This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

CEM: A Convolutional Embedding Model for Predicting Next Locations

Meng Chen, *Member, IEEE*, Yixuan Zuo, Xiaoyi Jia, Yang Liu, *Member, IEEE*, Xiaohui Yu, *Member, IEEE*, and Kai Zheng, *Member, IEEE*

***Abstract*— The widespread use of positioning devices and cameras has given rise to a deluge of trajectory data (e.g., vehicle passage records and check-in data), offering great opportunities for location prediction. One problem that has received much attention recently is predicting next locations for an object given previous locations. Several location prediction methods based on embedding learning have been proposed to tackle this issue. They usually focus on check-in trajectories and model sequential locations using an average of the embedding vectors. In this paper, we have proposed a Convolutional Embedding Model (CEM) to predict next locations using traffic trajectory data, via modeling the relative ordering of locations with a one-dimensional convolution. CEM is further augmented by considering constraints posed by road networks in the traffic trajectory data, learning a double-prototype representation for each location to eliminate the incorrect location transitions as well as modeling the combination of factors (such as sequential, personal, and temporal) that affect the human mobility patterns, and thus offers a more accurate prediction than just accounting for sequential patterns. Experimental results on two real-world trajectory datasets show that CEM is effective and outperforms the state-of-the-art methods.**

***Index Terms*— Trajectory embedding model, next location prediction, sequential patterns, traffic trajectory data.**

I. Introduction

THE increasing prevalence of positioning devices and surveillance cameras makes it possible to collect massive human mobility data. For example, the points-of-interest (POIs) that users have checked-in can be extracted when they share such information via online social networks (e.g., Foursquare) [1]; passing vehicles are photographed by the surveillance cameras, and structured vehicle passage records (VPRs) can be subsequently extracted from the pictures using optical character recognition (OCR) [2]. Such human mobility data are comprised of records with at least three attributes: *object IDs*, *location IDs*, and *time-stamps*. A consecutive sequence of such records generated by the same object ID constitutes a trajectory. While new technologies have made it easy to see objects' past locations, predicting their movement is nontrivial. Next location prediction is of great significance in many location-based services [3]–[9]. First, predictions may be used for providing personalized content, for example, in location-based advertising, places where users will go are important because they determine which kinds of ads to be posted. Second, aggregated predictions can be used for traffic management and long-term strategic planning, for instance, traffic management authorities may use the prediction on where the vehicle will go next to dynamically adjust traffic signals to alleviate traffic congestion. Finally, predictions could help traffic simulation, for example, we could generate a large number of vehicle trajectories given different variables based on a location predictor, and make what-if analysis in some tasks, e.g., traffic planing and signal timing.

The problem of next location prediction has been extensively studied. Existing methods can be roughly split into two categories based on the types of trajectory generation. One pays attention to next location prediction via the positioning data mostly generated by vehicles [2], [10]. The other focuses on the successive POI prediction and POI recommendation with check-in data on social networks, whereby they do not consider the constraints posed by road networks in the transportation systems [11], [12]. As the two kinds of trajectory data have different characteristics, one location prediction method cannot work well for both. In this paper, we aim to predict next locations using **traffic trajectory data** generated by vehicles. For example, as shown in Figure 1, an object moves from l_1 to l_4 and then to l_5 at around 7 a.m., and another moves from l_7 to l_8 and then to l_5 at around 9 a.m. Given the collection of trajectories of different objects, we predict the next location that an object will visit.

As multiple factors (e.g., object, preceding locations, and current time) affect the prediction of next locations, it is highly desirable to develop a unified framework that is able to model these factors and their relationships in a principled manner. We choose to use embedding techniques to map these different factors into the same vector

Manuscript received May 15, 2019; revised August 22, 2019 and December 29, 2019; accepted March 23, 2020. This work was supported in part by the Fundamental Research Funds of Shandong University, the Natural Science Foundation of Shandong Province of China under Grant ZR2019BF010, in part by the National Natural Science Foundation of China under Grants 61906107, 61572289, 61972069, 61836007, 61832017 and 61532018, and in part by the NSERC Discovery under Grant RGPIN-2017-05723 and Grant RGPIN-2018-06641. The Associate Editor for this article was S. Siri. *(Corresponding authors: Yang Liu; Xiaohui Yu.)*

Meng Chen and Xiaoyi Jia are with the School of Software, Shandong University, Jinan 250100, China (e-mail: mchen@sdu.edu.cn; jiaxiaoyishd@163.com).

Yixuan Zuo is with the School of Computer Science and Technology, Shandong Jianzhu University, Jinan 250101, China (e-mail: zxuan1004@163.com).

Yang Liu is with the Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo, ON N2L 3C5, Canada (e-mail: yliu@sdu.edu.cn).

Xiaohui Yu is with the School of Information Technology, York University, Toronto, ON M3J1P3, Canada (e-mail: xhyu@yorku.ca).

Kai Zheng is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: zhengkai@uestc.edu.cn).

Digital Object Identifier 10.1109/TITS.2020.2983647

1524-9050 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

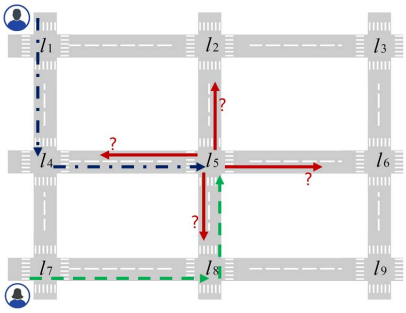


Fig. 1. An illustration of next location prediction.

space of real numbers (i.e., the embedded space), such that closely related objects/time slots/locations have similar representations in the embedded space. Some embedding methods [9], [11]–[13] have been proposed to mine the sequential context (i.e., the preceding locations) to predict next locations. They assume that locations in a sequential-context window are orderless (i.e., as in "bag-of-words" model), and represent the sequential context with an average of the vectors of locations within it. Furthermore, other methods [8], [14] deal with the sequential context based on Recurrent Neural Networks (RNN) or Long-Short Term Memory (LSTM). However, next location is mainly affected by its immediately preceding locations and each is not equally useful for accurate next location prediction.

Further, the constraints of the road networks should be seriously considered in the traffic trajectory data. For example, as shown in Figure 1, given two transitions $l_1 \rightarrow l_4$ and $l_4 \rightarrow l_5$, where l_i $(i = 1, 2, \ldots, 9)$ is a location, it is impossible for a vehicle to move directly from l_1 to l_5 (without passing any other locations) because of road network constraints. We call the transition $l_1 \rightarrow l_5$ a *phantom transition* because it may never happen in practice. Further, when an object arrives at location l_5, only four locations (l_2, l_4, l_6 and l_8) are the candidate next locations due to the constraints posed by road networks. Indeed, we do not need detailed road networks, as they can be generated using large number of trajectories [15]. In the best cases, if the underlying map topology is available, it can help refine prediction by pruning infeasible transitions.

Personal and temporal contexts are key factors in next location prediction but are not utilized in many existing methods. On one hand, people have personal preferences driven by their individual interests, habits and behaviors, which dominates their choices of where to go next. On the other hand, (as reported by [9], [12],) people tend to exhibit periodic moving behaviors, e.g., people usually leave home in the morning and return in the evening on weekdays. Furthermore, people may have different moving patterns at different time due to temporal constraints. For instance, as shown in Figure 1, turning right is forbidden on the road segment $l_8 \rightarrow l_5$ during morning rush hours, and people cannot arrive at l_6 directly from l_8, l_5. Hence, personal and temporal data can be used for improved prediction if used properly.

In order to address the above problems, we present a Convolutional Embedding Model (CEM) to predict next locations, via jointly using sequential, personal and temporal data as well as the constraints for road networks. In CEM, we first apply a one-dimensional convolution to the sequential data, modeling the relative ordering of locations. Furthermore, to consider the constraints of road networks, we distinguish between the roles of target locations and preceding locations, and learn a double-prototype representation per location to eliminate the *phantom transitions* from the trajectories. Finally, to model personal preferences and temporal factors, we represent the object and the time slot with different vectors, and embed the object ID, location ID, and time slot into a shared latent space, where a pair of more related attributes should be projected into closer embedding vectors. Based on the relations encoded in the latent representations, we are able to make successive location predictions.

Our major contributions are summarized as follows.

- We propose a Convolutional Embedding Model based on embedding learning to predict next locations using traffic trajectory data, which jointly models sequential patterns, personal preferences and temporal factors. Constraints of road networks as implicitly reflected by traffic trajectories are also taken into account by mapping each location to two different points in the embedding space.
- We learn the embedding vectors for objects, locations and time slots, and model the sequential context via a one-dimensional convolution, considering the relative ordering of locations in a context and learning the importance of each dimension of the sequential context to next location prediction.
- We conduct extensive experiments with real vehicle passage records and taxi trajectory data, and the experimental results confirm the superiority of CEM over alternative methods. Further, we create visualizations for the embedding vectors of objects and time slots, demonstrating the higher effectiveness of CEM.

II. Related Work

Trajectory data mining has become a hot research topic recently, and Zheng [16] conducted a systematic survey on this topic. As we focused on predicting next locations, we mainly discussed the recent progress in this field.

Most conventional methods adopt Markov models or frequent patterns mining models to mine human mobility patterns for location prediction. For example, Monreale *et al.* [17] considered the historical movements of all moving objects to build a T-pattern tree for prediction. Chen *et al.* [18] proposed to mine individual and collective movement patterns with an integrated variable-order Markov model to predict the successive locations. However, due to the high complexity of space and time, they usually model the first-order or second-order sequential transitions, because modeling the longer sequential context by standard counting methods is infeasible.

Recently, some new methods have been proposed that use recurrent neural networks to model sequential patterns in trajectory data for location prediction. For instance, Liu *et al.* [14] proposed a method called Spatial Temporal Recurrent Neural Networks (ST-RNN) to model the local

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

temporal and spatial contexts in each layer for mining mobility patterns. Yang *et al.* [19] presented a neural network by modeling social networks and mobile trajectories, where they employed RNN to capture the sequential relatedness in mobile trajectories. Yao *et al.* [20] proposed a method named Semantics-Enriched Recurrent Model (SERM) for location prediction using semantic trajectory data. SERM jointly learns embeddings of multiple factors (e.g., location, keyword) and transition parameters of a recurrent neural network in a unified framework. Kong and Wu [8] proposed a Hierarchical Spatial-Temporal LSTM model, which leverages historical visit information and spatial-temporal factors for location prediction. However, these RNN-based (or LSTM-based) methods focus on storing statistical weights for long-term transitions in a trajectory, and use the side features (e.g., geographical information, friendship network, semantic keywords) that do not exist in the traffic trajectory data.

Embedding objects from high-dimensional vectors into a lower-dimensional space is an important operation in machine learning, and has been successfully utilized for location prediction. Feng *et al.* [21] proposed a Personalized Ranking Metric Embedding method (PRME) to embed the POI transitions and user preferences in two latent spaces. Zhao *et al.* [13] proposed a Geo-Temporal sequential embedding rank (Geo-Teaser) that first encodes POIs in word2vec, treating check-ins in a day as a "sentence" and each POI as a "word". Zhou *et al.* [11] proposed a Multi-Context Trajectory Embedding Model (MC-TEM), considering various data types including user-level, location-level and temporal data. Chang *et al.* [22] proposed a content-aware POI embedding model to utilize the text content of a POI to improve prediction. Chen *et al.* [9] proposed a Mobility Pattern Embedding (MPE) method to embed the time slots, current locations and next locations together as points in a latent space. However, these embedding methods represent the sequential context with an average of the vectors of locations within it and ignore the relative ordering of locations.

In order to highlight our contribution, we summarized the main difference between our CEM and the aforementioned methods. First, to model the sequential context, we applied a one-dimensional convolution to the vectors of the preceding locations, learning how important each dimension is for location prediction. Second, we distinguished between the roles of "next locations" and "preceding locations", and learned a double-prototype representation per location to eliminate the *phantom transitions* from the trajectories. Finally, the sequential, personal and temporal data are modeled jointly to predict next locations.

III. The Convolutional Embedding Model

To better demonstrate the proposed CEM model, we first introduce some relevant definitions. Then we list the notations and their descriptions in Table I.

Definition 1 (Record): Each **record** is represented as a triplet $\langle o, l, t\rangle$, where o, l, t refer to the object ID, the location ID and the timestamp where o arrives at l, respectively.

TABLE I
Notations and Descriptions

Notations	Descriptions
o, l, t, T	object, location, time, trajectory
D	the embedding's dimensionality
K	the number of preceding locations in a context
$\mathcal{O}$	the set of objects
$\mathcal{L}^c$	the set of contextual locations
$\mathcal{L}^t$	the set of target locations
$\boldsymbol{v}$	an embedding vector
$\boldsymbol{v}_o$	the embedding vector for object o
$\boldsymbol{v}_t$	the embedding vector for time slot t
$\boldsymbol{v}_l$	the embedding vector for contextual location l
$\boldsymbol{v}'_l$	the embedding vector for target location l
$\mathbf{W}$	filter matrix

An example of triplet $\langle 101, 3006, 2016-01-14 \;\; 08:18\rangle$ tells that an object with the ID of 101 visits a location with the ID of 3006 at 8:18 a.m. on January 14, 2016.

Definition 2 (Trajectory): For an object o, its **trajectory** T is defined as a time-ordered sequence of records: $\langle o, l_1, t_1\rangle, \ldots, \langle o, l_i, t_i\rangle, \ldots, \langle o, l_N, t_N\rangle$, where N is the length of trajectory T and $t_i < t_{i+1}$ for $i < N$.

Definition 3 (Target and Contextual Location): Given a trajectory, the K locations $l_{i-K}, \ldots, l_{i-1}$ visited before one **target location** l_i are defined as the **contextual locations**. K is the pre-defined sequential context window size.

Note that, the target/contextual locations have the same meaning as next/preceding locations, and we will use target/contextual locations in the following sections.

Definition 4 (Contexts): Given a trajectory, the **contexts** of one target location l_i contain the contextual locations $l_{i-K}, \ldots, l_{i-1}$, the object o and the recent timestamp t_{i-1}.

A. Overview of CEM

Inspired by the recent progress of deep learning and neural networks [23], we propose to use a distributed representation method to model the generation of the given traffic trajectories. During the modeling process, we need to consider multiple kinds of contexts for a target location.

- An object's movement exhibits strong spatial-temporal regularity - the contextual locations and time slot can have strong influence on deciding the target location.
- The object's personal preference often plays an important role in choosing the target location.
- The constraints of road networks could help refine the target location.

Figure 2 illustrates our model, where object o_1 has visited locations l_6, l_5, l_4 between 8 a.m. and 9 a.m., and object o_2 has visited locations l_4, l_5, l_6, l_3 between 9 a.m. and 10 a.m. As multiple factors affect the prediction of next locations, we project objects, contextual locations, target locations and time slots into an embedding space, where closely related objects/time slots/locations should be projected into closer embedding vectors. For example, object o_1 visits locations between 8 a.m. and 9 a.m. (represented by time index t_8), while o_2 arrives at locations between 9 a.m. and 10 a.m.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

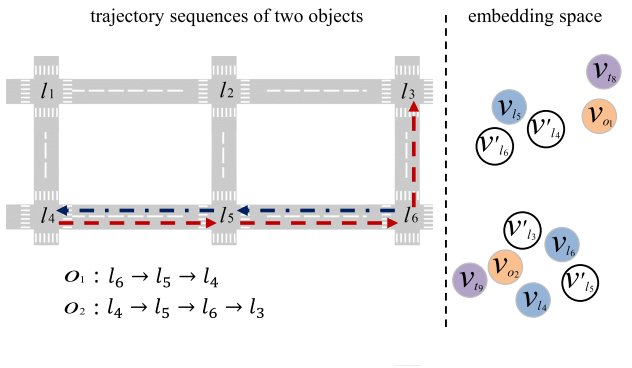


Fig. 2. An illustration of our method CEM. We respectively represent objects o_1 and o_2 with the embedding vectors $\boldsymbol{v}_{o_1}$ and $\boldsymbol{v}_{o_2}$ (denoted by circles with orange background), locations l_4, l_5 and l_6 taking the role of "contextual locations" with the embedding vectors $\boldsymbol{v}_{l_4}$, $\boldsymbol{v}_{l_5}$ and $\boldsymbol{v}_{l_6}$ (denoted by circles with blue background), locations l_3, l_4, l_5 and l_6 taking the role of "target locations" with the embedding vectors $\boldsymbol{v}'_{l_3}$, $\boldsymbol{v}'_{l_4}$, $\boldsymbol{v}'_{l_5}$ and $\boldsymbol{v}'_{l_6}$ (denoted by circles with white background). We use the hour index in a day to represent the time. The time duration between 8 a.m. and 9 a.m. is represented by t_8, corresponding to the embedding vector $\boldsymbol{v}_{t_8}$ (denoted by the circle with purple background), and the same with the duration between 9 a.m. and 10 a.m. Note that the distance in the embedding space is expected to capture the relatedness among locations, objects or time slots.

(represented by t_9). Therefore, the corresponding embedding vector $\boldsymbol{v}_{t_8}$ is closer to $\boldsymbol{v}_{o_1}$ than to $\boldsymbol{v}_{o_2}$. In the trajectory of o_2, l_4 and l_5 are two consecutive locations, and l_6 and l_3 are two consecutive locations; hence, $\boldsymbol{v}_{l_4}$ and $\boldsymbol{v}'_{l_5}$ tend to be closer, while $\boldsymbol{v}_{l_6}$ and $\boldsymbol{v}'_{l_3}$ tend to be closer, where $\boldsymbol{v}_l$ is the vector of the preceding location and $\boldsymbol{v}'_l$ is the vector of the target location. Since location l_3 is only visited by o_2, the location embedding $\boldsymbol{v}'_{l_3}$ is closer to $\boldsymbol{v}_{o_2}$ than to the embedding $\boldsymbol{v}_{o_1}$.

Formally, given a trajectory T generated by object o with N records: $\langle o, l_1, t_1\rangle, \ldots, \langle o, l_i, t_i\rangle, \ldots, \langle o, l_N, t_N\rangle$, the objective function is to maximize the average log probability for each target location l_i given its contexts $\bar{\boldsymbol{v}}_{l_i}$:

$$\frac{1}{N_T}\sum_{i=1}^{N_T}\log Pr(l_i|\bar{\boldsymbol{v}}_{l_i}), \tag{1}$$

where $\bar{\boldsymbol{v}}_{l_i}$ is a real-valued contextual vector consisting of all the contexts for the target location l_i and N_T is the length of trajectory T. We model each target location $l \in \mathcal{L}^t$ with a D-dimensional vector $\boldsymbol{v}'_l$, where $\mathcal{L}^t$ is the set of target locations, and assume that the generation of a target location is associated with its contexts. We then apply a multi-class classifier to generate a target location l_i based on its contextual vector via a softmax function as follows:

$$Pr(l_i|\bar{\boldsymbol{v}}_{l_i}) = \frac{\exp(\bar{\boldsymbol{v}}_{l_i}^T \cdot \boldsymbol{v}'_{l_i})}{\sum_{l\in\mathcal{L}^t}\exp(\bar{\boldsymbol{v}}_{l_i}^T \cdot \boldsymbol{v}'_l)}. \tag{2}$$

Such a model connects the target location and its contexts via the embedding representations, and regards next location prediction as a multi-classification task.

B. Modeling the Contexts

With the above general model, we now study how to model these contexts.

1) Modeling the Sequential Context: As we know, an object's next movement is influenced by its contextual locations in a trajectory [18], which can be caused by factors such as personal navigation habits and traffic. We reserve a D-dimensional embedding vector $\boldsymbol{v}_l$ for every raw contextual location $l \in \mathcal{L}^c$, where $\mathcal{L}^c$ is the set of contextual locations. To eliminate *phantom transitions* in each trajectory, we distinguish between the roles of target locations and contextual locations, and represent the same locations (e.g., l_5) via different vectors (e.g., $\boldsymbol{v}'_{l_5}$ for a target location and $\boldsymbol{v}_{l_5}$ for a contextual location as shown in Figure 2) in the same latent space depending on which role it takes. To illustrate why this is necessary, let us assume that location l_5 is mapped to a *single* point in the embedding space, irrespective of its role (contextual or target location). Given $l_6 \to l_5$ and $l_5 \to l_4$, if we knew that both l_6 and l_4 are close to l_5 in the space, then the transition $l_6 \to l_4$ could also exist with a high probability. However, it is very unlikely to observe $l_6 \to l_4$ due to the restriction of road networks or regulations, unless there is a direct route between l_6 and l_4.

Further, the target location is mainly affected by its immediately contextual locations, and each location does not contribute equally to the target location. However, most existing embedding methods [11]–[13] assume the preceding locations in a sequential-context window are orderless and represent these contexts with an average of the vectors of contextual locations. For example, given a trajectory sequence $l_5 \to l_6 \to l_3$ as shown in Figure 2, if we knew that the vector $\boldsymbol{v}'_{l_3}$ is close to the average of vector $\boldsymbol{v}_{l_5}$ and $\boldsymbol{v}_{l_6}$ in the embedding space, then it would be of high probability to acquire the sequence $l_6 \to l_5 \to l_3$. But it is impossible to observe $l_6 \to l_5 \to l_3$ in real traffic data, as there is not a direct road between l_5 and l_3. Therefore, we should consider the relative order of contextual locations and model the vectors of contextual locations with weights.

We represent the contextual locations with D-dimensional vectors, where each dimension could be regarded as a feature to decide which target location to visit. Hence we need to assign a separate weight to each feature. We introduce a one-dimensional convolution to model the weight of each feature for every contextual location, and produce a convolutional feature vector for the contextual locations. Given a trajectory T, we use a windowing approach that assumes that the target location l_i depends mainly on its K contextual locations. Formally, given the target location l_i, we build a matrix $\mathbf{U}^{\mathbf{i}} \in \mathbb{R}^{D\times K}$ via a lookup operation on the vectors of the K contextual locations. Each column of the matrix $\mathbf{U}^{\mathbf{i}}$ is the vector $\boldsymbol{v}_{l_{i-k}}$ of a contextual location $l_{i-k}, k = 1, 2, \cdots, K$. Then a one-dimensional convolution is used to yield a convolutional feature vector by taking the Hadamard product of the filter matrix $\mathbf{W} \in \mathbb{R}^{K\times D}$ with the matrix $\mathbf{U}^{\mathbf{i}}$ at the same dimension. After each row of $\mathbf{U}^{\mathbf{i}}$ is convolved with the corresponding column of the filter matrix, the new D-dimensional convolutional feature vector $\boldsymbol{v}_f$ is as follows,

$$\boldsymbol{v}_f = \sum_{k=1}^{K}\boldsymbol{v}_{l_{i-k}} \circ \mathbf{W}_k, \tag{3}$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

where $\circ$ is the Hadamard product. For two matrices $\mathbf{A}$ and $\mathbf{B}$ of the same dimension, the Hadamard product $\mathbf{A} \circ \mathbf{B}$ is a matrix of the same dimension as the operands, with elements given by $(\mathbf{A} \circ \mathbf{B})_{ij} = (\mathbf{A})_{ij}(\mathbf{B})_{ij}$. The trained weights in $\mathbf{W}$ are viewed as feature detectors learning the importance of each feature to the target location.

2) Modeling the Personal Context: Intuitively, the personal preferences are decisive in next location choices. For instance, as shown in Figure 1, two objects have arrived at location l_5, and they may visit different successive locations based on their preferences. Hence we need to capture each object's preference instead of treating them as equals. We reserve a D-dimensional vector $\boldsymbol{v}_o$ for every object $o \in \mathcal{O}$, where $\mathcal{O}$ is the set of objects. The distance between the object o ($\boldsymbol{v}_o$) and the target location l ($\boldsymbol{v}'_l$) in the latent space indicates the general preference of o over l.

3) Modeling the Temporal Context: Generally, an object is likely to have some regular activities, such as going to work in the morning and back home in the evening. Further, different movement patterns exist at different times. For example, as shown in Figure 1, if turning right is forbidden on the segment $l_8 \rightarrow l_5$ during rush hour (e.g., 8 a.m.-10 a.m.), objects will not choose location l_6 as next visit from l_8, l_5. Thus, the generation of a target location is likely to be influenced by the corresponding temporal information, such as a specific hour in a day. The original time in each record is a real-value timestamp. It is infeasible to embed every timestamp because time is continuous. We thus discretize a day into intervals of the same time duration and represent every timestamp with the time slot it belongs to. The duration of the slots can be determined experimentally.

We then consider each time slot t as a basic embedding unit, and reserve a D-dimensional vector $\boldsymbol{v}_t$ for each slot. When an object is to make next visit, the behavior is often influenced by the current time, and we choose the latest slot $\boldsymbol{v}_{t_{i-1}}$ as the temporal context of the target location l_i.

C. The Complete Model

To integrate the above three kinds of contexts, we maximize the objective function ℓ as follows,

$$\ell = \sum_{o \in \mathcal{O}} \sum_{T \in \mathcal{T}^o} \frac{1}{N_T} \sum_{i=1}^{N_T} \log Pr(l_i | l_{i-K} : l_{i-1}, o, t_{i-1}), \quad (4)$$

where $\mathcal{O}$ is the set of objects, $\mathcal{T}^o$ is the set of trajectories generated by object o and N_T is the length of trajectory T. We model each kind of context with a D-dimensional embedding vector. Given a target location l_i, the additivity assumption [9], [11] can be applied to multiple contexts. Then we derive its contextual embedding vector as follows,

$$\bar{\boldsymbol{v}}_{l_i} = \frac{1}{3}\left(\boldsymbol{v}_o + \boldsymbol{v}_{t_{i-1}} + \sum_{k=1}^{K} \boldsymbol{v}_{l_{i-k}} \circ \mathbf{W}_k\right). \quad (5)$$

We directly use the simple sum aggregation method. There can be other aggregation methods to integrate different kinds of context information, e.g., max pooling.

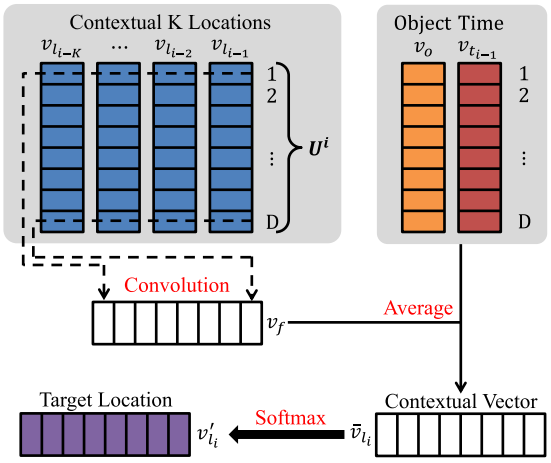


Fig. 3. The architecture of CEM. Orange, blue, purple and red boxes denote the embedding vectors for objects, contextual locations, target locations and time slots, respectively.

Finally, we apply a multi-class classifier to generate a target location l_j conditioned on $\bar{\boldsymbol{v}}_{l_j}$ based on Equation (2). The architecture of our CEM is shown in Figure 3. The input to CEM is the contexts of a target location l_i, including the D-dimensional vectors of its corresponding object o, time t_{i-1}, and K contextual locations $l_{i-k}, k = 1, 2, \cdots, K$. CEM models the ordering of contextual locations via a one-dimensional convolution and yields a convolutional feature vector $\boldsymbol{v}_f$. Then the contextual vector $\bar{\boldsymbol{v}}_{l_i}$ is generated by an average of the convolutional vector and the vectors of object and time. Finally, a multi-class classifier is employed to generate a target location l_i conditioned on $\bar{\boldsymbol{v}}_{l_i}$ based on Equation (2). We consider all the personal, temporal, and sequential contexts in CEM, and other available contexts can also be incorporated into this model by the average operation.

D. Parameter Learning

Our parameters include the vectors of objects $\{\boldsymbol{v}_o\}$, contextual locations $\{\boldsymbol{v}_l\}$, target locations $\{\boldsymbol{v}'_l\}$, time slots $\{\boldsymbol{v}_t\}$, and the filter matrix $\mathbf{W}$. For parameter learning, our CEM needs to maximize the objective defined in Equation (4). However, directly optimizing this objective is impractical because the cost of computing the full softmax for the multi-classifier is extremely high. In what follows, we derive how to update the parameters in CEM with hierarchical softmax.

The hierarchical softmax uses a binary tree representation for each target location as its leaves, and each node is explicitly associated with a vector for computing the relative probability to take one branch. Each leaf can be reached by an appropriate path from the root of the tree. In this way, instead of evaluating all the $|\mathcal{L}^t|$ output nodes to obtain the probability distribution, only about $\log_2(|\mathcal{L}^t|)$ nodes need to be evaluated.

All parameters are trained using Stochastic Gradient Descent (SGD). During the training process, the algorithm iterates over the target locations of the trajectories of all objects. At each time, a target location l_j with its contexts is used for update. After computing the hierarchical softmax, the error gradient is obtained via backpropagation and we use

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

the gradient to update the parameters in our model. Readers can refer to [11] for the details of gradient derivation and update. We iterate this procedure until the value of ℓ remains stable. Given the size of embedding vector (D) and the number of preceding locations in a context (K), the time complexity for an iteration is $O(W \cdot D \cdot K \cdot \log_2(|\mathcal{L}^t|))$, where W is the total number of training instances $(l_{j-K} : l_{j-1}, o, t_{j-1}, l_j)$ considered in an iteration and the log term comes from the binary tree to determine the target location from the $|\mathcal{L}^t|$ leaf nodes in the binary tree.

IV. Experiments

We first present experimental results on two real datasets to evaluate CEM on next location prediction, and then visualize the embedding vectors to confirm the effectiveness of CEM.

A. Datasets and Settings

1) Data: Two datasets are used in the experiments, the vehicle passage records (VPR) data and the publicly available taxi trajectory data.[1]

VPR data: We collected four-week (04/01/2016 - 31/01/2016) VPRs over the traffic surveillance system in the City of Jinan, China. For VPR data, passing vehicles are photographed by the surveillance cameras at fixed locations, and the cameras in the traffic surveillance system are considered as the sampling locations. To generate trajectories, for each vehicle, we first gathered all its records ($\langle vehicleID : o, locationID : l, timestamp : t\rangle$) and then sorted them according to the timestamp, and finally obtained a time-ordered sequence $\langle o, l_1, t_1\rangle, \cdots, \langle o, l_i, t_i\rangle, \langle o, l_{i+1}, t_{i+1}\rangle, \cdots$. We segmented the sequence into multiple trajectories using the time difference ($\Delta t = t_{i+1} - t_i$) between two consecutive visited locations. That is, if Δt exceeds 30 minutes, we could safely assume that the driver must have stopped somewhere in-between. In this paper, we set the time threshold at 30 minutes as the average distance between a neighboring pair of camera locations is approximately 3 km. Therefore, we regarded location l_i as the destination of a trajectory, and location l_{i+1} as the start point of another new trajectory. To make the model more robust, we included only the trajectories containing at least three locations. This results in 1,017,688 trajectories of 18,005 vehicles in the dataset.

Taxi data: The taxi data consists of all the complete trajectories of 442 taxis running in the city of Porto (Portugal) for a whole year (from 01/07/2013 to 30/06/2014). For Taxi dataset, each datapoint represents a complete taxi ride (which is considered as a trajectory): a sequence of GPS positions (latitude and longitude) measured every 15 seconds and the time of the beginning of the ride. The last position represents the destination and different trajectories have different GPS sequence lengths. As the number of GPS positions is enormous, we processed them with a mapping method. We discretized the region of interest into 200×100 grids with equal-sized cells (about 55 m $\times$ 53 m), and assigned a cell index for each GPS position. Then the cell index is regarded as the location ID. We also only kept the trajectories that have at least three locations, and obtained 410,803 trajectories in total.

[1]Detailed information about the data can be found at https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data

TABLE II
Data Statistics

	VPR data	Taxi data
♯objects	18,005	442
♯locations	482	14,684
♯trajectories	1,017,688	410,803
avg. ♯records of each object	342	27,895
avg. travel time between two locations	285.6	20.1
avg. ♯locations in each trajectory	6	30
avg. ♯trajectories of each object	56.5	929.4
density (loc/sq.km)	0.23	37.7

The statistical properties on both datasets are shown in Table II. ♯objects represents the count of objects, and avg. ♯locations is the average count of locations for each trajectory. From Table II we know that 1) VPR data contains more objects and Taxi data has more locations; 2) the average distance between a pair of neighboring locations of Taxi data is smaller than that of VPR data, and the average travel time (20.1 seconds) between two consecutive locations of Taxi data is also smaller; 3) each object from the Taxi data has more trajectories and each trajectory contains more locations; 4) the Taxi dataset contains denser records than the VPR dataset.

2) Evaluation Metrics: Given a trajectory $\langle o, l_1, t_1\rangle, \ldots, \langle o, l_i, t_i\rangle, \ldots, \langle o, l_N, t_N\rangle$ generated by object o with N records in the test set, we first built a trajectory sequence $\langle o, l_1, t_1\rangle, \ldots, \langle o, l_i, t_i\rangle, \ldots, \langle o, l_m, t_m\rangle$, where m is a random integer between 1 and $(N-1)$. The task of *next location prediction* is to predict the most likely next location l_{m+1}. We first built the contextual vector $\bar{\boldsymbol{v}}_l$ based on Equation (5), and then computed $Pr(l|\bar{\boldsymbol{v}}_l)$ for each candidate target location $l \in \mathcal{L}^t$ based on the following function:

$$Pr(l|\bar{\boldsymbol{v}}_l) \propto (\boldsymbol{v}_o + \boldsymbol{v}_{t_m} + \sum_{k=1}^{K} \boldsymbol{v}_{l_{m+1-k}} \circ \mathbf{W}_k)^T \cdot \boldsymbol{v}'_l. \quad (6)$$

Finally, we choosed the top r locations with the highest probabilities as the predicted next locations.

We utilized two well known metrics: *accuracy* and *average precision* (denoted by *acc* and *ap* respectively), to evaluate the prediction performance. *accuracy* is defined as the frequency of the true next location occurring in the list of predicted next locations. Let $P(l)$ be 1 it does and 0 otherwise. Then $acc = \frac{1}{|\mathcal{C}_t|} \sum P(l)$, where $|\mathcal{C}_t|$ is the trajectory count in the test set. Given a list of top-r predicted next locations, *average precision* is defined as $ap = \frac{1}{|\mathcal{C}_t|} \sum \frac{P(l)}{n}$, where n is the rank of the actual next location in the predicted list and $P(l)$ takes the value of 1 if the predicted location at the n-th position in the list is the true next location. Average precision puts larger weight to the top-ranked actual next location.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

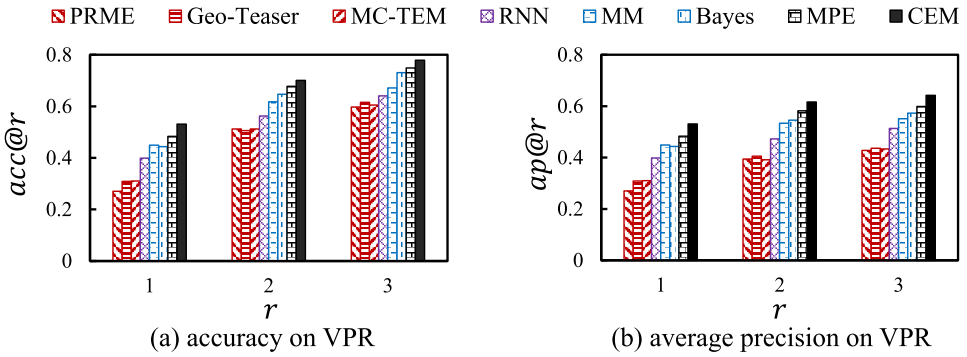


Fig. 4. Performance comparison for next location prediction on VPR data.

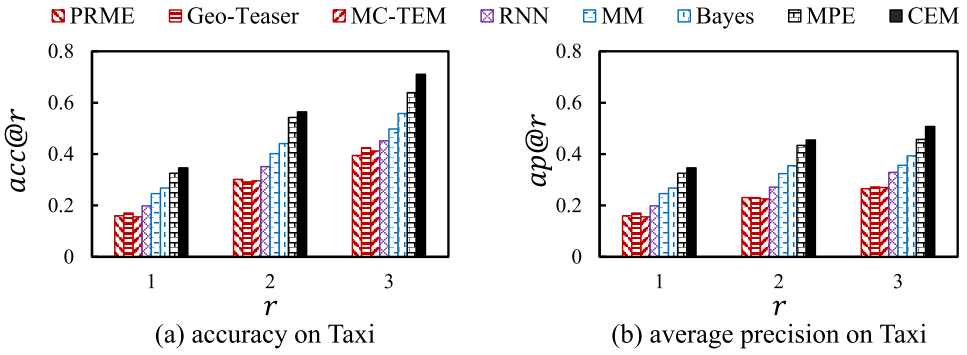


Fig. 5. Performance comparison for next location prediction on Taxi data.

3) Baselines: We compared with the following methods for predicting next locations to evaluate performance.

- **MM:** the Markov model [18], mining the mobility patterns of each object with its trajectories to predict next locations.
- **Bayes:** it computes the transition probability from the trajectory sequence to next location using Bayes' rules under the assumption that the attributes (object, location, and time) in the trajectory are independent.
- **RNN:** a recurrent neural network [14], [19], [20], modeling the long-term sequential context in a trajectory for predicting next locations.
- **PRME:** the personalized ranking metric embedding method [21], considering both the sequential information and user preference in training embedding vectors.
- **Geo-Teaser:** the geo-temporal sequential embedding rank model [13], incorporating the personal and temporal information into word2vec.
- **MC-TEM:** the multi-context trajectory embedding model [11], taking the user-level, location-level and temporal contexts into consideration.
- **MPE:** the mobility pattern embedding method [9], mainly modeling the transitions from the current locations to next.

In the task of next location prediction, we used 10-fold cross-validation and reported the average results. All the experiments are done on a 3.4GHz Intel Core i7 PC with 16GB main memory. For these embedding models, we searched the optimal hyper-parameters, and set the vector size and context window size at 100 and 5. For our CEM, we adopted the hierarchical softmax algorithm and set the number of embedding dimensions (D), the sequential context window size (K), and the learning rate at 100, 2, and 10^{-3}, respectively. In the following, we evaluated the effect of these hyper-parameters.

B. Experimental Results

We predicted top-r next locations and compared CEM with the baselines on VPR data and Taxi data. The prediction performance is shown in Figure 4 and Figure 5. The improvements from our CEM over all the baselines are statistically significant in terms of paired t-test [24] with p value < 0.01.

1) All the methods perform better on the VPR data than on the Taxi data, as the routes taken by taxis are more diverse/random.
2) MM considers the historical trajectories of each object, and gets decent accuracy and average precision on the VPR data and Taxi data. Bayes considers all the objects, contextual locations and time slots and outperforms MM. RNN focuses on capturing the long-term sequential transitions in trajectories, and it performs worse than MM and Bayes, as the choice of next location is usually affected by the recently visited locations. Our CEM considers the ordering of contextual locations and models the contextual locations, objects and time slots jointly instead of treating them independently, and gains an obvious improvement. For example, compared with Bayes, our CEM achieves an improvement of 19.6% on

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

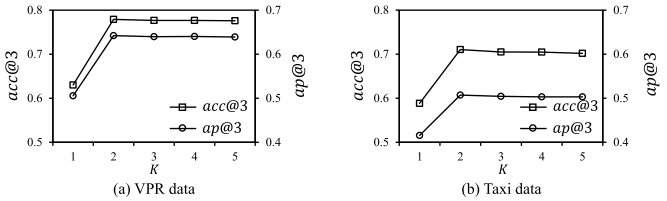


Fig. 6. Effects of varying the number of contextual locations K.

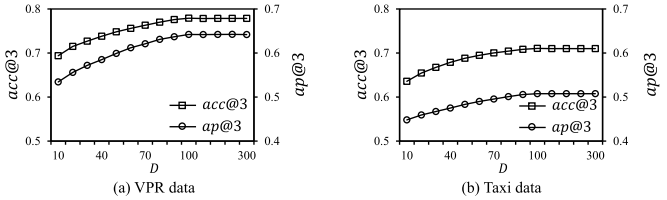


Fig. 7. Effects of varying the number of embedding dimensions D.

average in terms of top-1 accuracy on the VPR data, and 29.1% on average on the Taxi data.

3) PRME, Geo-Teaser and MC-TEM do not perform well, and the reasons are two-fold. On one hand, they represent both the contextual locations and the target locations with the same vector set, which is not applicable to the traffic trajectory data; on the other hand, they neglect the ordering of contextual locations, limiting their prediction performance. For instance, according to top-3 average precision, our CEM improves by 48.5% on the VPR data and 88.1% on the Taxi data compared with MC-TEM. MPE only considers the current location instead of the sequential context in the prediction, and it performs worse than our CEM.

C. Hyper-Parameter Optimization

As we needed to map the time-stamp of each record to the time slot it belongs to, we first set the size of the time slot at 1, 5, 10, 15, 30, 60 and 120 minutes respectively. The optimal size of the slot is 30 minutes for the VPR data, and 15 minutes for the Taxi data. There are two important hyper-parameters to tune in CEM: the sequential context window size (K) and the number of embedding dimensions (D). To measure their effect, we tuned them one by one on the validation set. The default values for K and D are 2 and 100, respectively. The tuning results on both datasets with top-3 accuracy and average precision are reported in Figure 6 and Figure 7, and the impacts of varying these parameters are discussed below.

We first varied K from 1 to 5, and found that the performance has an obvious improvement when K increases from 1 to 2 for both datasets. When we increased K further, the prediction performance starts to decline, because it takes some unrelated contextual locations into consideration. Further, we varied D from 10 to 300 and observed that the prediction performance improves evidently when we increased D from 10 to 100, and then remains flat when D is larger than 100.

TABLE III
RUNTIME OF ONE ITERATION (UNIT: SECOND)

embedding's dimensionality (D)	VPR data	Taxi data
10	0.03	0.07
50	0.06	0.15
100	0.10	0.26
200	0.17	0.41
300	0.26	0.62

D. Efficiency Analysis

In this part, we analyze the efficiency of the learning algorithm in our CEM with a hierarchical softmax. The time complexity for per-iteration training is $O(W \cdot D \cdot K \cdot \log_2(|\mathcal{L}^t|))$, where W is the total number of training instances considered in an iteration and the log term from the binary tree is used to determine the target location from the $|\mathcal{L}^t|$ leaf nodes in the binary tree. In the experiments, 100,000 training instances are used in one iteration, and CEM converges after about 200 iterations on the VPR data and 350 iterations on the Taxi data. When the dataset is given, the embedding's dimensionality D and the sequential context window size K are two hyper-parameters affecting the algorithm's efficiency. As K is set to a relatively small value, e.g., $K \leq 5$, we only studied how the time varies with the increasing D.

Table III shows the runtime of one iteration for both datasets with different D. On one hand, the runtime increases gradually when we raised D; on the other hand, because the Taxi dataset has more locations, its runtime is longer compared with the VPR dataset for the same D. Note that, we could train CEM offline in advance, and use the learned embeddings to support real-time applications.

E. Qualitative Analysis With Embedding Vectors

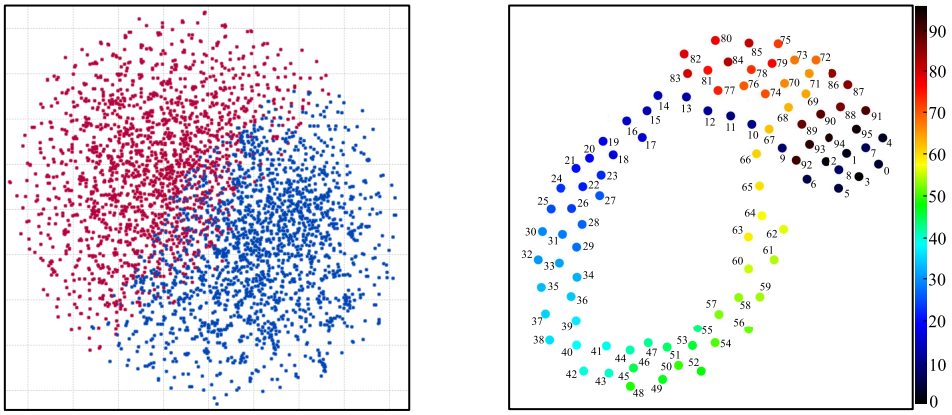
A major merit of our proposed CEM is that various kinds of contextual information (e.g., object, time) are projected into the same low-dimensional space, which allows us to visually explore the relations among objects or time slots.

1) Object Visualization: As we know, there are differences between the mobility patterns of private vehicles and those of taxis. To validate whether the private vehicles and taxis are visually distinguished in the learned latent space, we randomly selected 2000 private vehicles and 2000 taxis from our VPR data. We obtained the embedding vectors of the corresponding objects and projected them into a two-dimensional space with a 2D t-SNE [25] projection. As shown in Figure 8 (a), we observed two obvious classes, where the private vehicles are colored blue and the taxis are red, demonstrating that the embedding vectors are effective features for object classification.

2) Time Visualization: In our CEM model, we embedded the time slots into the latent space and measured the relationship between slots with their corresponding vectors. Here we took the Taxi data as an example, and visualized the relationship among these time slots. Figure 8 (b) shows the 2D t-SNE projection for the vectors of 96 time slots (the size of slots is 15 minutes for the Taxi data), where the time slots

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



(a) Object visualization (taxis are colored red and private cars are blue.)

(b) Time visualization (96 time slots are represented with gradient colors.)

Fig. 8. Visualization of embedding vectors.

are labeled with gradient colors. Some interesting phenomena can be observed in this figure. First, the time slots roughly scatter in a ring form and the temporal adjacent ones are still close to each other in the space. Second, the distances between "opposite" time slots in the ring are pretty large. For example, the positions of time slots in the morning are far from those in the afternoon. One likely reason is that there are different functional regions in a city [26], and people often move in opposite directions in the morning and in the afternoon (e.g., going to work vs. going home).

V. Conclusion

In this paper, we have proposed a Convolutional Embedding Model (CEM) to predict next locations with the traffic trajectory data. CEM models the sequential, personal and temporal contexts simultaneously and has two advantages. First, it models the ordering of contextual locations and learns the importance of each dimension of the sequential context to the target location via a one-dimensional convolution. Second, it accounts for restrictions of road networks by distinguishing between the roles of a target location and a contextual location. We evaluated the performance of CEM on two real datasets, and experimental results show that CEM outperforms the baselines.

In the future, we aim to consider more types of traffic data (e.g., Uber ride data) and incorporate some side features (e.g., semantic labels of locations) into our embedding model, and explore other potential applications (e.g., user clustering and location category prediction) with these embeddings. In addition, as the trajectory data is a typical streaming data, we will consider how to update these embedding vectors effectively with the new trajector

References

[1] S. Feng, G. Cong, B. An, and Y. M. Chee, "Poi2vec: Geographical latent representation for predicting future visitors," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 102–108.

[2] M. Chen, X. Yu, and Y. Liu, "TraLFM: Latent factor modeling of traffic trajectory data," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4624–4634, Dec. 2019.

[3] K. Soltani Naveh and J. Kim, "Urban trajectory analytics: Day-of-Week movement pattern mining using tensor factorization," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 7, pp. 2540–2549, Jul. 2019.

[4] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 543–554.

[5] H. Ying *et al.*, "Time-aware metric embedding with asymmetric projection for successive POI recommendation," *World Wide Web*, vol. 22, no. 5, pp. 2209–2224, Sep. 2019.

[6] M. Chen, X. Yu, and Y. Liu, "PCNN: Deep convolutional networks for short-term traffic congestion prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3550–3559, Nov. 2018.

[7] 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, Nov. 2017.

[8] D. Kong and F. Wu, "HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2341–2347.

[9] M. Chen, X. Yu, and Y. Liu, "MPE: A mobility pattern embedding model for predicting next locations," *World Wide Web*, vol. 22, no. 6, pp. 2901–2920, Nov. 2019.

[10] R. Trasarti, R. Guidotti, A. Monreale, and F. Giannotti, "MyWay: Location prediction via mobility profiling," *Inf. Syst.*, vol. 64, pp. 350–367, Mar. 2017.

[11] N. Zhou, W. X. Zhao, X. Zhang, J.-R. Wen, and S. Wang, "A general multi-context embedding model for mining human trajectory data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 1945–1958, Aug. 2016.

[12] W. X. Zhao, N. Zhou, A. Sun, J.-R. Wen, J. Han, and E. Y. Chang, "A time-aware trajectory embedding model for next-location recommendation," *Knowl. Inf. Syst.*, vol. 56, no. 3, pp. 559–579, Sep. 2018.

[13] S. Zhao, T. Zhao, I. King, and M. R. Lyu, "Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation," in *Proc. Int. Conf. World Wide Web Companion*, 2017, pp. 153–162.

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

[15] S. Karagiorgou and D. Pfoser, "On vehicle tracking data-based road network generation," in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst. (SIGSPATIAL)*, 2012, pp. 89–98.

[16] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, p. 29, 2015.

[17] 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. Discovery Data Mining*, 2009, pp. 637–646.

[18] M. Chen, X. Yu, and Y. Liu, "Mining moving patterns for predicting next location," *Inf. Syst.*, vol. 54, pp. 156–168, Dec. 2015.

[19] C. Yang, M. Sun, W. X. Zhao, Z. Liu, and E. Y. Chang, "A neural network approach to jointly modeling social networks and mobile trajectories," *ACM Trans. Inf. Syst.*, vol. 35, no. 4, pp. 1–28, Aug. 2017.

[20] 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. Manage.*, 2017, pp. 2411–2414.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.

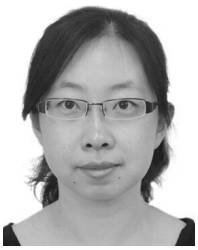
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[21] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new poi recommendation," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2069–2075.
[22] B. Chang, Y. Park, D. Park, S. Kim, and J. Kang, "Content-aware hierarchical Point-of-Interest embedding model for successive POI recommendation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3301–3307.
[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 3111–3119.
[24] D. Hull, "Using statistical testing in the evaluation of retrieval experiments," in *Proc. 16th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 1993, pp. 329–338.
[25] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[26] N. J. Yuan, Y. Zheng, X. Xie, Y. Wang, K. Zheng, and H. Xiong, "Discovering urban functional zones using latent activity trajectories," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 712–725, Mar. 2015.

Meng Chen (Member, IEEE) received the Ph.D. degree in computer science and technology from Shandong University, China, in 2016. From 2016 to 2018, he was a Post-Doctoral Fellow with the School of Information Technology, York University, Canada. He is currently an Assistant Professor with the School of Software, Shandong University. His research interest is in the area of data mining and traffic management.

Yixuan Zuo received the M.S. degree in visual design from Chung-Ang University, South Korea, in 2015. She is currently a Lecturer with the School of Computer Science and Technology, Shandong Jianzhu University, China. Her research interest is in the area of multimedia data mining.

Xiaoyi Jia received the B.S. degree in digital media technology from Qingdao University, China, in 2017. She is currently pursuing the M.S. degree in computer science and technology with Shandong University, China. Her research interest is in the area of traffic data mining.



Yang Liu (Member, IEEE) received the Ph.D. degree in computer science and engineering from York University, Canada, in 2008. She is currently an Associate Professor with the Department of Physics and Computer Science, Wilfrid Laurier University, Canada. Her main areas of research are data mining and information retrieval.

Xiaohui Yu (Member, IEEE) received the Ph.D. degree in computer science from the University of Toronto, Canada, in 2006. He is currently an Associate Professor with the School of Information Technology, York University, Toronto. His research interests are in the areas of database systems and data mining.

Kai Zheng (Member, IEEE) received the Ph.D. degree in computer science from The University of Queensland in 2012. He is currently a Professor of computer science with the University of Electronic Science and Technology of China. He has been working in the area of spatio-temporal databases, uncertain databases, social-media analysis, in-memory computing, and blockchain technologies. He has published over 100 articles in prestigious journals and conferences in data management field such as SIGMOD, ICDE, VLDB Journal, ACM Transactions and the IEEE TRANSACTIONS.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on April 15,2020 at 12:36:46 UTC from IEEE Xplore. Restrictions apply.