Pre-training Time-Aware Location Embeddings from Spatial-Temporal Trajectories

Huaiyu Wan, Yan Lin, Shengnan Guo, Youfang Lin

Abstract—With the increasing accumulation of spatial-temporal trajectory data, location-based data mining has recently been extensively studied. A fundamental research topic in this field is learning the embedding vectors of locations through self-supervised pre-training. Pre-trained embedding vectors can utilize the highly available unlabeled trajectory data, and benefit downstream tasks in multiple aspects. However, most existing methods ignore the temporal information hidden in the visited time of locations in trajectories. Considering that human activities are highly regulated by specific periods of a day, temporal information can reflect some intrinsic characteristics of locations, so it is necessary to fuse them into location embedding vectors. In this paper, we propose a Time-Aware Location Embedding (TALE) pre-training method based on the CBOW framework, which is able to incorporate temporal information into the learned embedding vectors of locations. A novel temporal tree structure is designed to extract temporal information during the calculation of Hierarchical Softmax. In order to verify the effectiveness of TALE, we apply the learned embedding vectors into three downstream location-based prediction tasks, i.e., location classification, location visitor flow prediction and user next location prediction. Experiments are conducted on four real-world user trajectory datasets, and the experimental results demonstrate that our TALE model can obviously help downstream tasks gain better performance.

Index Terms—Spatial-temporal data, location embedding, pre-training, trajectory modeling.

1 Introduction

With the increasing availability of location-based service (LBS) data, such as cellular signaling records, check-ins to point-of-interests (POIs) and taxi trajectories, mining spatial-temporal data has been extensively studied. Various tasks have gained much attention in recent years, including modeling users' mobility behaviors [1], [2], [3], predicting or recommending locations for users [4], [5], [6], [7], predicting visitors or crowd flows of locations [8], [9], and classifying functionalities of locations or areas [10], [11], etc. Among these researches, learning embedding vectors of locations through self-supervised pre-training is a very fundamental and critical problem, for the learned embedding vectors can benefit downstream tasks and applications in multiple aspects. Firstly, some location-based mining tasks, like location classification, are suffering from insufficient labeled-data for acceptable generalization performance. This can be solved by pre-training a set of location embedding vectors on large-scale trajectory datasets, which are often abundant. Secondly, compared to task-specific objectives, pre-training models can incorporate more comprehensive information into location embedding vectors, thus helping downstream tasks achieve better performance. Thirdly, by using general training objectives, embedding vectors learned by pre-training models can be utilized by a wide range of downstream tasks, which can reduce overall computational cost.

The key of learning embedding vectors for locations is to accurately model their sequential correlations in trajectories. Fortunately, the researches on neural network language models have developed some elegant methods such as word2vec [12], [13] to effectively capture the sequential semantic relationships among words. User check-ins to POIs and mobile trajectories, are also compatible with word2vec model for sequential influence modeling. Inspired by this idea, some researchers employ word2vec for trajectory data, like users' sequential check-ins or mobile signal data, to capture semantic relationships among locations [10], [14], [15], [16], [17].

When applied on trajectory data, word2vec only considers contextual information of locations. Yet, trajectories also possess some unique properties, including geographical positions of locations and users' personal interests, which can be dug to further improve the quality of location representations. Recently, Geo-Teaser [16] and POI2Vec [18] are proposed to incorporate geographical influence, i.e., users tend to visit nearby locations, into the word2vec model. Their experimental results give us an insight into how extra information incorporated into location embedding vectors can be beneficial for downstream tasks.

Ignored by most existing location embedding methods, temporal information involved in users' trajectories can also reflect intrinsic characteristics of locations. In daily life, people usually visit different destinations with corresponding purposes at appropriate time, *e.g.*, working in offices in business hours, having lunches in restaurants at noon, jogging or walking in parks in evening, sleeping at their own homes at night. Figure 1 gives an example of various visited frequency distributions along the time-of-day of different types of locations. It is clear that the functions carried by

Corresponding author: hywan@bjtu.edu.cn, yflin@bjtu.edu.cn.

- *Huaiyu Wan, Yan Lin, Shengnan Guo and Youfang Lin are with the Beijing Key Laboratory of Traffic Data Analysis and Mining, School of Computer and Information Technoloty, Beijing Jiaotong University, Beijing 100044, China, and the Key Laboratory of Intelligent Passenger Service of Civil Aviation, CAAC, Beijing, 101318, China. E-mail: hywan@bjtu.edu.cn; ylincs@bjtu.edu.cn; guoshn@bjtu.edu.cn; yflin@bjtu.edu.cn.*

Manuscript received xx xx, xxxx; revised xx xx, xxxx.

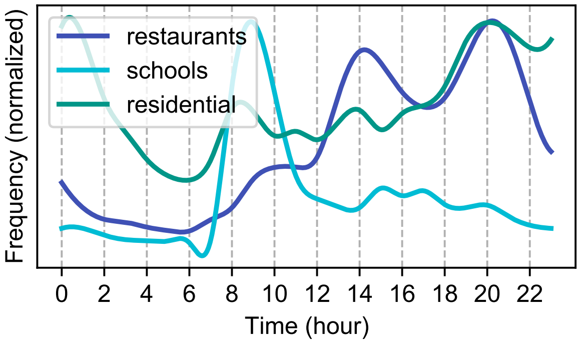


Fig. 1. The visit frequency distributions of three types of locations by Foursquare users in New York.

locations will affect users' mobility behaviors. Correspondingly, temporal information in trajectories implies users' preferences for locations. By taking temporal information into account, richer characteristics can be fused into the embedding vectors of locations. Yet, most latent location representation models ignore the temporal information.

In this paper, we propose a novel *Time-Aware Location Embedding* (TALE) model, which is inspired by the word2vec model, but able to further incorporate the temporal information in trajectory data for learning more exquisite location embedding vectors. In the model, each location is assigned with a low-dimensional embedding vector, as the latent representation of its "semantic" characteristics. The inner product of two embedding vectors reflects the relevance between the two corresponding locations.

To learn the embedding vectors efficiently, we exploit the Hierarchical Softmax (HS) technique [19], which constructs a binary tree structure over items. This technique is widely used in neural network language models. The structure of the binary tree controls the core calculation process of Hierarchical Softmax, so it can greatly affect the quality of the resulting embedding vectors [20]. In our TALE model, we propose a novel Hierarchical tree structure to incorporate the temporal information in trajectory data into the model. The tree structure is divided into two parts from top to bottom. The top part is a two-layer multi-branch tree, with one root node and T leaf nodes corresponding to T time slices of a day with the same length . And the bottom part consists of a set of Huffman sub-trees, constructed according to the visited frequency of locations being assigned into each time slice. A location can appear multiple times in the tree structure, as the same location can be visited in different time slices during a day. By utilizing the proposed tree structure, we are able to extract the temporal information in users' trajectories and incorporate it into the embedding vectors of locations in the training process. Theoretically, the resulting embedding vectors are more accurate and will contain richer characteristic information about locations.

To verify the efficiency of the proposed model, we apply the location representations learned by TALE into three downstream prediction tasks, i.e., location classification, location visitor flow prediction and user next location prediction. In addition to three public available check-in datasets from Foursquare, we also collect a mobile phone signaling dataset, as a showcase of data with higher density compared to traditional check-in data, to perform our experiments. The experimental results demonstrate that the TALE model can obviously improve the performance of the three downstream tasks.

In summary, the main contributions of this paper are as follows:

- We propose a time-aware location embedding model TALE, which utilizes trajectories generated by users to learn distributed location embedding vectors. The model is able to extract temporal information hidden in trajectories, and incorporate it into the embedding vectors of locations.
- A novel hierarchical tree structure is designed to model the temporal information in users' trajectories, which consists of T Huffman sub-trees by dividing one day into T time slices. Each sub-tree consists of locations that are visited during the corresponding time slice, and are organized according to their visited frequencies.
- We employ our TALE model into three downstream location-based prediction tasks, and conduct experiments on three real-world user check-in datasets plus one mobile phone signaling dataset. Experimental results show that the prediction performance is significantly improved, which demonstrate the effectiveness of our TALE model.
- Besides, we theoretically analyze how TALE model can incorporate temporal information into location embedding vectors from the parameter learning perspective, and also visualize some location embedding vectors as examples to prove our theoretical analysis.

The preliminary version of this paper has been published in DASFAA 2019 [21]. Compared to [21], we have made the following major improvements:

- We improve the temporal tree structure by proposing a new time splitting strategy. When constructing the tree, the original version fixed the length of one time slice to 1 hour, i.e., splitting one day into 24 time slices. In this paper, we treat the length of time slices as a hyper-parameter, so that it can be adjusted during the training process.
- When assigning locations into different time slices, locations that have close visited time might be assigned into different time nodes, due to the hard split of time slices. This can cause loss of temporal relationship. To address this problem, we introduce an "influence span" mechanism in this paper to smoothen the assignment process, so that the correlation between locations that are visited in close time period can be retained.
- We expand our experiments by applying our TALE model in different downstream tasks and introducing new datasets. Location classification task is added as an new task. DeepMove [22] is introduced as one of the downstream models in the task of user next location prediction. We also introduce three publicly available check-in datasets from Foursquare.

- We add a theoretical analysis and case visualization with regard to the effectiveness of the TALE model, from the perspective of parameter learning (as presented in Section 4.3 and 5.7). They prove that our model is indeed capable of capturing the temporal information in trajectory data.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 describes our TALE model in details. Section 4 presents the experimental evaluations. Section 5 gives the conclusion of this work.

2 Related Work

2.1 From Word Embedding to Location Embedding

The researches on location representation are largely motivated by the success of pre-trained word embedding in neural language processing (NLP), that utilizes probabilistic language modeling as its training objective [12], [23], [24], [25]. During the training of a language model, the appearance probability of a sentence $P_S = P(w_1, w_2, \dots, w_n)$ is given by the corpus. The embedding model aims to optimize its parameters, so that its estimation of the probability is precise enough. Word2vec [12] is a very popular word embedding method. It models P_S by calculating the posterior probability of the observing target word given its context, denoted as $P(w_i|C(w_i))$, where $C(w_i)$ is the context of the word w_i. By maximizing the posterior probability, word2vec is able to capture semantic information in sentences. Yet, the original word2vec model have a very high calculation expense, due to the involvement of Softmax process. In other words, the model has to implicitly calculate a probability for every word in the corpus, which has a time complexity of $O(|W|)$.

One of the solutions to improve word2vec's training efficiency is Hierarchical Softmax [13]. This approach builds a binary Huffman tree based on the occurrence frequency of words, where each word is represented by a leaf node, and every inner node can be treated as a binary classifier. Under the Hierarchical Softmax framework, $P(w_i|C(w_i))$ is equal to the probability of reaching the corresponding target leaf node of w_i through a path in the Huffman tree. Hierarchical Softmax diminish the time complexity from $O(|W|)$ to $O(\log |W|)$. This can be regard as a huge leap in calculation efficiency, especially considering that $|W|$ is usually a huge number.

The efficiency improvement of word2vec brings it into a usable state, thus the reputation expands into other research fields. Word2vec has been successfully applied in various domains, like user modeling [26], item modeling [27] and item recommendation [28]. These successful applications lead us to believe that word2vec definitely can be applied in location embedding. This is also due to sequential trajectories share many similar characteristics with sentences. For example, sentences and trajectories are both continuous sequences, and the functionalities of locations are akin to the semantic information of words. In the same time, user trajectories own some distinct characteristics compared to sentences. The most obvious one being there are temporal information assigned to trajectory data. Human activities are regulated by temporal states, i.e., people usually visit certain locations during certain periods. Correspondingly, temporal information in trajectories can reflect characteristics of locations, and shouldn't be ignored by pre-training location embedding methods.

In recent researches, [10], [14], [17], [29] employed word2vec to learn a set of location embedding vectors from users' check-in data. They treated each location as a word, and each visitor's sequence of visited locations as a sentence. Chang et al. [30] proposed a content-aware location embedding model, which integrates additional semantic information of text content into the POI embedding. Feng et al. [18] considered the effect of geographical relationships among locations in the learning process. In these researches, however, temporal information in trajectories were ignored. Cao et al. [31] replaced the fixed length context window in word2vec with a temporal span window, yet they didn't directly incorporate the visited time of locations into the result representations, thus will suffer from information loss. Zhao et al. [16] incorporated the effects of temporal influence into the representation learning model, but they only differentiated mobility behaviors during weekdays and weekends, which is not the full picture of temporal information. Chen et al. [28] tried to project various types of information like objects, locations and time into an integrated low-dimensional latent space. Yet, they only considered the relationships between two adjacent trajectory points, and ignored the sequential relationships among the locations. Yang et al. [32] presented a location embedding model STES, which uses feature vectors to encode geographic, temporal and functional information for location representation. The limitation of STES is that the functions of all the locations are needed as a prepositive information, which is impractical in most real application scenarios.

There are also a few location embedding methods that are based on N-gram architecture [33] or representation learning on graphs [34]. Shimizu et al. [11] implemented an N-gram structure on users' trajectories to learn embedding vectors for locations. Yet, N-gram structure is unidirectional, and is unable to consider contextual locations from both sides in a trajectory. Wang et al. [15] constructed a flow graph based on users' movements between geographical areas. Yet, by constructing flow graph rather than directly utilizing trajectories, some semantic information will be lost.

2.2 Location Embedding for Downstream Tasks

Most location-based data mining models are feature-based, that is, they require locations to be represented by latent embedding vectors. One of the simplest embedding strategies is to use one-hot vector. The length of a one-hot vector is the same as the total number of locations. It consists of 0s in all positions, except for a single 1 in the corresponding position used to identify a specific location. While simple, one-hot vectors cannot indicate any additional information besides the index of locations. In reality, a location carries rich characteristic information, such as its functionality and geographical position. There are also complex relationships between locations, like functional similarities. One-hot vector is unable to represent these kinds of information.

In addition to one-hot embedding technique, many location-based deep learning models use embedding layers to obtain task-specified embedding vectors [1], [22],

[35], [36]. In most cases, an embedding layer is essentially an embedding matrix $\mathbf{Z}$. A matrix multiplication process $\boldsymbol{z}(l_i) = \boldsymbol{o}(l_i)^\top \boldsymbol{Z}$ is used to fetch the embedding vector of location l_i, where $\boldsymbol{o}(l_i)$ is the one-hot vector of l_i. The embedding matrix $\boldsymbol{Z}$ is trained together along with the whole model using backward propagation. This will make the row vectors in $\boldsymbol{Z}$ becoming task-specific embedding vectors, since their training are guided by the model's supervised objective function. However, this approach have a few downsides. Firstly, the task-specific embedding vectors are difficult to migrate to other models and tasks. Secondly, using embedding layers will make models easily over-fitting on small-scale training data, and do not generalize well in practice.

Pre-training embedding vectors are widely used in computer vision [37] and neural language processing [38], and has recently attracted much attention in other fields. Pre-trained location embedding vectors can be learned through self-supervised objective, and only require unlabeled trajectory data, which is often abundant. Implementing such approach have many advantages. Firstly, pre-trained embedding vectors incorporate universal information of locations, such as functionalities and relative positions. This can help downstream tasks to achieve better generalization and accuracy. Secondly, pre-trained embedding vectors can be utilized by a wide range of downstream tasks and models without too much adjustment, which can improve overall calculation efficiency.

There are different approaches to pre-train general embedding vectors for locations, like using auto-encoder [39] to reduce the dimension of extracted feature vectors [40], utilizing supervised next-location prediction tasks [11], or extracting contextual information from trajectory data [10]. Among them, methods that are inspired by word2vec [12] are every popular, due to their high efficiency and flexibility. The basic idea is migrated from language modeling, which captures semantic information from sequences. By designing the structure of the embedding model, we can incorporate more information into the embedding vectors, thus helping downstream tasks to achieve better performance.

3 Preliminaries

In this section, we firstly introduce some definitions, and then give the *Problem Statement*.

Definition 1. **Spatial-Temporal Trajectory.** In a spatial-temporal dataset, user movements can be represented by a set of user trajectories H, in which each trajectory h consists of consecutive visiting points. A visiting point (l_i^u, t_i) indicates that user u visited location l_i at time t_i. If the visited time information is not utilized, a visiting point can be simplified into l_i^u. For easy presentation, we also denote the set of all locations as L, and the set of all users as U.

Definition 2. **Location Embedding Vector.** The embedding vector for a location l is a fixed length vector $\boldsymbol{z}(l) \in \mathbb{R}^d$, where the dimension d is regarded as a hyper-parameter. The embedding vector contains latent information about the location, *e.g.*, the functions and geographical position of the location, the relations with other locations, and so on.

Problem Statement. *Time-Aware Representation Learning of Location Embedding Vectors.* Given a set of historical user spatial-temporal trajectories H, we aim to learn an embedding vector $\boldsymbol{z}(l_i)$ for each location l_i in the set L. Apart from the sequential information, temporal information should also be extracted from the trajectories, to make the embedding vectors more precise.

4 Methodology

In this work we propose a Time-Aware Location Embedding (TALE) model to involve the temporal influence into location representation. Our method is motivated by the recent progress in pre-trained language modeling [12], [26], [27], [41], [42], which has been proved effective in capturing the semantic relationships among words from sequential data.

The model proposed in this paper is based on the Continuous Bag-of-Words (CBOW) [13] framework, one of the model architectures of word2vec. The basic idea of CBOW is to maximize the occurrence probability of a target word given its context, which guides embedding vectors of words with similar contextual environments being closer in the latent space. In this way, sequential relationships and semantic information in the corpus can be incorporated into word embedding vectors.

In this section, we first introduce how to transfer the basic CBOW framework and its efficiency-improved variant to the location embedding domain, and utilize CBOW framework to extract sequential information from user trajectories. Then we present a novel temporal tree structure for Hierarchical Softmax to incorporate temporal information into the location embedding.

4.1 Basic Location Representation Model

In order to incorporate characteristic information of locations into their embedding vectors, CBOW aims to make embedding vectors of locations with similar contextual environments being closer in the latent space. Specifically, given a user u and one of his/her visited location l_i^u in the trajectory $h_u = \{l_1^u, l_2^u, \ldots, l_n^u\}$, we define $C(l_i^u) = \{l_j^u, |j-i| \le \epsilon\}$ as the set of contextual locations of l_i^u in h_u, where ϵ is a hyper-parameter to control the context window size. The goal of location sequential modeling is to maximize the probability of a user visiting the true target location given its contextual locations.

In the basic CBOW framework, each location l_i is represented by two vectors, an input vector $\boldsymbol{z}(l_i)$ and a output vector $\boldsymbol{z}'(l_i)$. The input vector will be fetched as the result embedding vector of l_i, and the output vector is used only for training. To calculate the appearance probability of target location l_i given its contextual locations $C(l_i)$, we have:

$$P(l_i|C(l_i)) = \exp(\boldsymbol{z}'(l_i)^\top \boldsymbol{\phi}(C(l_i)))/Z(C(l_i)), \quad (1)$$

where $\boldsymbol{\phi}(C(l_i)) = \sum_{l_j \in C(l_i)} \boldsymbol{z}(l_j)$ is the element-wise sum of the input vectors of all the contextual locations, and $Z(C(l_i)) = \sum_{l_k \in L} \exp(\boldsymbol{z}(l_k)\boldsymbol{\phi}(C(l_i)))$ is a normalization factor. The training objective of CBOW is to maximize the aforementioned probability across all target-context pairs in the trajectory set H.

The computational cost of Equation (1) is very expensive, for it includes a Softmax progress. To be more exact, the computation of $Z(C(l_i))$ requires to traverse each location $l_k \in L$, leading to a time complexity of $O(|L|)$. In order to calculate the probability without traverse every location, we introduce the Hierarchical Softmax technique.

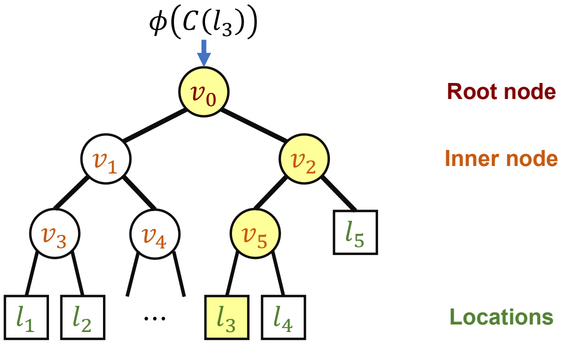


Fig. 2. Huffman tree structure constructed from user trajectories.

Hierarchical Softmax is widely used in Softmax computation. For implementation, we initialize each location as a leaf node, and using the occurrence frequencies of locations in the trajectory data to build a Huffman tree. The structure of the tree is shown in Figure 2. Each inner node v_i can be regarded as a binary classifier, with a hidden vector $\boldsymbol{\Psi}(v_i)$ as its parameter. By utilizing the tree structure, the probability $P(l_i|C(l_i))$ can be computed as the route probability from the root node v_0 to the leaf node l_i. Formally, we have:

$$P(l_i|\phi(C(l_i))) = \prod_{v_j \in path} \sigma(\langle v_j \rangle \cdot \boldsymbol{\Psi}(v_j)^\top \phi(C(l_i))), \quad (2)$$

where $\sigma(x) = 1/(1+e^{-x})$ is the Sigmoid function, $path$ denotes the set of inner nodes appearing in the path from v_0 to l_i, and $\langle v_j \rangle$ is a special function defined as:

$$\langle v_j \rangle = \begin{cases} -1 & \text{if } v_j \text{ choices left child in the path} \\ 1 & \text{otherwise} \end{cases}. \quad (3)$$

Thus, $\sigma(\langle v_j \rangle \cdot \boldsymbol{\Psi}(v_j)^\top \phi(C(l_i)))$ in Equation (2) can be regarded as the classification result of inner node v_j.

It is not hard to verify that $\sum_{l_i \in L} P(l_i|\phi(C(l_i))) = 1$, which means the result of Hierarchical Softmax is a valid multinoulli distribution among all locations. The use of Huffman tree structure makes it possible to get the shortest average path length through the whole training process, and achieve a time complexity of $O(\log |L|)$.

4.2 Time-Aware Location Embedding

4.2.1 Constructing Temporal Tree Structure

Human activities are usually time-regulated and location-constrained, so there is a strong correlation between a location's function and the users' arrival time. Therefore, in users' spatial-temporal trajectories, the time users arrive at a certain location contains information about the characteristics of the location. For example, if a user arrives at a certain place at 9:00 am on weekdays, that place is likely to be a work place. If a user arrives at a location at 6:00 pm on weekdays, that place may be a transportation hub or a restaurant. It is clear that incorporating temporal information into location embedding can improve the quality of embedding vectors.

If a location is visited only during a small time range, it implies that the location have relatively specific functions. For instance, the visiting records to a nightclub will centralize in late night. On the other hand, multi-functional locations are common in real world, and this type of locations will often be visited during a wide time range. For example, the visiting records to a large mall which undertakes the functionalities of supermarket, restaurant and cinema will scatter across the whole day. Figure 3 shows another example. Location l_1 was accessed by u_1 at t_1, u_2 at t_2 and u_3 at t_4, which means that l_1 is probably a multi-functional place. Meanwhile, location l_2 was only visited at t_3 by user u_2 and u_3, which means that l_2 is probably a single functional place. We can also see that both location l_2 and l_3 are visited by users at the same time t_3, which means that they probably have the same functions.

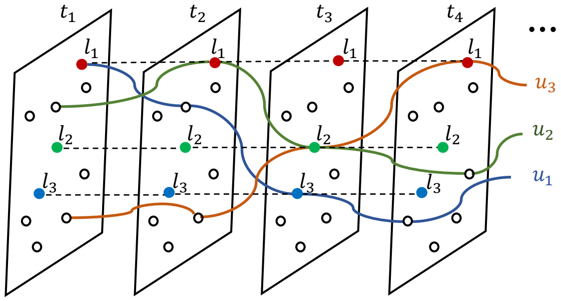


Fig. 3. An example of users' spatial-temporal trajectories.

In order to incorporate the temporal influence in users' trajectories into the location representation learning, we present a novel *temporal tree* structure for Softmax calculation. The temporal tree consists of two parts from top to bottom, as shown in Figure 4. The top part is a two-layer multi-branch tree. The first layer only contains one root node v_0, and the second layer contains T "time nodes" from v_{s_1} to v_{s_T}. We divide the time of a day into T equally-long time slices, denoted as $\{s_1, s_2, \ldots, s_T\}$, and let each time slice s_τ correspond to one time node v_{s_τ} $(1 \leq \tau \leq T)$. The length of each time slice is regard as a hyper-parameter and denoted as ι_{slice}. The bottom part of the tree is generated from user trajectories. We assign all visit records into these time slices according to their arrival time, and build a Huffman sub-tree for each time slice based on the visit frequency of locations. Then, for the sub-tree which contains visit records within time slice s_τ, we take time node v_{s_τ} in the top part as its root node. The construction process of temporal tree is given in Algorithm 1.

4.2.2 Soft Assignment Strategy of Visit Records

As a certain location is usually visited by users during different periods of a day, it is highly possible that one location is assigned to multiple time slices. But a problem still arises as the hard division of time slices conflicts with the continuous nature of time. This will lead to some degree of temporal information loss. Imagine that two restaurants being visited at 11:55 a.m. and 12:05 p.m. respectively. The visit times are very close, but if we divide one day into 24 time slices, these two locations will be assigned into different time slices and lost their correlation. In that scenario, we fail to capture the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3057875, IEEE Transactions on Knowledge and Data Engineering

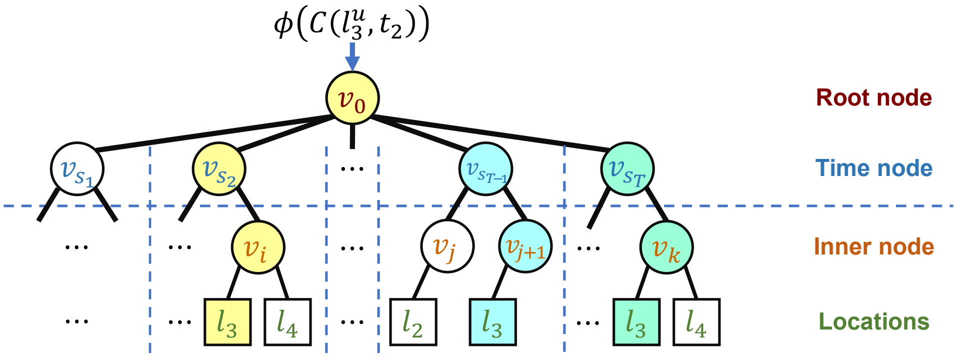


Fig. 4. The temporal tree structure of the TALE model.

```
Algorithm 1 Construction of temporal tree structure
Input: User set $U$, location set $L$, historical trajectories set $H$, length of time slice $\iota_{\text{slice}}$;
Output: Temporal tree structure;
 1: Initialize $T = \lceil 24\text{hours}/\iota_{\text{slice}} \rceil$ and time slices $\mathcal{S} = \{s_1, s_2, \ldots, s_T\}$, with time slice $s_\tau$ corresponding to time span $[\tau \cdot \iota_{\text{slice}}, (\tau+1) \cdot \iota_{\text{slice}})$;
 2: Create time nodes $\mathcal{V} = \{v_{s_1}, v_{s_2}, \ldots, v_{s_T}\}$, with time node $v_{s_\tau}$ corresponding to time slice $s_\tau$;
 3: Create a root node $v_0$ and set time nodes $\mathcal{V}$ as its child nodes;
 4: for Each time node $v_{t_\tau} \in \mathcal{V}$ do
 5:     Collect the set of visiting records $H_\tau = \{(l^u, t) | (l^u, t) \in H, t \in [\tau \cdot \iota_{\text{slice}}, (\tau+1) \cdot \iota_{\text{slice}}]\}$;
 6:     Collect location set $L_\tau = \{l | l$ appears in $H_\tau\}$ and calculate the occurrence frequency of each $l \in L_\tau$ in $H_\tau$;
 7:     Build a Huffman sub-tree $\mathcal{T}_\tau$ according to $L_\tau$ and their occurrence frequency;
 8:     Set time node $v_{s_\tau}$ as the root node of tree $\mathcal{T}_\tau$;
 9: end for
10: Return root node $v_0$, time nodes $\mathcal{V}$ and set of Huffman sub-trees $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$ as the temporal tree structure;
```

temporal relationship between locations that are visited in different time slices.

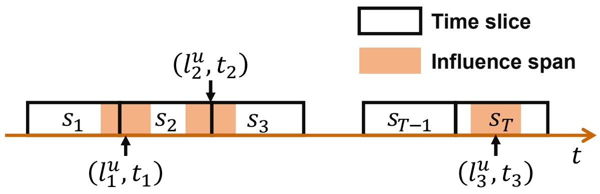


Fig. 5. The illustration of influence span overlapping multiple time slices.

In order to compensate for the temporal information loss, we intend to assign a visit record (l^u, t) to multiple time slices, so the temporal influence of a record can spread further. We consider the "influence span" of a record for that purpose. Formally, the influence span of visit record (l^u, t) is defined as a time period of length ι_{influ} and centered at t, i.e., time span $[t - \iota_{\text{influ}}/2, t + \iota_{\text{influ}}/2]$. If the influence span of (l^u, t) overlaps with a time slice s_τ, we assign this record to s_τ. Take a temporal tree with 24 time slices as an example. As shown in Figure 5, location l_2 is visited by a user at time t_2, and the time span centered at time t_2 overlaps with time slices s_2 and s_3, thus record (l_2^u, t_2) is assigned both to s_2 and s_3. The process of building the temporal tree stays mostly the same as denoted in Algorithm 1, except that in Step 5, the definition of record set H_τ is changed to:

$$H_\tau = \{(l^u, t) | (l^u, t) \in H, \\ [t - \iota_{\text{influ}}/2, t + \iota_{\text{influ}}/2] \cap \\ [\tau \cdot \iota_{\text{slice}}, (\tau+1) \cdot \iota_{\text{slice}}] \neq \varnothing\}. \tag{4}$$

It is easy to prove that one visit record will be assigned to no more than $\lceil \iota_{\text{influ}}/\iota_{\text{slice}} \rceil + 1$ time slice(s). We denote the set of time slices to which visit record (l^u, t) is assigned as $\Omega^{(l^u, t)}$.

If a visit record (l^u, t) is assigned to more than one time slice, we compute the probability of l belonging to time slice s_τ as:

$$P(s_\tau) = \mathcal{L}_{s_\tau}^{(l^u, t)} / \sum_{s_\kappa \in \Omega^{(l^u, t)}} \mathcal{L}_{s_\kappa}^{(l^u, t)}, \tag{5}$$

where $\mathcal{L}_{s_\tau}^{(l^u, t)}$ is the length of overlap between influence span $[t - \iota_{\text{influ}}/2, t + \iota_{\text{influ}}/2]$ and time slice s_τ.

As shown in Figure 4, there may be multiple paths for one location, each of which belongs to a different time slice. For example, l_3 appears three times, respectively in time slice s_2, s_{T-1} and s_T. Our model can learn and fuse the latent features of locations from different time slices.

In summary, TALE has two advantages against the pure Huffman tree structure used in word2vec [13]. First, our model incorporates the temporal influence of trajectory points into the process of building the tree structure. Locations appearing in similar time slices tend to exhibit similar characteristics, thus introduce richer function information into embedding vectors. Second, unlike the conventional models where each location only appears once, in our TALE model, a location may appear multiple times in the tree, which makes it be able to learn the location representations more accurately, since one location can be multi-purpose, and also, there exhibits various relationships between locations.

4.2.3 Probability Estimation

Using the proposed temporal tree structure, we can compute the probability of a user u visiting location l at time t given the contextual locations $C(l^u,t)$, i.e., $P(l^u,t|C(l^u,t))$. The Hierarchical Softmax method performs a Softmax classification by calculating the probability of a path from the root to the leaf node. In the temporal tree introduced above, leaf nodes corresponds to locations, and the other nodes are inner nodes. The root node v_0 has T branches, and can be regarded as a multi-class classifier. Other inner nodes can be regarded as binary classifiers, just like in the original Huffman tree.

The path from the root v_0 to leaf node l in time slice s_τ can be defined as a sequence of tree nodes, denoted as $path = (v_0^{s_\tau}, v_{s_\tau}^l, v_1^l, v_2^l, \ldots, v_n^l)$. We divide $path$ into two segments according to the structure of the temporal tree, i.e., $path = path_1 + path_2$. The first segment, $path_1 = (v_0^{s_\tau})$, only contains the root node that chooses the branch corresponding to time slice s_τ in the path. The second segment, $path_2 = (v_{s_\tau}^l, v_1^l, v_2^l, \ldots, v_n^l)$ belongs to a binary Huffman sub-tree, with which the time node v_{s_τ} as its root node. The probability of observing l^u in time slice s_τ along $path$ can be estimated by:

$$\begin{aligned} & P(l^u, s_\tau|C(l^u,t))^{path} \\ & = P(v_0^{s_\tau}|\boldsymbol{\phi}(C(l^u,t))) \prod_{v_i^l \in path_2} P(v_i^l|\boldsymbol{\phi}(C(l^u,t))) \\ & = P(s_\tau|C(l^u,t))^{path_1} \cdot P(l^u|C(l^u,t), s_\tau)^{path_2}. \end{aligned} \tag{6}$$

That is to say, given the contexts, the joint probability that a user u will visit the location l in the time slice s_τ is the product of two segments: the probability that the arrival time is in s_τ, and the probability that the visited location in s_τ is l. For a visit record (l^u,t) that is assigned to multiple time slices $\Omega^{(l^u,t)}$, we can calculate the probability of observing l^u at time t by summing up the probabilities of all paths according to Equation (5):

$$\begin{aligned} & P(l^u,t|C(l^u,t)) = \\ & \sum_{s_\tau \in \Omega^{(u,l,t)}} P(s_\tau) \cdot P(l^u, s_\tau|C(l^u,t))^{path}. \end{aligned} \tag{7}$$

Now we will explain how to calculate the probability of the two segments of path in detail. The root node v_0 has a latent matrix $\boldsymbol{M} \in \mathbb{R}^{T \times d}$, which can be treated as the parameters of the multi-class classifier. So the first segment in Equation (6) can be calculated as:

$$\begin{aligned} & P(s_\tau|C(l^u,t))^{path_1} = \\ & \exp(\boldsymbol{M}(\tau)^\top \boldsymbol{\phi}(C(l^u,t)))/Z(C(l^u,t)), \end{aligned} \tag{8}$$

where $\boldsymbol{M}(\tau)$ is the τ-th row of $\boldsymbol{M}$, $\boldsymbol{\phi}(C(l^u,t)) = \sum_{l_j \in C(l^u,t)} \boldsymbol{z}(l_j)$ is the sum of input vectors of all the locations in $C(l^u,t)$, and $Z(C(l^u,t)) = \sum_{\kappa=1}^{T} \exp(\boldsymbol{M}(\kappa)^\top \boldsymbol{\phi}(C(l^u,t)))$ is the normalization factor.

Each inner node v_{s_τ} and v_i $(i \geq 1)$ has a latent vector $\boldsymbol{\Psi}(v_i) \in \mathbb{R}^d$, which can be treated as the parameters of a binary classifier. $P(v_i^l|\boldsymbol{\phi}(C(l^u,t)))$ can be defined as:

$$P(v_i^l|\boldsymbol{\phi}(C(l^u,t))) = \sigma(\langle v_i^l \rangle \cdot \boldsymbol{\Psi}(v_i^l)^\top \boldsymbol{\phi}(C(l^u,t))), \tag{9}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the Sigmoid function, $\langle v_i^l \rangle$ is defined in Equation (3). Now we can calculate the second segment in Equation (6) as:

$$\begin{aligned} & P(l|C(l^u,t), s_\tau)^{path_2} \\ & = \prod_{v_i^l \in path_2} \sigma(\langle v_i^l \rangle \cdot \boldsymbol{\Psi}(v_i^l)^\top \boldsymbol{\phi}(C(l^u,t))). \end{aligned} \tag{10}$$

For better understanding of the calculation process of Equation (6), we give an example here. As illustrated in Figure 4, given a visit record (l_3^u, t_2), suppose that t_2 is within time slice s_2. The path for location l_3 in time slice s_2 is $path = (v_0, v_{s_2}, v_i)$. The probability of this path is:

$$\begin{aligned} & P(l_3^u, s_2|C(l_3^u,t_2))^{path} \\ & = \frac{e^{(\boldsymbol{M}(2)^\top \boldsymbol{\phi}(C(l_3^u,t_2)))}}{Z(C(l_3^u,t_2))} \\ & \times \sigma(\boldsymbol{\Psi}(v_{s_2})^\top \boldsymbol{\phi}(C(l_3^u,t_2))) \\ & \times \sigma(-\boldsymbol{\Psi}(v_i)^\top \boldsymbol{\phi}(C(l_3^u,t_2))). \end{aligned} \tag{11}$$

The time complexity of calculating the probability $P(l|C(l))$ in Equation (1) is $O(|L|)$. The maximum number of leaf nodes in our TALE model is $(T \times |L|)$, where T is the number of time slices. In our temporal tree structure, the average path length for all the leaf nodes is $log|L|+1$. So the time complexity of calculating the probability in Equation (6) is $O(\log|L|+1)$, which is much lower than $O(|L|)$.

4.2.4 Parameter Learning

The goal of the TALE model is to maximize the posterior probability of observing all visit records of the target location, given its contextual content in user trajectories. Assuming that the observed trajectories are independent with each other, then the optimization objective is:

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{argmax} \prod_{(l^u,t) \in h,\ h \in H} P(l^u,t|C(l^u,t)) \tag{12}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{Z}, \boldsymbol{M}, \boldsymbol{\Psi}\}$ is the set of parameters of the model, in which $\boldsymbol{Z}$ is the set of location embedding vectors, $\boldsymbol{M}$ is the parameter of the root node in the temporal tree structure, and $\boldsymbol{\Psi}$ is the set of parameters of all other inner nodes. We employ the Stochastic Gradient Descent (SGD) method [43] to learn all the parameters of the model.

For better understanding how the TALE model is able to incorporate temporal information from trajectories, we give a theoretical analysis from the parameter learning perspective below. And a case visualization of the embedding vectors is also presented in Section 5.7 to intuitively demonstrate the relations between the embedding vectors of locations.

4.3 Theoretical Analysis

In this section, we theoretical analyze how the TALE model is able to incorporate temporal information into location embedding vectors. Assuming one training instance with (l_i^u, t_i) being the target visiting record, $path$ being the set of inner nodes appears in the path form the root node to leaf node l_i^u which falls in time slice s_τ, $t_i \in s_\tau$. We can define the loss value for $path$ as follows. For simplicity, we denote $\boldsymbol{\phi}(C(l_i^u, t_i))$ as $\boldsymbol{c}$ here.

$$\begin{aligned}\mathcal{L}_{path} =& -\log P(l_i^u, t_i|\boldsymbol{c})\\ =& -\log \frac{e^{(\boldsymbol{M}(\tau)^\top \boldsymbol{c})}}{\sum_{\kappa=1}^{T} e^{(\boldsymbol{M}(\kappa)^\top \boldsymbol{c})}}\\ & -\log \prod_{v_i \in path} \sigma(\langle v_i \rangle \cdot \boldsymbol{\Psi}(v_i)^\top \boldsymbol{c})\\ =& -\boldsymbol{M}(\tau)^\top \boldsymbol{c} + \log \sum_{\kappa=1}^{T} \exp(\boldsymbol{M}(\kappa)^\top \boldsymbol{c})\\ & - \sum_{v_i \in path} \log \sigma(\langle v_i \rangle \cdot \boldsymbol{\Psi}(v_i)^\top \boldsymbol{c})\end{aligned} \tag{13}$$

We divide the above equation into two parts. The first part, $\mathcal{L}_1 = -\boldsymbol{M}(\tau)^\top \boldsymbol{c} + \log \sum_{\kappa=1}^{T} \exp(\boldsymbol{M}(\kappa)^\top \boldsymbol{c})$, which denotes the training loss of the multi-class classifier v_0. The second part, $\mathcal{L}_2 = -\sum_{v_i \in path} \log \sigma(\langle v_i \rangle \cdot \boldsymbol{\Psi}(v_i)^\top \boldsymbol{c})$, which denotes the training loss of the inner nodes in the Huffman sub-tree corresponding to the time node v_{s_τ}. Take the derivative of $\mathcal{L}_1$ with regard to $\boldsymbol{M}(j)^\top \boldsymbol{c}$, we have:

$$\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{M}(j)^\top \boldsymbol{c}} = \frac{\exp(\boldsymbol{M}(j)^\top \boldsymbol{c})}{\sum_{\kappa \in 1}^{T} \exp(\boldsymbol{M}(\kappa)^\top \boldsymbol{c})} - r_j^{v_0}, \tag{14}$$

where $\frac{\exp(\boldsymbol{M}(j)^\top \boldsymbol{c})}{\sum_{\kappa \in 1}^{T} \exp(\boldsymbol{M}(\kappa)^\top \boldsymbol{c})}$ is the output value of the j-th unit of v_o, which we denoted as $o_j^{v_0}$ in the following for simplicity. $r_j^{v_0}$ resembles the "true value" corresponding to $o_j^{v_0}$, and $r_j^{v_0} = 1$ only if s_j is the actual time slice this $path$ falls in, i.e., $j = \tau$, otherwise $r_j^{v_0} = 0$. This derivation result can be seen as the prediction error of the classifier v_0. Then we calculate the derivative of $\mathcal{L}_1$ with regard to $\boldsymbol{c}$ as:

$$\begin{aligned}\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{c}} =& \sum_{j \in T} \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{M}(j)^\top \boldsymbol{c}} \cdot \frac{\partial \boldsymbol{M}(j)^\top \boldsymbol{c}}{\partial \boldsymbol{c}}\\ =& \sum_{j \in T} (o_j^{v_0} - r_j^{v_0}) \cdot \boldsymbol{M}(j),\end{aligned} \tag{15}$$

which can be interpreted as sum of rows of classifier v_0's parameters, weighted by its prediction error.

Now we take the derivative of $\mathcal{L}_2$ with regard to $\boldsymbol{\Psi}(v_j)\boldsymbol{c}$, we have:

$$\begin{aligned}\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\Psi}(v_j)^\top \boldsymbol{c}} =& (\sigma(\boldsymbol{\Psi}(v_j)^\top \boldsymbol{c}) - 1)\langle v_j \rangle\\ =& \sigma(\boldsymbol{\Psi}(v_j)^\top \boldsymbol{c}) - r^{v_j}\end{aligned} \tag{16}$$

where $\sigma(\boldsymbol{\Psi}(v_j)^\top \boldsymbol{c})$ is the prediction result of classifier v_j, which we denoted as o^{v_j} in the following. r^{v_j} denotes the "true value" corresponding to o^{v_j}, where $r_j = 1$ if $\langle v_j \rangle = 1$, and $r_j = 0$ otherwise. This derivation result can be seen as the prediction error of the classifier v_j. Then we calculate the derivative of $\mathcal{L}_2$ with regard to the $\boldsymbol{c}$ as:

$$\begin{aligned}\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{c}} =& \sum_{v_i \in path} \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\Psi}(v_i)^\top \boldsymbol{c}} \cdot \frac{\partial \boldsymbol{\Psi}(v_i)^\top \boldsymbol{c}}{\partial \boldsymbol{c}}\\ =& \sum_{v_i \in path} (o^{v_j} - r^{v_j}) \cdot \boldsymbol{\Psi}(v_i),\end{aligned} \tag{17}$$

which can be interpreted as sum of inner nodes' parameters, weighted by their prediction loss.

Because we use gradient descent to train our model, and considering that $\boldsymbol{c} = \boldsymbol{\phi}(C(l_i^u, t_i)) = \sum_{l_k \in C(l_i^u, t_i)} \boldsymbol{z}(l_k)$, we can give the update equation for input vector of the one location $l_k \in C(l_i^u, t_i)$ as:

$$\boldsymbol{z}(l_k)^{(\text{new})} = \boldsymbol{z}(l_k)^{(\text{old})} - \frac{1}{|C|} \cdot \eta \cdot (\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{c}} + \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{c}}), \tag{18}$$

where η is the learning rate, $|C|$ is the number of contextual locations. Combining the information from Equation (15), (17) and (18), we can understand the process of parameter training in TALE as adding a portion of parameters of root node and inner nodes to the input vector of locations. The prediction loss of the root node v_0 and inner nodes $v_j \in path$ will decide whether to "push" $\boldsymbol{z}(l_k)$ away from their parameters, or to "pull" it closer. As we iterate through the whole dataset during the training process, this "push and pull" effect of parameter update will accumulate, leading to the result that embedding vectors of locations which are always visited during similar periods, and share more contextual neighbors being dragged closer in the embedding space.

From the above analysis, it can be seen that both contextual information and temporal influence in trajectories take part in the training process of TALE. This proves that our model is able to incorporate temporal information into location embedding vectors.

5 Experiments

To demonstrate the effectiveness of our proposed representation model, we incorporate location embedding vectors pre-trained by different models into multiple downstream applications, and conduct experiments on four real-world spatial-temporal trajectory datasets.

5.1 Datasets

We conduct our experiments on four datasets, three of which are Foursquare check-ins in New York, Tokyo and Jakarta, denoted as Foursquare-NYC, Foursquare-TKY and Foursquare-JKT. Check-in data are formed by users' arriving at functional locations, thus can be regard as users' trajectories.

The fourth dataset is constructed from mobile phone signaling data in Beijing, denoted as Mobile-PEK. It records the switching events between telecommunication base stations of mobile users in 5 consecutive workdays. We treat base stations as locations. To guarantee the quality of the trajectories, we filter out the ping-pong switches which are very common in mobile phone signaling data, and ignore the trajectory points that users just passed by.

For all the four datasets, we discard the locations which are visited by less than 5 users, and the users with less than 10 check-in records. The statistics of datasets after the pre-processing are shown in Table 1.

5.2 Baseline Location Embedding Models

To prove the effectiveness of the embedding vectors learned by our TALE model, we include some classic word embedding models and state-of-the-art location embedding models for comparison.

TABLE 1
Statistics of datasets.

Dataset	#User	#Location	#Check-in
Foursquare-NYC	1,077	3,908	82,091
Foursquare-TKY	2,290	7,057	389,063
Foursquare-JKT	9,193	13,105	536,792
Mobile-PEK	8,319	7,274	944,763

- **CBOW** [13]: An variation of the original word2vec [12]. Although proposed for word embedding, word2vec can be easily generalized to other embedding tasks based on sequential data. It can capture the semantic information based on the correlation between the target words and their contexts. In our implementation, we utilize the Hierarchical Softmax technique to accelerate its training process.
- **Skip-Gram** [13]: Another approach aiming to improve the training efficiency of the original word2vec. Compared to CBOW, it takes target words as input and contextual words as output. In our implementation, we utilizes the negative sampling technique to accelerate its training process.
- **POI2Vec** [18]: A location embedding model based on CBOW. This model considers that the geographical relationships among locations have impacts on user mobility behaviors and incorporates geographical features into the embedding learning process.
- **Geo-Teaser** [16]: Geo-temporal sequential embedding rank, a location embedding model based on Skip-Gram. It incorporates the effects of temporal influence into the location embedding through concatenating a temporal state vector with the target location's embedding vector as the input of Skip-Gram. The temporal state indicates whether the location is visited on weekdays or on weekends.

5.3 Downstream Prediction Models

Pre-trained location embedding methods capture general information of locations. Theoretically, the learned embedding vectors of locations should be beneficial for multiple location-based data mining tasks. On the other hand, the loss values of embedding methods are not consistent for a crosswise comparison. In this paper, we use three downstream tasks, i.e., location classification, location visitor flow prediction and user next location prediction, to verify the effectiveness of our model. Their results can be an indication of the quality of location embedding vectors.

5.3.1 Location Classification

Locations can usually be classified into multiple types based on their urban functions and population mobility patterns. Accurate location classification requires ample high-quality features of locations to be fed into the classifier. In this paper, we utilize two approaches to combine embedding vectors for location classification:

- **FC**: a multi-layer fully-connected network. For one location, we take its embedding vector as the input of the classifier, and the output vector as the classification result of the location.
- ***k*NN**: k-Nearest Neighbor [44] is a commonly used supervised classifier. Given a test location, the predicted class label is voted by its k nearest training locations. In our experiments, we regard the Euclidean distances between embedding vectors as the distance measurement of corresponding locations.

5.3.2 Location Visitor Flow Prediction

The visitor flow prediction of locations is a standard time series prediction task. Apart from the temporal correlation in historical flow sequences, information of locations can also help to achieve higher prediction accuracy. In this paper, we utilize a popular sequential modeling structure to fuse location embedding vectors into plain sequential modeling:

- **Seq2seq**: Sequence-to-sequence architecture [45] is a widely used approach for sequential correlation modeling. Given one target location, we first fuse each element of its visitor flow sequence with the embedding vector by casting them into the same dimension, and then concatenate them. The sequence of concatenated vectors is then split into the input of a GRU [46]-based seq2seq model's encoder and decoder. Finally, we utilize the output vectors of the decoder and a fully connected layer to calculate the prediction of future visitor flow values.

5.3.3 User Next Location Prediction

The goal of user next location prediction is to predict a user's next visiting location given a certain length of his/her historical trajectory. Accurate prediction of users' future moving choices requires locations to be represented by high-quality embedding vectors. In this paper, we incorporate location embedding vectors into two representative user next location prediction models:

- **GRU**: Gated Recurrent Unit [46] can be utilized to model users' sequential movement pattern. In our experiments, we implement a single-layer GRU network, and regard a user's historical trajectory as its input. The output hidden vector is then fed into a fully connected layer to make the prediction of the user's next visited location.
- **DeepMove** [22]: one of the state-of-the-art location prediction models, and a multi-module attentional recurrent network which utilize attention mechanism to model the multi-level periodicity of human mobility.

5.4 Settings

For all datasets, we firstly calculates the earliest and latest timestamp of all records, and split trajectories into training, evaluation and testing trajectory sets by 6:2:2 along time axis (one day is regarded as the smallest unit). Location embedding models are only trained on the training trajectory sets. We also split all locations into training, evaluation and testing location sets by 6:2:2 for location classification task. All downstream prediction models are trained on the

TABLE 2
Performance comparison of different approaches towards location classification.

Prediction Model		FC					kNN	
Metric		Acc@1 (%)	Acc@5 (%)	Acc@10 (%)	Acc@20 (%)	macro-F1 (%)	Acc@1 (%)	macro-F1 (%)
Dataset	Embedding Method							
Foursquare-NYC	Skip-gram	18.465±0.19	33.453±0.47	44.220±0.52	58.798±0.82	1.626±0.15	6.266±0.72	1.811±0.22
	CBOW	18.414±0.36	33.542±0.57	43.811±0.33	57.583±0.64	1.651±0.18	6.532±0.89	1.694±0.27
	POI2Vec	19.587±0.34	36.019±0.46	47.187±0.66	61.679±0.64	1.954±0.26	7.015±0.92	1.749±0.24
	Geo-Teaser	21.723±0.81	39.290±0.62	49.457±0.93	62.852±0.79	2.606±0.38	7.399±0.33	1.918±0.33
	TALE	**22.232±0.43**	**41.176±1.11**	**51.005±0.75**	**64.194±0.55**	**2.664±0.35**	**7.709±0.62**	**2.228±0.39**
Foursquare-TKY	Skip-gram	17.783±0.55	39.542±0.65	53.864±0.45	68.288±0.60	4.196±0.42	13.031±0.56	2.607±0.24
	CBOW	17.057±0.38	39.324±0.55	53.432±0.96	67.601±0.95	3.855±0.52	12.535±0.57	3.595±0.17
	POI2Vec	18.944±0.56	40.667±0.89	54.171±0.70	68.670±0.51	4.584±0.61	13.414±0.65	3.556±0.41
	Geo-Teaser	19.283±0.36	41.053±0.43	55.043±0.84	69.795±0.89	4.547±0.35	14.164±0.17	3.086±0.19
	TALE	**20.516±0.52**	**42.408±0.85**	**55.645±1.02**	**70.534±0.66**	**4.793±0.66**	**15.492±0.66**	**4.090±0.31**
Foursquare-JKT	Skip-gram	5.914±0.39	19.210±0.29	30.485±0.53	45.345±0.94	1.106±0.07	2.724±0.15	0.955±0.14
	CBOW	5.779±0.28	19.233±0.50	31.251±0.59	46.578±0.56	1.251±0.12	2.772±0.10	1.042±0.17
	POI2Vec	6.620±0.41	20.282±0.66	32.125±0.60	47.222±0.48	1.456±0.43	2.943±0.26	1.123±0.13
	Geo-Teaser	6.725±0.32	21.302±0.76	33.216±0.84	**48.595±0.84**	**1.666±0.23**	2.976±0.26	1.009±0.21
	TALE	**7.044±0.37**	**21.576±0.23**	**33.766±0.65**	48.522±0.27	1.578±0.39	**3.256±0.32**	**1.198±0.21**

TABLE 3
Performance comparison of different approaches towards location visitor flow prediction.

Metric	MAE	RMSE
Embedding Method		
Skip-gram	2.835±0.01	4.258±0.03
CBOW	2.771±0.01	4.143±0.02
POI2Vec	2.599±0.01	3.871±0.03
Geo-Teaser	2.535±0.01	3.773±0.02
TALE	**2.399±0.01**	**3.560±0.02**

training sets, validated on the evaluation sets to implement early-stopping technique, and tested on the testing sets to generate the final results. Noted that class labels of locations are only available in Foursquare check-in datasets, thus we only utilize these datasets for location classification task; Foursquare check-in datasets have too sparse visiting records for visitor flow calculation, thus only the mobile phone signaling dataset is utilized in visitor flow prediction task.

It's worth noting that the loss value is not a credible indication of the quality of a embedding method's generated representation vectors. Thus, we tuned the hyperparameters of embedding methods with the help of user next location prediction task and the DeepMove downstream model.

In our implementation, we use a time interval of one hour for visitor flow calculation. We standardize all flow values by removing the mean and scaling unit variance, i.e., $X' = (X - \mathrm{mean}(X))/\mathrm{std}(X)$, where $\mathrm{mean}(X)$ and $\mathrm{std}(X)$ are the mean and standard deviation of input flow value X, respectively. The prediction model for flow prediction is trained with Mean Square Error (MSE) loss. Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are chosen as evaluation metrics of flow prediction task. The prediction models for location classification and next location prediction are trained with Cross Entropy loss. Top-N accuracy (i.e., Acc@N, $N \in [1, 5, 10, 20]$) and macro-F1 are chosen as evaluation metrics of location classification and user next location prediction tasks.

We implement our TALE model based on the PyTorch [47] framework. For all embedding methods, we set the size of embedding vectors to 128, batch size to 64, window size to 2, and choose the Stochastic Gradient Descent optimizer with an initial learning rate of 0.001. For our TALE model, we set the time slice length ι_{slice} to 240 minutes, and the influence span length ι_{influ} to 60 minutes. For all downstream prediction models, we set the hidden size to 256, and choose the Adam optimizer with an initial learning rate of 0.001.

5.5 Experimental Results and Analysis

The result representation vectors calculated by baseline location embedding methods are incorporated into multiple downstream prediction models to get the results. Table 2, 3 and 4 show the performance comparison of different approaches for location classification, location visitor flow prediction and user next location prediction, respectively. Our TALE significantly outperforms all the baseline location embedding methods across all the datasets and downstream prediction models for most of cases.

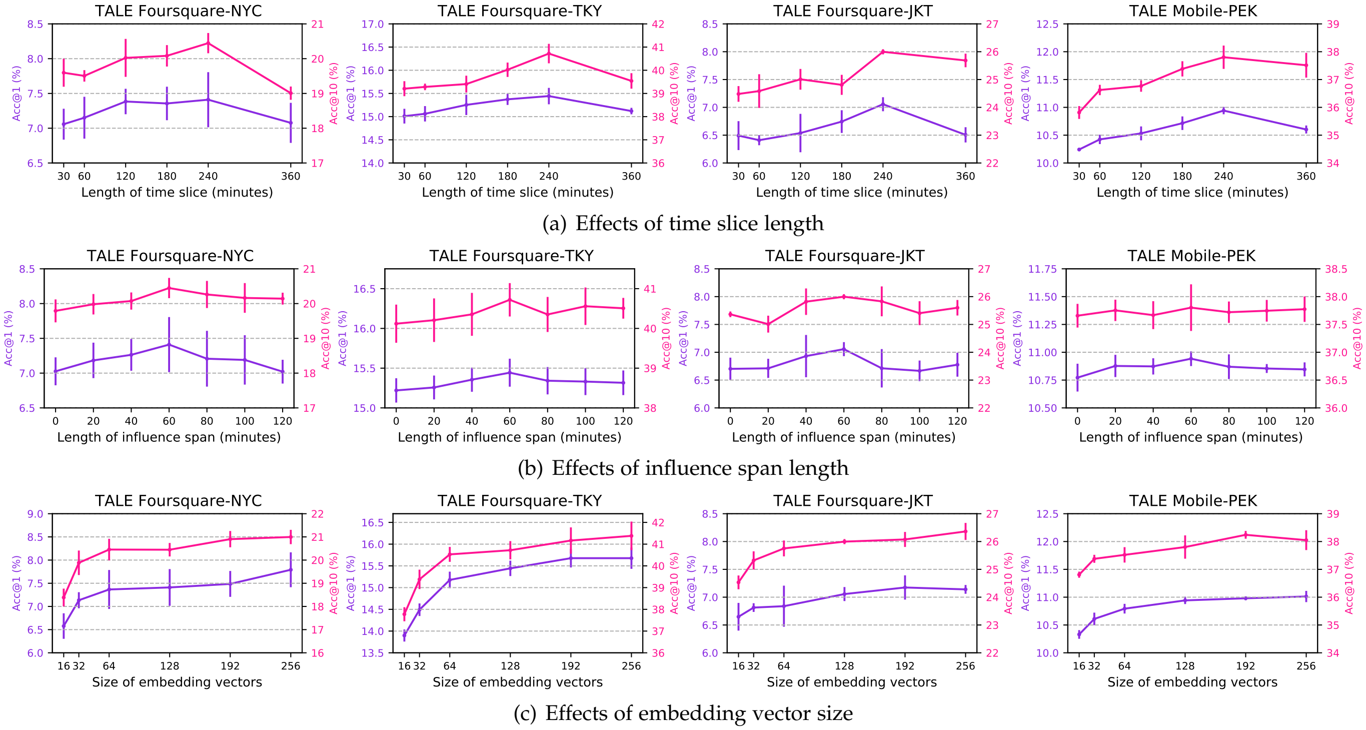
CBOW and Skip-gram are directly borrowed from the language modeling domain, and capture semantic information of locations from their relationships with contexts. Yet, they ignore spatial and temporal information, which are important aspects of user trajectory data, and can reflect characteristics of locations, as we discussed in Section 4.2. Unsurprisingly, location embedding vectors generated by these two methods generally perform the worst when incorporated into downstream models. POI2Vec takes spatial influence into consideration, based on the idea that users tend to visit nearby locations in a limited period. However, spatial information are not comprehensive and accurate enough for representing locations, for functionalities can be diverse across different locations in the real world, even in a small area.

Temporal information have a strong correlation with functionalities of locations, since locations with a certain function will have similar visited time distribution. Geo-Teaser incorporates temporal influence along with spatial information into the model. But for temporal influence

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3057875, IEEE Transactions on Knowledge and Data Engineering

TABLE 4
Performance comparison of different approaches towards user next location prediction.

Prediction Model		GRU					DeepMove				
Metric											
Dataset	Embedding Method	Acc@1 (%)	Acc@5 (%)	Acc@10 (%)	Acc@20 (%)	macro-F1 (%)	Acc@1 (%)	Acc@5 (%)	Acc@10 (%)	Acc@20 (%)	macro-F1 (%)
Foursquare-NYC	Skip-gram	5.466±0.23	11.870±0.33	14.986±0.34	18.878±0.37	1.254±0.12	6.170±0.21	13.351±0.52	16.555±0.58	20.310±0.56	1.354±0.16
	CBOW	5.248±0.22	10.849±0.22	13.522±0.16	16.367±0.30	1.203±0.09	6.191±0.10	13.282±0.47	16.304±0.49	19.814±0.40	1.556±0.11
	POI2Vec	5.935±0.19	12.529±0.35	15.910±0.30	19.961±0.31	1.471±0.10	6.357±0.28	13.786±0.25	17.239±0.22	21.237±0.31	1.515±0.12
	Geo-Teaser	6.209±0.32	13.677±0.43	17.602±0.37	21.996±0.39	1.533±0.11	7.007±0.28	15.472±0.41	19.458±0.17	23.741±0.56	1.717±0.08
	TALE	**6.918±0.39**	**14.874±0.44**	**18.911±0.34**	**23.018±0.31**	**1.685±0.20**	**7.410±0.40**	**16.455±0.41**	**20.447±0.29**	**24.711±0.19**	**1.867±0.10**
Foursquare-TKY	Skip-gram	10.705±0.17	25.417±0.27	32.986±0.30	40.944±0.34	1.650±0.08	13.173±0.14	28.965±0.23	36.651±0.28	44.529±0.32	2.269±0.07
	CBOW	10.373±0.10	23.899±0.21	30.690±0.24	37.899±0.23	1.720±0.09	12.823±0.18	27.391±0.42	34.340±0.43	41.528±0.39	2.350±0.15
	POI2Vec	11.425±0.13	26.785±0.22	34.570±0.26	42.854±0.26	2.023±0.04	14.306±0.11	30.532±0.21	38.533±0.17	46.665±0.15	2.958±0.01
	Geo-Teaser	12.024±0.18	28.696±0.38	37.176±0.40	45.851±0.45	2.413±0.10	14.669±0.15	32.231±0.28	**41.989±0.44**	**50.540±0.60**	3.150±0.10
	TALE	**12.637±0.16**	**29.753±0.39**	**38.476±0.40**	**47.239±0.37**	**2.848±0.10**	**15.443±0.18**	**33.453±0.27**	40.720±0.42	49.077±0.47	**3.622±0.07**
Foursquare-JKT	Skip-gram	5.524±0.17	14.100±0.17	19.853±0.20	26.840±0.14	0.776±0.07	5.713±0.15	14.423±0.23	20.498±0.26	27.654±0.15	0.835±0.13
	CBOW	5.290±0.07	13.773±0.25	19.367±0.31	26.055±0.35	0.898±0.09	5.736±0.16	14.309±0.22	19.965±0.27	26.815±0.29	1.025±0.15
	POI2Vec	5.876±0.17	15.356±0.24	21.650±0.27	29.392±0.19	0.846±0.09	6.229±0.11	15.904±0.07	22.350±0.32	30.029±0.43	1.076±0.16
	Geo-Teaser	**6.376±0.13**	16.740±0.30	23.808±0.50	32.148±0.37	1.025±0.12	6.533±0.11	17.225±0.40	24.248±0.31	32.300±0.30	1.444±0.07
	TALE	6.278±0.20	**17.901±0.28**	**25.424±0.44**	**34.132±0.38**	**1.374±0.09**	**7.056±0.13**	**18.640±0.17**	**25.997±0.09**	**34.711±0.31**	**1.469±0.07**
Mobile-PEK	Skip-gram	7.994±0.09	22.574±0.14	31.232±0.18	40.819±0.20	3.453±0.10	8.650±0.10	23.564±0.09	31.703±0.17	40.486±0.22	3.996±0.07
	CBOW	8.458±0.07	22.796±0.16	30.418±0.18	38.517±0.19	4.122±0.05	8.895±0.05	23.980±0.14	31.778±0.16	39.923±0.17	4.632±0.09
	POI2Vec	9.535±0.09	25.761±0.03	34.883±0.14	44.416±0.14	4.789±0.14	9.762±0.15	26.254±0.06	34.977±0.05	44.151±0.16	5.044±0.10
	Geo-Teaser	9.351±0.08	25.680±0.15	34.791±0.17	44.508±0.26	4.523±0.12	9.639±0.06	25.856±0.12	34.517±0.14	43.711±0.21	4.819±0.20
	TALE	**10.939±0.06**	**29.319±0.13**	**38.860±0.19**	**48.490±0.22**	**5.010±0.12**	**10.942±0.07**	**28.479±0.29**	**37.802±0.42**	**47.301±0.54**	**5.524±0.22**



(a) Effects of time slice length

(b) Effects of influence span length

(c) Effects of embedding vector size

Fig. 6. Effects of hyper-parameters validated on DeepMove.

it only differentiates the human mobility trajectories occurred on weekdays and weekends. Compared to the forementioned baselines, TALE utilizes a temporal tree struture, and can model the relationships between the locations that are visited during similar time periods of a day. In this way, TALE is able to incorporate more detailed and accurate functional information of locations into their generated embedding vectors. Thus, downstream location-based prediction tasks gain highest accuracy when coupled with our TALE model.

5.6 Effects of Parameters

In this section, we evaluate the effects of three hyper-parameters: time slice length ι_{slice}, influence span length ι_{influ} and the size of embedding vectors. The experiments are conducted on user next location prediction using DeepMove downstream model, and while evaluating one of the hyper-parameters, we lock the other ones to optimum.

5.6.1 *Effects of Time Slice Length*

Figure 6(a) shows the experimental results on the hyper-parameter tuning of time slice length ι_{slice}. From these figures, we observe that the performance first improves as we lengthening the time slice, then deteriorates as it exceed the optimum point. A small ι_{slice} means only locations that are visited in very close time interval are clustered into the same time slice, which fails to capture the relations between locations that are visited during similar periods. A big ι_{slice} will clusters too many locations into one time slice. This will make the method failing to model the temporal information in trajectories, and degenerate into the basic CBOW [13] model. A moderate ι_{slice} can capture relations between locations with similar visiting time patterns, while also distinguish locations that are typically visited during different periods. Hence, we set ι_{slice} to 240 minutes across all datasets.

5.6.2 Effects of Influence Span Length

Figure 6(b) shows the experimental results on the hyper-parameter tuning of influence span length ι_{influ}. A small ι_{influ} will cause some locations which are visited in close time intervals be assigned into different time slices, thus discarding the relations between these locations. A big ι_{influ} will cluster one location into irrelevant time slices. This will cause the method unable to model distinct visiting time patterns for locations. In conclusion, we set ι_{influ} to 60 minutes for all datasets.

5.6.3 Effects of Embedding Vector Size

Figure 6(c) shows the experimental results on the hyper-parameter tuning of the size of embedding vectors. This figure demonstrates that the performance improves steadily as we increase the size of location embedding vectors. Longer vectors are able to contain more comprehensive information, thus helping downstream tasks gain better results. Yet, we observe that the degree of performance improvement is limited when the size is bigger than 128, and longer embedding vectors will lead to higher computational expense. In conclusion, we set the size of embedding vectors to 128 globally, by considering the trade-off between effectiveness and efficiency.

5.7 Case Visualization of Location Embeddings

We choice a small subset of locations in the Foursquare-TKY dataset, and visualize their embedding vectors in a 2-dimensional space to get a clear acknowledge of their relations in the latent space. We use t-SNE method [48] for dimension reduction. Then we choice different pairs of locations, whose embedding vectors are close to each other, or away from each other, respectively, and visualize users' visiting patterns to these locations by drawing a histogram of visiting time for each location. The visualization results of our TALE model are shown in Figure 7. We can see that locations whose embedding vectors are closed to each other often have similar visited time patterns, like locations #1555 and #457, which are both train stations. In the mean time, locations whose embedding vectors are far away from each other often have high divergence in visited time patterns, like locations #1555 and #12, in which location #12 is a subway.

We do the same visualization to the embedding vectors learned by CBOW, as shown in Figure 8. It is clear that locations with every different visited time patterns can be closed to each other in the embedding space acquired by CBOW, meaning CBOW totally ignores the temporal information in trajectory data, and only uses the contextual information to guide the training process. Compared to our TALE model, this will lead to information loss and decrease in embedding vectors' quality.

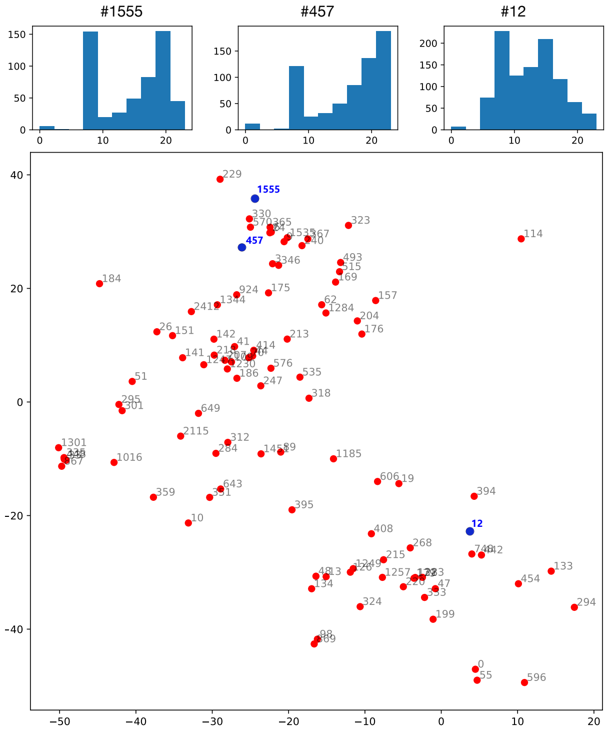


Fig. 7. Visualization of location embedding vectors learned by TALE.

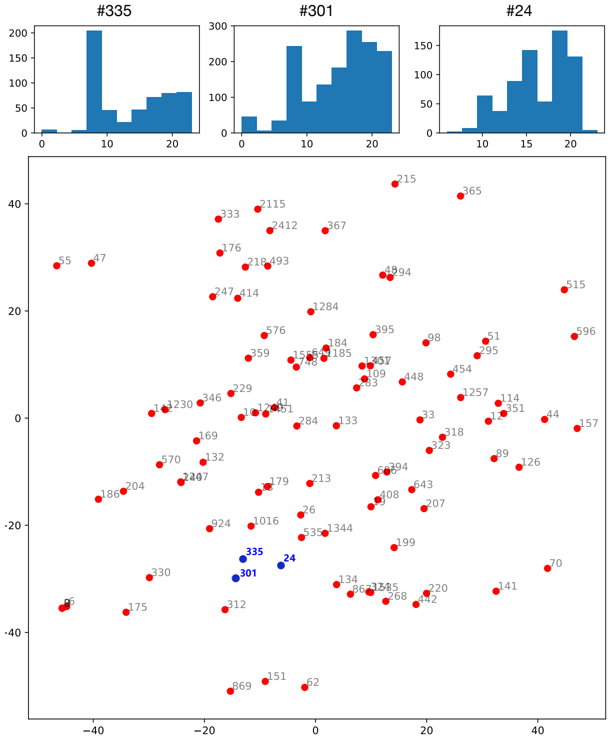


Fig. 8. Visualization of location embedding vectors learned by CBOW.

6 Conclusion

In this paper, we propose a novel time-aware location embedding pre-training model TALE. It is able to incorporate the temporal information in users' mobility trajectories into the embedding vectors of locations. A novel tree structure is designed based on the Hierarchical Softmax to model the temporal influence. In order to evaluate the effectiveness of the learned embedding vectors, we involve TALE into three location-based mining tasks, i.e., location classification, location visitor flow prediction, and user next location prediction. Experimental results show that our model can improve the performance of various downstream tasks compared to other existing location embedding models.

There are some interesting issues that can be further studied. Firstly, we only consider the arrival time of locations in users trajectories. However, the information that how long a user stay in a location may also be helpful to represent the characteristics of locations. If we can incorporate such duration information into location representations, more meaningful location embedding vectors may be learned. Secondly, based on effective location representations, a lot of location-based mining tasks can be improved, such as location recommendation, path recommendation, etc.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities, China (Grant No. 2019JBM024).

References

[1] D. Kong and F. Wu, "HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 2341–2347.

[2] X. Li, G. Cong, A. Sun, and Y. Cheng, "Learning travel time distributions with deep generative model," in *Proceedings of the The World Wide Web Conference 2019*, 2019, pp. 1017–1027.

[3] T.-Y. Fu and W.-C. Lee, "TremBR: Exploring road networks for trajectory representation learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 11, no. 1, pp. 1–25, 2020.

[4] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new POI recommendation," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015, pp. 2069–2075.

[5] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "ATRank: An attention-based user behavior modeling framework for recommendation," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 4564–4571.

[6] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, 2018, pp. 108–116.

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

[8] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "Urban traffic prediction from spatio-temporal data using deep meta learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1720–1730.

[9] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal sychronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2020.

[10] Z. Yao, Y. Fu, B. Liu, W. Hu, and H. Xiong, "Representing urban functions through zone embedding with human mobility patterns." in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3919–3925.

[11] T. Shimizu, T. Yabe, and K. Tsubouchi, "Learning fine grained place embeddings with spatial hierarchy from human mobility trajectories," *arXiv preprint arXiv:2002.02058*, 2020.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the 1st International Conference on Learning Representation*, 2013.

[14] X. Liu, Y. Liu, and X. Li, "Exploring the context of locations for personalized location recommendations," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016, pp. 1188–1194.

[15] H. Wang and Z. Li, "Region representation learning via mobility flow," in *Proceedings of the 26th ACM International Conference on Information and Knowledge Management*, 2017, pp. 237–246.

[16] S. Zhao, T. Zhao, I. King, and M. R. Lyu, "Geo-Teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation," in *Proceedings of the 26th International Conference on World Wide Web Companion*, 2017, pp. 153–162.

[17] Y. Zhou and Y. Huang, "DeepMove: Learning place representations through large scale movement data," in *Proceedings of the 6th IEEE International Conference on Big Data*, 2018, pp. 2403–2412.

[18] S. Feng, G. Cong, B. An, and Y. M. Chee, "POI2Vec: Geographical latent representation for predicting future visitors." in *Proceedings of the 31th AAAI Conference on Artificial Intelligence*, 2017, pp. 102–108.

[19] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model." in *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics*, 2005, pp. 246–252.

[20] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 1081–1088.

[21] H. Wan, F. Li, S. Guo, Z. Cao, and Y. Lin, "Learning time-aware distributed representations of locations from spatio-temporal trajectories," in *Proceedings of the 24th International Conference on Database Systems for Advanced Applications*, 2019, pp. 268–272.

[22] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1459–1468.

[23] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, pp. 2227–2237.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.

[25] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 5754–5764.

[26] D. Tang, B. Qin, Y. Yang, and Y. Yang, "User modeling with neural network for review rating prediction," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015, pp. 1340–1346.

[27] D. Tang, B. Qin, and T. Liu, "Learning semantic representations of users and products for document level sentiment classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, 2015, pp. 1014–1023.

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

[29] Y.-S. Lu, W.-Y. Shih, H.-Y. Gau, K.-C. Chung, and J.-L. Huang, "On successive point-of-interest recommendation," *World Wide Web: Internet and Web Information Systems*, vol. 22, no. 3, pp. 1151–1173, 2019.

[30] B. Chang, Y. Park, D. Park, S. Kim, and J. Kang, "Content-aware hierarchical point-of-interest embedding model for successive POI recommendation." in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3301–3307.

[31] H. Cao, F. Xu, J. Sankaranarayanan, Y. Li, and H. Samet, "Habit2vec: Trajectory semantic embedding for living pattern

recognition in population," *IEEE Transactions on Mobile Computing*, vol. 19, no. 5, pp. 1096–1108, 2019.

[32] J. Yang and C. Eickhoff, "Unsupervised learning of parsimonious general-purpose embeddings for user and location modeling," *ACM Transactions on Information Systems*, vol. 36, no. 3, pp. 1–33, 2018.

[33] A. Pauls and D. Klein, "Faster and smaller N-gram language models," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 258–267.

[34] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 40, no. 3, pp. 52–74, 2017.

[35] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proceedings of the 13th AAAI conference on artificial intelligence*, 2016, pp. 194–200.

[36] Q. Guo, Z. Sun, J. Zhang, and Y.-L. Theng, "An attentional recurrent neural network for personalized next location recommendation," in *Proceedings of the 17th AAAI Conference on Artificial Intelligence*, 2020, pp. 83–90.

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the 25th IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[38] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *arXiv preprint arXiv:2003.08271*, 2020.

[39] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[40] J. Du, Y. Zhang, P. Wang, J. Leopold, and Y. Fu, "Beyond geo-first law: Learning spatial representations via integrated autocorrelations and complementarity," in *Proceedings of the 19th IEEE International Conference on Data Mining*, 2019, pp. 160–169.

[41] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[42] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31th International Conference on Machine Learning*, 2014, pp. 1188–1196.

[43] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[44] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[45] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 3104–3112.

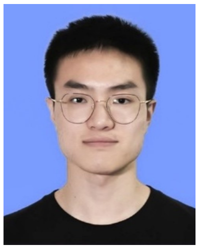
[46] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[48] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, pp. 2579–2605, 2008.

Huaiyu Wan received the Ph.D. degree in computer science and technology from Beijing Jiaotong University, Beijing, China, in 2012.

He is an Associate Professor with the School of Computer and Information Technology, Beijing Jiaotong University. His current research interests focus on spatial-temporal data mining, social network mining, information extraction, and knowledge graph.



Yan Lin received the B.S. degree in computer science from Beijing Jiaotong University, Beijing, China, in 2019.

He is currently working toward the Ph.D. degree in the School of Computer and Information Technology, Beijing Jiaotong University. His research interests include spatial-temporal data mining and graph neural networks.

Shengnan Guo received the B.S. degree in computer science from Beijing Jiaotong University, Beijing, China, in 2015.

She is currently working toward the Ph.D. degree in the School of Computer and Information Technology, Beijing Jiaotong University. Her research interests focus on the area of deep learning and spatial-temporal data mining.

Youfang Lin received the Ph.D. degree in signal and information processing from Beijing Jiaotong University, Beijing, China, in 2003.

He is a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. His main fields of expertise and current research interests include big data technology, intelligent systems, complex networks, and traffic data mining.