 9th ACM Conf. Recommend. Syst. (RecSys)*, 2015, pp. 211–218.

[21] D. P. Kingma and B. Jimmy, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015.

[22] H. Li, "Learning to rank," in *Encyclopedia of Machine Learning and Data Mining*. New York, NY, USA: Springer, 2016, pp. 1–6.

[23] Y. Liu, C. Liu, B. Liu, M. Qu, and H. Xiong, "Unified point-of-interest recommendation with temporal interval assessment," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2016, pp. 1015–1024.

[24] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. 13th AAAI Conf. Artif. Intell. (AAAI)*, 2016, pp. 194–200.

[25] B. Liu, H. Xiong, S. Papadimitriou, Y. Fu, and Z. Yao, "A general geographical probabilistic factor model for point of interest recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1167–1179, Jan. 2015.

[26] J. Manotumruksa, C. Macdonald, and I. Ounis, "A contextual attention recurrent architecture for context-aware venue recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retriev. (SIGIR)*, 2018, pp. 555–564.

[27] J. Manotumruksa, C. Macdonald, and I. Ounis, "A deep recurrent collaborative filtering framework for venue recommendation," in *Proc. ACM Conf. Inf. Knowl. Manag. (CIKM)*, 2017, pp. 1429–1438.

[28] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "WhereNext: A location predictor on trajectory pattern mining," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2009, pp. 637–646.

[29] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden Markov models," in *Proc. ACM Conf. Ubiquitous Comput. (UbiComp)*, 2012, pp. 911–918.

[30] R. Mehta and K. Rana, "A review on matrix factorization techniques in recommender systems," in *Proc. 2nd Int. Conf. Commun. Syst. Comput. IT Appl. (CSCITA)*, 2017, pp. 269–274.

[31] T. Qian, B. Liu, Q. V. H. Nguyen, and H. Yin, "Spatiotemporal representation learning for translation-based POI recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, pp. 1–24, 2019.

[32] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, 2010, pp. 811–820.

[33] R. Rosenkrantz, "Information theory and statistical mechanics I (1957)," in *Papers on Probability, Statistics and Statistical Physics*, E. T. Jaynes, Eds. Dordrecht, The Netherlands: Kluwer, 1989, pp. 4–16.

[34] V. Sridhar, "Unsupervised text normalization using distributed representations of words and phrases," in *Proc. 1st Workshop Vector Space Model. Nat. Lang. Process.*, 2015, pp. 8–16.

[35] W. Wang, H. Yin, X. Du, Q. V. H. Nguyen, and X. Zhou, "TPM: A temporal personalized model for spatial item recommendation," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 6, pp. 1–25, 2018.

[36] W. Wang, H. Yin, S. Sadiq, L. Chen, M. Xie, and X. Zhou, "SPORE: A sequential personalized spatial item recommender system," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, 2016, pp. 954–965.

[37] J. Waters, *All You Need to Know About Participating in Today's Most Popular Online Communities*. Avon, MA, USA: Adams Media, 2010, pp. 202–211.

[38] Q. Wu and C. Pu, "Modeling and implementing collaborative editing systems with transactional techniques," in *Proc. 6th Int. ICST Conf. Collab. Comput. Netw. Appl. Worksharing (CollaborateCom)*, 2010, pp. 1–10.

[39] L. Xiong, X. Chen, T. Huang, J. Schneider, and J. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization," in *Proc. SIAM Int. Conf. Data Min. (SDM)*, 2010, pp. 211–222.

[40] H. Yin, B. Cui, X. Zhou, W. Wang, Z. Huang, and S. Sadiq, "Joint modeling of user check-in behaviors for real-time point-of-interest recommendation," *ACM Trans. Inf. Syst.*, vol. 35, no. 2, pp. 1–44, Oct. 2016.

[41] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for POI recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 11, pp. 2537–2551, Jan. 2017.

[42] J. Yoo and S. Choi, "Probabilistic matrix tri-factorization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2009, pp. 1553–1556.

[43] M. Ye, P. Yin, W. Lee, and D. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. (SIGIR)*, 2011, pp. 325–334.

[44] J.-D. Zhang and C.-Y. Chow, "CRATS: An LDA-based model for jointly mining latent communities, regions, activities, topics, and sentiments from geosocial network data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 2895–2909, Nov. 2016.

[45] J. Zhang and C. Chow, "Spatiotemporal sequential influence modeling for location recommendations," *ACM Trans. Intell. Syst. Technol.*, vol. 7, no. 1, pp. 1–25, 2015.

[46] Y. Zhang *et al.*, "Sequential click prediction for sponsored search with recurrent neural networks," in *Proc. 28th AAAI Conf. Artif. Intell. (AAAI)*, 2014, pp. 1369–1375.

[47] B. Zhang and Y. Feng, "Improving temporal recommendation accuracy and diversity via long and short-term preference transfer and fusion models," in *Web Technologies and Applications* (Lecture Notes in Computer Science), 2016. Cham, Switzerland: Springer, pp. 174–185.

[48] S. Zhao, M. Lyu, and I. King, "STELLAR: Spatial-temporal latent ranking model for successive POI recommendation," in *Point-of-Interest Recommendation in Location-Based Social Networks* (SpringerBriefs in Computer Science). Singapore: Springer, 2018, pp. 79–94.

[49] S. Zhao, M. Lyu, and I. King, "Geo-teaser: Geo-temporal sequential embedding rank for POI recommendation," *Point-of-Interest Recommendation in Location-Based Social Networks* (SpringerBriefs in Computer Science). Singapore: Springer, 2018, pp. 57–78.

[50] Y. Zhu *et al.*, "What to do next: Modeling user behaviors by time-LSTM," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 3602–3608.

[51] H. Zhang, Y. Yang, H. Luan, S. Yang, and T. Chua, "Start from scratch: Towards automatically identifying, modeling, and naming visual attributes," in *Proc. ACM Int. Conf. Multimedia (MM)*, 2014, pp. 187–196.

[52] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han, "GMove: Group-level mobility modeling using geo-tagged social media," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2016, pp. 1305–1314.

Yi-Cheng Chen received the Ph.D. degree from the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, in 2012.

He is currently an Assistant Professor with the Department of Information Management, National Central University, Taoyuan, Taiwan. His research interests include machine learning, social network analysis, data mining, and cloud computing.

Tipajin Thaipisutikul received the master's degrees (with Distinction) from the University of Sydney, Sydney, NSW, Australia, and National ICT Australia, Sydney, in 2012. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan.

Her research mainly focuses on applied deep learning, data mining, and social network analysis.

Timothy K. Shih received the Ph.D. degree in computer engineering from Santa Clara University, Santa Clara, CA, USA, in 1993.

He is a Distinguished Professor and the Vice Dean of the College of EECS, National Central University, Taoyuan, Taiwan.

Dr. Shih has received many research awards, including the IIAS Research Award, the HSSS Award, the Brandon Hall Award, and the Google MOOC Focused Research Award in 2015. He is an Associate Editor of IEEE COMPUTING, the IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES, the *ACM Transactions on Internet Technology*, and the IEEE TRANSACTIONS ON MULTIMEDIA. He is a Fellow of the Institution of Engineering and Technology, London, U.K.