Representation Learning with Multi-level Attention for Activity Trajectory Similarity Computation

An Liu, Yifan Zhang, Xiangliang Zhang, Guanfeng Liu, Yanan Zhang, Zhixu Li, Lei Zhao, Qing Li, and Xiaofang Zhou, *Fellow, IEEE*

Abstract—Massive trajectory data stem from the prevalence of equipment-supporting GPS and wireless communication technology. Especially, activity trajectory from LBSN (Location-based Social Network) endows traditional trajectory data with additional user semantic activities, e.g., visiting work/home/entertainment places. Measuring the similarity between activity trajectories is to compare their proximity in multiple dimensions such as time, location, and semantics. In this way, we can mine implicit user preference and apply it to route planning, POI recommendation or any other online tasks. The key challenge of comparing activity trajectories (i.e., computing their similarity) lies in two aspects. One is the uneven sampling rate in both time and space. The other is the discrepancy of individual activities. Previous effort alleviates the issue of uneven sampling rate via trajectory complements, which is limited to spatial-temporal information. In this paper, we propose to learn a representation for one activity trajectory by jointly considering the spatio-temporal characteristics and the activity semantics. The similarity of two trajectories is computed by weighting individual trajectory points and contextual features with multi-level attention mechanisms. In specific, we propose a point-level and feature-level attention mechanism to adaptively select critical elements and contextual factors for learning trajectory representation. Our proposed approach, called At2vec, demonstrates better performance than existing baselines in extensive experimental evaluation on real trajectory databases.

Index Terms—Trajectory similarity, representation learning, activity trajectory, attention mechanisms.

1 INTRODUCTION

ADVANCES in sensor technology have driven GPS devices to be capable of recording increasing valuable location information. *Trajectory*, a time-ordered sequence of visited locations, is the most popular type of data studied for many applications including vehicle routing, path selection, and spatial crowdsourcing [1], [2], [3], [4]. Measuring the *similarity* of two trajectories is the most fundamental operation in many applications, such as trajectory outlier detection [5], trajectory clustering [6], trajectory encryption [7]. Due to the importance of this problem, researchers have proposed advanced similarity computation approaches based on the matching of trajectory points, including LCSS [8], and DTW [9]. Nevertheless, these approaches restrict their

applications to trajectories that have even sampling rate. In reality, trajectories often include location points that are recorded non-uniformly w.r.t. time, due to device failures or most likely intentional down time [10]. For example, location service is often switched off by mobile users due to privacy concerns. In these cases, the point-matching methods from prior work (LCSS and DTW) cannot appropriately measure the trajectory similarity because of this called *trajectory heterogeneity* issue.

The main approaches addressing the heterogeneity of trajectories follow the core idea of trajectory modification and completion. The EDwP [11] proposed by Ranu fills low-sampling trajectory through projection and coverage. Su et al. [12] employ HMM model to get anchor points learning from historical data and establish reference system to align or complete trajectory by anchor points. Li et al. [13] further apply deep learning models and achieved the state-of-the-art result. The common intrinsic property of these approaches is to unify the sampling rates of trajectories with spatio-temporal characteristics, so that location points on two trajectories are matchable. One example is illustrated in Fig. 1, where two trajectories *A* and *B* are spatially adjacent but trajectory *A* lacks point B_3 from *B* if matching them point by point. To run point-matching similarity methods, we need to complete trajectory *A* with a point corresponding to B_3 , to unify the sampling frequency of two trajectories. Suppose there are two candidate points α_1 , α_2 , which have the same distance to B_3 . Without any other information in addition to the trajectory location points, it is difficult to determine which one to choose for completion *A*. Even more noteworthy, the selected completion point

- A. Liu, Y. Zhang, Y. Zhang, and L. Zhao are with the School of Computer Science and Technology, Soochow University, Jiangsu 215000, China. E-mail: anliu@suda.edu.cn, {20175227050, 20175227082}@stu.suda.edu.cn, zhaol@suda.edu.cn.
- X. Zhang is with the King Abdullah University of Science and Technology, Jeddah, Thuwal 23955-6900, Saudi Arabia. E-mail: xiangliang.zhang@kaust.edu.sa.
- G. Liu is with the Department of Computing, Macquarie University, Sydney, Australia. E-mail: guanfeng.liu@mq.edu.au.
- Z. Li is with the School of Computer Science and Technology, Soochow University, Jiangsu 215000, China. He is also with the iFLYTEK Research, Suzhou, China and the State Key Laboratory of Cognitive Intelligence, iFLYTEK, P.R. China. E-mail: zhixuli@suda.edu.cn
- Q. Li is with the Department of Computing, the Hong Kong Polytechnic University, Hong Kong, China. E-mail: qing-prof.li@polyu.edu.hk.
- X. Zhou is with the School of Information Technology and Electrical Engineering, the University of Queensland, Brisbane QLD 4072, Australia. E-mail: zxf@itee.uq.edu.au.

Manuscript received September 8, 2019;
(Corresponding author: Zhixu Li.)

Fig. 1: Illustration of the limitation of trajectory completion when computing the similarity of trajectory A and B

has significant impact on calculating the similarity between trajectories A and B.

With the development of LBSN (Location-based Social Network), trajectories have a new augmented “activity” feature. In such a kind of *activity trajectory*, each location point is associated with a keyword, describing the occasion of a performed activity, e.g., *hotel*, *garden*, *school*, *shop* in parentheses next to each trajectory location point shown in Fig. 1. These keywords provide valuable additional information for trajectory completion. For example, α_2 point is more likely to be chosen than α_1 because the keywords of α_2 (Bank) and B_3 (ATM) are more similar in semantics. Therefore, the activity trajectory similarity should be measured by **integrating the spatio-temporal characteristics of location points and their additional keyword information.**

Our recent work in [14] is the first attempt of addressing this problem of activity trajectory similarity computation. Despite its effectiveness, the model in [14] has several limitations. First, it doesn’t consider the importance of different trajectory features (spatial, temporal, activity information) in the similarity measure. For example, the spatial characteristics of a wavy trajectory play a more important role than other features because a wavy trajectory has a larger spatial conversion between each point than a flat one. Second, it doesn’t highlight the key points in trajectories, such as the inflection points, in the training progress. The inflection points symbolize the change of the trajectory direction. In other words, these points can reflect the general shape of the trajectory. Ignoring such key inflection points will result in a decrease in the accuracy of the similarity computation, due to the neglect of the shape of the trajectory. Last, it used RNN, which is unable to capture long-term dependencies in long sequences, and brings about the loss of massive context relationships in trajectory representation.

We address the above-mentioned limitations by proposing a new model with a multi-level attention mechanism, and replacing RNN by LSTM in this work. Attention mechanism has recently proven its effectiveness in many fields [15], [16], [17], [18]. The core goal of attention mechanism is to select the most relevant part of information instead of using all. This naturally meets the requirement that trajectory similarity calculation collects contributions at different level from points and features. Moreover, we combine LSTM with another attention mechanism to compensate for the inability of the original model to handle long sequences.

One important merit of our proposed model is that it compares two trajectories in a semantic mapping space,

with no requirement on trajectory completion. Specifically, it maps each trajectory to a vector by integrating the spatio-temporal and activity information. Our proposed model is thus called At2vec (Activity trajectory to vector). The learned semantic vectors can be compared by simple metrics like dot product for computing the trajectory similarity. Although exempted from the tricky completion process, the trajectory comparison still faces the high uncertainty of low-sampling trajectory. Suppose a low-sampling trajectory is a subsequence of its complete high-sampling trajectory. The representation learned from the low-sampling trajectory is thus expected to be able to recover the high-sampling trajectory. Therefore, we take advantage of the suitable Seq2Seq model, which is designed for mapping arbitrary-length sequences to other arbitrary-length sequences, and has been successfully used for machine translation and image captioning [19]. By giving a low-sampling trajectory as input, and a high-sampling trajectory as desired output, we train the model to extract a representation vector from the low-sampling trajectory (encoding), based on which to reconstruct the corresponding high-sampling trajectory (decoding). With the minimized reconstruction error, the learned representation best preserves the trajectory information at the abstracted semantic level, and will thus not be affected by a wrong choice for completion.

We summarize our contributions as follows:

- To the best of our knowledge, this is the first study of adopting the activity information for trajectory similarity measure. The activity keywords enrich the trajectory information and promotes the similarity measurement with the understanding of user movement purpose, as shown in Fig. 1. Such trajectory similarity at the semantic level can be applied to various online applications such as route planning, and POI recommendation.
- We propose an At2vec model to learn the representation of the activity trajectory jointly based on the spatio-temporal characteristics and activity keywords of location points in trajectories. The uneven sampling issue is addressed by training an autoencoder deep learning model to map trajectories as semantic representation vectors in the same dimension.
- To capture critical elements among all available trajectory points and features, we integrate point-level and feature-level attention mechanisms into the At2vec model. Furthermore, an LSTM model with another attention mechanism is employed due to its better capability on learning long-term dependencies from trajectories than RNN architecture.
- Experimental results on real trajectory datasets demonstrate that the At2vec model outperforms existing methods on trajectory similarity analysis in the matter of both efficiency and effectiveness.

The rest of this paper is organized as follows. Section 2 introduces the problem statement and some background knowledge of deep learning models. The details of our model are presented in Section 3. We then perform experimental evaluation in Section 4. We discuss the related work in Section 5, and conclude the paper in Section 6.

TABLE 1: Summary of notations

Notation	Definition
cp	Check-in point
$\mathbb{T}$	Activity trajectory
V	Vocabulary
c	Final trajectory representation
$y_{1:t}$	A sequence with position from 1 to t
q_t^e, H_t, h_t	Cell state and hidden state for LSTM
$e(L, t), e(P, t)$	Spatio-temporal embedding vector
$E_p(i, \cdot), E_p(\cdot, j)$	Rows and columns of the matrix
LP, TP, AC	Trajectory feature embeddings for input
w_{lc}, w_{tp}, w_{ac}	Weights for loss function

Fig. 2: An example of a single activity trajectory

2 PRELIMINARIES

In this section, we first give the problem definition and then briefly introduce some deep learning models used in the paper. We list the major notations used throughout the paper in Table 1:

2.1 Problem definition

A check-in point is denoted by $cp = \langle L, P, M \rangle$ where L is the location point of the check-in and usually presented as $\langle \text{longitude}, \text{latitude} \rangle$, P is the timestamp of the check-in, and M is a keyword describing the activity occasion. An activity trajectory, denoted by $\mathbb{T}$, is composed of a sequence of check-in points, which can be formulated as $\mathbb{T} = \{cp_1, cp_2, \dots, cp_n\}$, as shown in Fig. 2.

Given an activity trajectory, our target is to learn a fixed-length vector representing the trajectory in a low-dimensional space that jointly captures the trajectory characteristics and activity keyword by their semantic meanings. The learned representation can be then adapted to the similarity calculation between the trajectories of different sampling rates.

2.2 Deep learning models

LSTM network. As shown in Fig. 3, a single LSTM cell obtains the output through the current input x_t and the previous hidden state h_{t-1} . Different from RNN network, LSTM enhances RNN by adding several gating units and cell states. The most significant change is the new cell state q_t , which is a special variable recording the current information in step t . Moreover, LSTM utilizes multiple gating units to determine how much data can be stored. Specifically, the forget gate f_t controls the current input x_t and the previous hidden state h_{t-1} via filtering out the unimportant

Fig. 3: Illustration of the LSTM network

Fig. 4: Illustration of the Seq2Seq model

elements and recording more valuable information. Similar to forget gate, input gate i_t and output gate o_t also perform linear transformation on x_t and h_{t-1} . Meanwhile, the significant cell states $\tilde{q}_t$ is generated through the tanh activation function instead of sigmoid function. According to the output of the forget gate and the input gate, the data with insufficient information value is further forgotten and the cell state value is updated to q_t . In the final stage, the output gate combines the cell unit state q_t and the output o_t to determine the hidden state h_t of the LSTM network under the time step t . In our work, we utilize LSTM as the basic network architecture for its excellent performance on handling long trajectory sequences. Meanwhile, LSTM can solve the vanishing gradient problem caused by the gradual reduction in the gradient back-propagation process.

Encoder-decoder model. The Seq2Seq model has an encoder-decoder framework mapping an input sequence to an output sequence, as illustrated in Fig. 4. Given two sequences x and y of different length $|x|$ and $|y|$. The model has an objective function to maximize:

$$\mathbb{P}(y|x) = \mathbb{P}(y_1|x) \prod_{t=2}^{|y|} \mathbb{P}(y_t|y_{1:t-1}, x) \quad (1)$$

As depicted in Fig. 4, the Seq2Seq model is composed of two components, encoder and decoder:

1) Encoder: The encoding process is relatively brief to use RNN [20] [21] directly, while we utilize LSTM in this paper, for semantic vector generation. The reason for using LSTM instead of RNN is the superiority of LSTM in handling long sequences. Assuming that the generated intermediate vector is c , the encoding procedure is as follows:

$$[h_t; q_t^e] = LSTM_e(x_t, h_{t-1}, q_{t-1}^e) \quad (2)$$

$$c = \phi(h_1, \dots, h_t) \quad (3)$$

where $LSTM_e$ is the LSTM cell of the encoder, h_{t-1} is the output of the previous hidden node, q_t^e is the cell states of

the encoder and x_t is the input of the current time. The vector c is normally the last hidden node in the RNN or the weighted sum of multiple hidden nodes.

2) Decoder: The decoding process of this model utilizes another LSTM to predict the current output symbol y_t through the current implicit state H_t , where both hidden state H_t and cell state q_t^d are related to its previous states and output of encoder c :

$$[H_t; q_t^d] = LSTM_d([c, y_{t-1}], H_{t-1}, q_{t-1}^d) \quad (4)$$

With the encoder-decoder model, we can rewrite the conditional probability in Eq. (1) as:

$$\mathbb{P}(y_t|y_{1:t-1}, x) = \mathbb{P}(y_t|y_{1:t-1}, c) = \mathbb{P}(y_t|H_t) \quad (5)$$

$$\mathbb{P}(y_t = j|H_t) = \frac{\exp(W_j^T H_t)}{\sum_{j' \in V} \exp(W_{j'}^T H_t)} \quad (6)$$

where H_t can be projected from the hidden state into the vocabulary space by the projection matrix W_j^T and j denotes a probable member of the vocabulary V .

3 OUR AT2VEC MODEL

3.1 The framework

The framework of our At2vec model is shown in Fig. 5. The model input is a low-sampling activity trajectory distorted from a high-sampling activity trajectory, which is the desired output. For both the low-sampling and high-sampling trajectory, they go through spatial, temporal and activity embedding layers, which are simplified as an embedding layer in the figure. Those obtained isometric feature vectors are then concatenated to get the representation vector of the trajectory. The vector of the low-sampling trajectory is then given as the input to the decoder, while the embedding vector of the original high-sampling trajectory is taken as the learning target of the decoder. In the At2vec model, we add two attentions to capture the importance of trajectory points and multiple trajectory features in the encoder part. When it comes to the decoder module, the corresponding basic attention mechanism is also applied. The obtained intermediate vector c is utilized as a final representation of the trajectory for future similarity computation. Detailed steps are described in the next sections.

3.2 Trajectory feature extraction

A trajectory is composed of check-in locations, timestamps and activity keywords. We first learn feature embedding of them as follows.

1) Spatial information embedding: The check-in locations are recorded as continuous spatial points. Since the input and output of the Seq2Seq model must be a set of discrete tokens, we first convert continuous trajectory points into discrete values. A discretization strategy commonly utilized in spatial-temporal data analysis is dividing the whole studied area into many equal-sized grid cells with discrete numbers. In this way, the spatial value of a trajectory point is converted into the ID of a grid cell where it belongs to. To learn the representation of these discrete spatial points,

we use the skip-gram model that maximizes the function below:

$$\frac{1}{N} \sum_{t=1}^N \sum_{-k \leq j \leq k, j \neq 0} \log \mathbb{P}(L_{t+j}|L_t) \quad (7)$$

where L_t is the current cell, and L_{t+j} is a neighboring cell of L_t . Two close neighboring cells should have similar representations. Thus, $\mathbb{P}(L_{t+j}|L_t)$ is calculated using the softmax function as follows:

$$\mathbb{P}(L_{t+j}|L_t) = \frac{\exp(v_{L_{t+j}}^T v_{L_t})}{\sum_{L_i \in L} \exp(v_{L_i}^T v_{L_t})} \quad (8)$$

Here, v_{L_t} denotes vector representation of a cell L_t . In this way, the raw spatial trajectory $\mathbb{T}_i^{spatial}$ is transformed to a spatial embedding LC_i :

$$\mathbb{T}_i^{spatial} \rightarrow LC_i = \{e(L, 1), \dots, e(L, t), \dots, e(L, T)\} \quad (9)$$

where $e(L, t)$ is the cell representation v_{L_t} learned above and T is the length of the trajectory.

Note that LC_i is only a part of the input for encoding the spatial information. We formulate temporal and activity information similarly as follows.

2) Temporal information embedding: The temporal information in a trajectory sequence is described by timestamps. We first discretize the timestamps by even time intervals, and then obtain a sequence of P . Similarly, we use skip-gram model to learn the embedding of each timestamp. The temporal part representation of the trajectory $\mathbb{T}_i^{temporal}$ is formed as TP_i :

$$\mathbb{T}_i^{temporal} \rightarrow TP_i = \{e(P, 1), \dots, e(P, t), \dots, e(P, T)\} \quad (10)$$

where $e(P, t)$ is the embedding vector of t -th timestamp in the trajectory $\mathbb{T}_i$, obtained from skip-gram model.

3) Activity information embedding: The activity keywords describe user behaviors when visiting different locations, and thus provide important additional information for trajectory comparison. We build a vocabulary set including all the appeared keywords. Then we turn the trajectory into a usable token sequence according to the position of each word in the vocabulary. We use the pre-trained GloVe word vectors to initialize each word in order to speed up the whole process, and eventually obtain the following representation for the activity part of a trajectory:

$$\mathbb{T}_i^{activity} \rightarrow AC_i = \{e(M, 1), \dots, e(M, t), \dots, e(M, T)\} \quad (11)$$

where $e(M, t)$ is the embedding vector of t -th activity in the trajectory, coming from GloVe pretrained word vectors.

3.3 At2vec model

After getting ready of the above three representations of one trajectory, we elaborate on the proposed At2vec model. As stated in Introduction, low-sampling trajectories have missed check-in points and result in high uncertainty in representation learning. Our goal is to align a low-sampling trajectory with its most similar high-sampling trajectory in the extracted semantic space, rather than completing with make-up location points. The alignment is implemented by mapping a low-sampling trajectory to a representation

Fig. 5: At2vec model with multi-level attention

vector, from which its most similar high-sampling trajectory can be reconstructed. This is exactly how the Seq2Seq model is designed based on autoencoder framework [22]. Instead of predicting the output sequence given an input sequence like in machine translation, we target on learning the representation vector of a low-sampling trajectory by minimizing the reconstruction error of its corresponding high-sampling trajectory.

To be specific, from a primitive trajectory $\mathbb{T}_i$, we first extract its spatio-temporal and activity representation LC_i , TP_i and AC_i , respectively. Then we concatenate the three representations to get $\mathbb{T}_i$ as the desired output of our At2vec model. Next, we sample from $\mathbb{T}_i$ to obtain a low-sampling representation $\mathbb{T}_i^{low}$, at a sampling rate p . By concatenating similarly the spatio-temporal and activity representations, we obtain $\mathbb{T}_i^{low}$ for the low-sampling trajectory. Then given $\mathbb{T}_i^{low}$ as input and $\mathbb{T}_i$ as the desired output, the overall objective function of At2vec is:

$$\text{maximize } \prod_{i=1}^N \mathbb{P}(\mathbb{T}_i | \mathbb{T}_i^{low}) \quad (12)$$

where

$$\mathbb{T}_i = \text{concat}(LC_i, TP_i, AC_i) \quad (13)$$

$$\mathbb{T}_i^{low} = \text{concat}(LC_i^{low}, TP_i^{low}, AC_i^{low}) \quad (14)$$

3.4 Multi-level attention mechanisms

At2vec addresses the trajectory heterogeneity by training an effective model to map one activity trajectory to a semantic representation vector. However, one remaining issue is to distinguish the importance of individual trajectory point and various features. For example, inflection points in a trajectory are significantly less than the non-inflection points.

Therefore, these non-inflection points dominant the model training process and also the result of representation. Trajectories sharing non-inflection points become indistinguishable even they differ on few important points. Therefore, we propose multi-level attention mechanisms for At2vec model to solve these problems. The basic principle is to extract the most relevant information instead of using all for analysis. For an activity trajectory, the aim is to assign weights to points or features in such a way that more important points/features receive higher weights. The effectiveness of this mechanism can be justified by the experimental results on the attention weights of different kinds of points, which will be discussed in Section 4.5.

We first propose hierarchical attention from two aspects: *point-level* and *feature-level*. The *point-level* attention is applied to trajectory location points. The *feature-level* attention is applied to contextual features. In addition, a *encoder-level* attention is introduced to alleviate the context length limitation problem.

1) Point-level Attention: This module measures the importance of each check-in location point in a trajectory. Intuitively, an inflection point in a trajectory is more important than a normal point. From equation (13) in the previous section, LC_i , TP_i , AC_i are matrices with the same size $d * T$, where T is the trajectory length (the number of location points in the trajectory) and d is the dimension of the latent feature factors. After concatenating these three matrices by column, a trajectory input is formalized as a numerical matrix $E \in \mathbb{R}^{3d \times T}$. To simplify the notation, we denote $N = 3d$. Therefore, the input to the model is a matrix $E \in \mathbb{R}^{N \times T}$ of N rows (N features) and T columns (T trajectory points).

In the point-level attention module, we extract the j -th

column of the trajectory matrix E to represent j -th trajectory point, by treating three features as a union, noted as $E(\cdot, j)$. We calculate the point-level attention score at step t based on the previous hidden state h_{t-1} and cell state q_{t-1}^e of the encoder, and then normalize the attention weight by a softmax function:

$$a_{tj} = v_1^T \tanh(W^{(1)}[h_{t-1}; q_{t-1}^e] + W^{(2)}E(\cdot, j) + b_1) \quad (15)$$

$$\alpha_{tj} = \frac{\exp(a_{tj})}{[\sum_{k=1}^T \exp(a_{tk})]^{1/t}}, 1 \leq j \leq T \quad (16)$$

where $W^{(1)} \in \mathbb{R}^{N \times 2N_e}$, $W^{(2)} \in \mathbb{R}^{N \times N}$, $b_1, v_1 \in \mathbb{R}^N$ are parameters to be learned. Here we modify the traditional softmax function by adding an exponent of $1/t$ to the denominator. With the growth of the sequence length, the value of the denominator will be suppressed. Via this way, the attention score will increase from the interval $[0, 1]$ to $[1, +\infty]$, which can solve the problem of matrix decreasing. According to the attention score α_{tj} and the original trajectory sequence, we get n -th row of the attentive embedding matrix:

$$E_p(n, \cdot) = \{E(n, 1)\alpha_{t1}, E(n, 2)\alpha_{t2}, \dots, E(n, T)\alpha_{tT}\} \quad (17)$$

Thus, the point-level attentive embedding matrix E_p can be formulated as:

$$E_p = \{E(1, \cdot), \dots, E(n, \cdot), \dots, E(N, \cdot)\} \quad (18)$$

2) Feature-level Attention: To measure the importance of different trajectory feature factors, which are spatial-temporal features and activity keywords in our paper, we extract the i -th row of the trajectory matrix E_p to represent the i -th trajectory feature, $E_p(i, \cdot)$. Similar to the point-level attention, the feature-level attention scores at step t are calculated based on the state from last time step and rows of feature matrix. We also use the softmax function for normalization:

$$a_{it} = v_2^T \tanh(W^{(3)}[h_{t-1}; q_{t-1}^e] + W^{(4)}E_p(i, \cdot) + b_2) \quad (19)$$

$$\alpha_{it} = \frac{\exp(a_{it})}{[\sum_{k=1}^N \exp(a_{kt})]^{1/t}}, 1 \leq i \leq N \quad (20)$$

where $W^{(3)} \in \mathbb{R}^{T \times 2N_e}$, $W^{(4)} \in \mathbb{R}^{T \times T}$, $b_2, v_2 \in \mathbb{R}^T$ are parameters to be learned. Based on the point-level attentive embedding vector obtained previously, we multiply it by feature-level attention score to get τ -th col of the input matrix E_f :

$$E_f(\cdot, \tau) = \{E_p(1, \tau)\alpha_{1t}, E_p(2, \tau)\alpha_{2t}, \dots, E_p(N, \tau)\alpha_{Nt}\} \quad (21)$$

The final input matrix E_f can be represented as:

$$\hat{x}_t = E_f = \{E_f(\cdot, 1), \dots, E_f(\cdot, \tau), \dots, E_f(\cdot, T)\} \quad (22)$$

3) Encoder-level Attention: As shown previously, we represent the trajectory by the intermediate vector c from encoder to decoder. In order to get the intermediate vector, we also add another attention mechanism named *encoder-level attention*. The function of encoder-level attention is mainly to weight the output of the encoder module, which makes the intermediate vector c able to distinguish the importance of hidden states. Different from the above two methods, the

calculation of attention score is relevant to the decoder last cell state q_{t-1}^d , hidden states H_{t-1} and the encoder hidden states h_l , which can be written as follows.

$$b_t^l = v_3^T \tanh(W^{(5)}[H_{t-1}; q_{t-1}^d] + W^{(6)}h_l + b_3) \quad (23)$$

$$\beta_t^l = \frac{\exp(b_t^l)}{[\sum_{k=1}^T \exp(b_t^k)]^{1/t}}, 1 \leq l \leq T \quad (24)$$

where $W^{(5)} \in \mathbb{R}^{N_p \times 2N_d}$, $W^{(6)} \in \mathbb{R}^{N_p \times N_e}$, $b_3, v_3 \in \mathbb{R}^{N_p}$ are parameters to be learned. N_e, N_d, N_p are the dimensions of the hidden states of encoder, decoder and intermediate vector respectively. The decoder output c_t is derived from the product of the attention score and the encoder hidden states as follow:

$$c_t = \sum_{l=1}^T \beta_t^l h_l \quad (25)$$

We generate the final trajectory representation by combining the output c_t of all time steps. Actually, the dimension of the final representation vector equals to sequence length of the encoder module:

$$c = \overbrace{\{c_1, c_2, \dots, c_t \dots\}}^{\text{sequence length}} \quad (26)$$

After obtaining final trajectory representation, we calculate the similarity between the original trajectories according to the similarity between these intermediate vectors.

3.5 STA aware Loss function

Loss function is the core of neural network, which determines the direction and ultimate goal of network training. Among all cells, we need to show the semantic proximity of each cell in the loss. In this case, we propose a novel STA (spatial-temporal-activity) aware loss function to capture the spatial-temporal and activity information relationship between the cells.

In the field of natural language processing, the loss function of Seq2Seq model is usually defined as:

$$\mathcal{L}_1 = -\log \prod_t \mathbb{P}(y_t | y_{1:t-1}, x) \quad (27)$$

where $y_t, y_{1:t-1}, x$ are the same variables in Eq.(1). In our case, the encoder-decoder model targets on predicting the conditional probability of a cell in the vocabulary, denoted as L_i in Eq.(7). The loss function will be adjusted to:

$$\mathcal{L}_1 = -\log \prod_t \prod_i \mathbb{P}(y_t = L_i | y_{1:t-1}, x) \quad (28)$$

Eq.(6) provides a way to calculate conditional probabilities above. However, for our studied problem, the geographic location between the prediction and the ground truth should be considered in the loss function. Also, the loss function should take into account the degree of difference refers to spatial, temporal or semantic aspects between the prediction and the ground truth. Therefore, we introduce weights for each cell, measuring the spatial, temporal or semantic

difference between the predicted cell L_i and the ground truth cell $\tilde{L}_i$ of y_t . The new loss STA-aware function is

$$\begin{aligned} \mathcal{L}_2 = & - \sum_{t=1}^T \left(\sum_{L_i^{lc} \in V_{lc}} w_{lc} \log \mathbb{P}(y_t = L_i^{lc} | y_{1:t-1}, x_{lc}) \right. \\ & + \sum_{L_i^{tp} \in V_{tp}} w_{tp} \log \mathbb{P}(y_t = L_i^{tp} | y_{1:t-1}, x_{tp}) \\ & \left. + \sum_{L_i^{ac} \in V_{ac}} w_{ac} \log \mathbb{P}(y_t = L_i^{ac} | y_{1:t-1}, x_{ac}) \right) \end{aligned} \quad (29)$$

Here the weights are

$$w_{lc} = \frac{\exp(-\|v_{L_i^{lc}} - v_{\tilde{L}_i^{lc}}\|_2 / \alpha)}{\sum_{L_j \in V_{lc}} \exp(-\|v_{L_j} - v_{\tilde{L}_i^{lc}}\|_2 / \alpha)} \quad (30)$$

$$w_{tp} = \frac{\exp(-\|v_{L_i^{tp}} - v_{\tilde{L}_i^{tp}}\|_2 / \beta)}{\sum_{L_j \in V_{tp}} \exp(-\|v_{L_j} - v_{\tilde{L}_i^{tp}}\|_2 / \beta)} \quad (31)$$

$$w_{ac} = \frac{\exp(-\cos(v_{L_i^{ac}}, v_{\tilde{L}_i^{ac}}) / \gamma)}{\sum_{L_j \in V_{ac}} \exp(-\cos(v_{L_j}, v_{\tilde{L}_i^{ac}}) / \gamma)} \quad (32)$$

where v_{L_i} and $v_{\tilde{L}_i}$ are vector representation of a predicted cell L_i and the ground truth cell $\tilde{L}_i$, respectively. The spatial and temporal weights are defined by applying Euclidean distance on the representation vectors, while the activity weight uses cosine distance because it involves text similarity and our experiment also shows that cosine distance is the best choice. In general, the weight of the predicted cell $y_t = v_{L_i}$ is inversely proportional to its physical distance to the true target cell $\tilde{L}_i$. A larger weight is attached to a predicted cell that is close to the true target. Parameter α , β , γ are three smoothing factors for adapting to different influencing factors. When three factors simultaneously tend to 0, the loss function becomes $\mathcal{L}_1$. To reduce the computational cost without scarifying the performance, we utilize the k nearest neighbors of the target cell $\tilde{L}_i$, rather than all cells in vocabulary in our experiment. That is to say, the V of the above formula is replaced by $N_k(\tilde{L}_i)$, whose size is smaller than V but collects the most relevant neighbors.

4 EXPERIMENTS

In this section, we evaluate the accuracy and effectiveness of At2vec. We use multiple trajectory similarity evaluation criteria to carry out experiments and analyze the impact of various parameters on the experimental results. All the methods are implemented in PyTorch on a computer with GTX 1050Ti GPU and Intel i7 7700K CPU.

4.1 Dataset

We use two real activity trajectory datasets by crawling from Foursquare within the areas of Los Angeles (LA) and New York City (NYC). Each record of the datasets contains the user ID, venue with POI, time of check-in, and tips written in English. The details are shown in Table 2.

TABLE 2: Dataset

Dataset	NYC	LA
#POIs	2165786	3526190
#trajectories	29521	34325
#keywords per point	3.25	2.63

TABLE 3: Parameter Settings

Parameters	Value
learning rate	0.001
hidden layer size	256
dropout rate	0.1
maximum gradient norm	5
batch size	64

4.2 Hyper parameters setting

1) Spatial cell size: In the process of discretizing the original spatial values, we divide the whole city into $300m \times 300m$ grid cells following the work in [23]. For the LA dataset, we get 12489 cells and for NYC we get 9856 cells.

2) Time period size: Most valid check-in intervals are small, so we use 10 minutes to segment the whole time span (2014-1 to 2014-5) and obtain the vocabulary of time. The vocabulary position of each time point is used for initializing its temporal representation. Then the temporal representations of time points are learned by skip-gram algorithm.

3) Parameters of neural network: Table 3 lists all the parameters we use for training Seq2Seq model. These values were determined by grid search due to their better performance than other settings.

4.3 Performance metrics

By referring to literature, we analyze the effectiveness of our model in terms of the following three metrics:

1) Self-similarity measure (SSM): Following the evaluation in [13], this metric is to measure the capability of the model in recognizing two sub-trajectories sampled from the same trajectory. We extract 2000 trajectories, denoted as D_1 , from the test dataset as the query. Afterwards, we randomly choose from the test set to have 200 trajectories, denoted as D_2 with no overlap with D_1 . From every trajectory in D_1 , we intercept two sub-trajectories forming two sets $D_1^a = \{T_i^a\}$ and $D_1^b = \{T_i^b\}$, respectively. Then, we construct a target dataset D , which consists of D_1^b and D_2 (taken as noise): $D = D_1^b \cup D_2$. The evaluation target is to retrieve the top- k trajectories most similar to T_i^a (from D_1^a) in D . It is expected that T_i^b should be ranked at the top (higher than those noisy trajectories D_2 in D), because T_i^a and T_i^b come from the same original trajectory. In our experiments, we define the value of SSM as average ranking value for all trajectories in query dataset D_1^a .

2) Cross-distance (CSD): This is a criterion for evaluating the robustness of the model to trajectory variants [12]. Trajectory variants can be obtained via dropping or distorting some certain location points (at a given the dropping rate) in the original trajectory. For each evaluated model, we check whether the original distance of two different trajectories

can be maintained regardless of the variation done to the two trajectories. That is to say, T_A and T_B are two different trajectories with original distance $d(T_A, T_B)$. Their variants are $V(T_A)$ and $V(T_B)$ with the distance $d(V(T_A), V(T_B))$. For each evaluation method, we take $d(T_A, T_B)$ as the ground-truth and measure the degree of its difference from $d(V(T_A), V(T_B))$, defined as

$$dev(T_A, T_B) = \frac{|d(V(T_A), V(T_B)) - d(T_A, T_B)|}{d(T_A, T_B)} \quad (33)$$

We can see that a smaller cross-distance means that the influence of the trajectory variants on the evaluated model is slighter.

3) KNN queries (KNN): This evaluation is to measure how each model is robust to the KNN query regarding trajectory variations. In our experiment, 500 trajectories are used as queries, denoted as D^{query} and 5,000 trajectories form a target database, denoted as D^{target} . There is no intersection between D^{query} and D^{target} . For each query trajectory in D^{query} , we find its k nearest neighbors KNN (original) in the D^{target} database as ground-truth. Similar to the CSD method, we take trajectory variants and discover its KNN trajectories KNN (variants) in the database. Finally, we calculate the proportion of KNN (variants) in the ground-truth KNN (original) as the precision of this method.

4.4 Model comparison

4.4.1 At2vec model with comparative baselines

To illustrate the effectiveness of the At2vec model, we compare it with several existing models using dynamic programming and deep learning.

1) LCSS [8]: LCSS adopts the idea of dynamic programming to find the longest common substring, which is very effective in solving the problem of massive noise points in trajectory.

2) EDR [24]: Due to the fact that LCSS cannot distinguish the trajectories with the same common substrings, the EDR distance is raised.

3) EDwP [11]: Edit Distance with Projections (EDwP) is proposed to match trajectories under inconsistent and variable sampling rates through dynamic interpolation.

4) APM [12]: Anchor Points based Method (APM) learns the anchor point based on the historical data combined with the HMM model, and unifies the sampling rate of trajectories through a reference system composed of anchor points.

5) t2vec [13]: The t2vec settles the problem of low sampling through the deep learning model, which is also the state of the art trajectory similarity computation method

6) At2vec [14]: The At2vec is our original model based on RNN network and without multi-level attention mechanism.

7) At2vec-Sem: To realize a fair comparison with models that not considered semantic information, we tailor the At2vec model by removing the associated keyword information from the inputted activity trajectory.

Performance Comparison: In the following, we demonstrate the experimental results on self-similarity measure, cross-distance and KNN queries, denoted as SSM, CSD and

Fig. 6: At2vec with comparative baselines: evaluation comparison of SSM on NYC and LA dataset

Fig. 7: At2vec with comparative baselines: evaluation comparison of CSD on NYC and LA dataset

KNN respectively. Fig. 6 illustrates the results of different models in various dropping rate, which refers to the proportion of discarded points when generating trajectory variants.

Note that in CSD and KNN, we skip the baselines LCSS and EDR since they ignore the problem of trajectory heterogeneity. From these figures, we discover that At2vec model achieves better performance than all the baselines.

In addition, from Fig. 6(a) and Fig. 6(b), we can see that t2vec performs the best among all the baselines. Both LCSS and EDR perform poorly for the ignorance of trajectory heterogeneity problem. The At2vec model outperforms these existing methods on SSM evaluation criterion and the effectiveness of our method is more obvious with the increase of dropping rate.

When it comes to CSD (as shown in Fig. 7(a) and Fig. 7(b)), we remove the baselines (LCSS, EDR), which ignore the problem of different trajectory sampling strategies for their poor performance. From the figures, we can see that our At2vec model has been superior to the state of the art methods. With the increase of the dropping rate,

Fig. 8: At2vec with comparative baselines: evaluation comparison of KNN@20 on NYC and LA dataset

the influence of trajectory heterogeneity on other baseline methods keep increasing. However, the results of At2vec model is basically in a stable state. Since the model is less affected by the dropping rate, the problem of different trajectory sampling strategies can be solved.

Fig. 9: At2vec with comparative baselines: evaluation comparison of KNN@30 on NYC and LA dataset

Fig. 10: At2vec with comparative baselines: evaluation comparison of KNN@40 on NYC and LA dataset

The results of KNN queries are shown in Fig. 8, Fig. 9 and Fig. 10, with different settings of k . Compared to APM and EDwP, our approach has improved a lot owing to the utilization of deep learning model. The effect of our At2vec method is slightly better than t2vec since the clustering method mainly refers to the spatio-temporal features of the trajectory. As the dropping rate increases, the accuracy of all methods decreases.

The results in figures 6-10 also show that, without the help of semantic information, At2vec-Sem cannot beat t2vec, which is also based on deep learning models. This just demonstrates the importance of semantic information for activity trajectory similarity computation. That is why At2vec outperforms existing methods.

What's more, we discover that our model is not as effective on the LA dataset as it is in the NYC dataset. After analyzing the two datasets (as shown in Table 2), we learn that the LA dataset has more trajectory points while each point contains fewer keywords attached. In our model, richer textual information is needed, so the results of LA dataset are not as good as that of the NYC dataset.

4.4.2 At2vec model against At2vec+Atten model

After showing At2vec is better than existing methods, we now demonstrate that it can be further enhanced by the proposed multi-level attention mechanisms. Let **At2vec+Atten** denote the At2vec model with multi-level attention mechanisms. Since it performs a series of optimizations over

At2vec, we adopt the same datasets and experimental settings described earlier to evaluate whether the new model can go further on the previous basis.

Fig. 11: At2vec against At2vec+Atten: evaluation comparison of SSM on NYC and LA dataset

Fig. 12: At2vec against At2vec+Atten: evaluation comparison of CSD on NYC and LA dataset

Performance Comparison: As shown in Fig. 11, the SSM indicator of At2vec has been greatly improved when three attention mechanisms are added. In the best case, At2vec+Atten achieves an improvement of 28.1% and 36.8% over At2vec in terms of SSM on NYC and LA dataset, respectively. As the dropping rate increases, the problem of trajectory heterogeneity becomes more serious but the A2tec+Atten model maintains good robustness to low-sampling trajectories.

Fig. 12 shows the performance of A2tec+Atten on cross distance (CSD). As mentioned above, a smaller cross distance means a better performance in solving trajectory heterogeneity. From the figure, we can see that A2tec+Atten is superior to At2vec thanks to the optimization made by the multi-level attention mechanisms.

Fig. 13 to Fig. 15 illustrate the performance of the two models in KNN queries with different K values. It is clear from these figures that the A2tec+Atten model is superior to At2vec in various situations since the latter ignores the importance of points and features. As the value K increases, both models perform worse, but At2vec+Atten can consistently achieve better results.

4.5 Importance of inflection points

From the performance study reported above, we can see that multi-level attention mechanisms can effectively enhance the At2vec model. In order to verify whether different types of trajectory points actually get unequal attention weights, we calculate the sum of attention weight of inflection points and non-inflection points for each activity trajectory in a

Fig. 13: At2vec against At2vec+Atten: evaluation comparison of KNN@20 on NYC and LA dataset

Fig. 14: At2vec against At2vec+Atten: evaluation comparison of KNN@30 on NYC and LA dataset

real-world dataset, and report the result in Fig. 16. Clearly, for the same trajectory, the sum of the attention weights obtained by the inflection points is much larger than that of the non-inflection points, which again validates the motivation of using multi-level attention mechanisms in activity trajectory similarity computation.

Fig. 15: At2vec against At2vec+Atten: evaluation comparison of KNN@40 on NYC and LA dataset

Fig. 16: Attention weights of inflection points and non-inflection points in test datasets

Fig. 17: Performance of STA-aware loss function.

Fig. 18: Impact of the network structure.

4.6 Performance of STA-aware loss function

The improvement of the loss function is a key point of the work of this paper, which takes the proximity of semantic into account. To demonstrate the effectiveness of our proposed loss function, we evaluate the resulting loss magnitude at different sampling frequencies.

As shown in Fig. 17, $\mathcal{L}_1$ is the loss in the original Seq2Seq model and $\mathcal{L}_2$ is the new one we propose. We make a comparative analysis on both datasets and find that the experimental results using the loss function proposed in this paper are much better than the original NLL loss function. With the decrease of sampling rate, the final loss value of training will increase correspondingly. It can be vividly seen from the figure that our loss function eventually reaches a smaller loss value. The main reason for this result is that we utilize additional activity information when the trajectory is completed through the deep network, which makes the previously mis-corrected or missed points be added correctly.

4.7 Impact of the network structure

In order to compensate for the original model's shortcomings in the extraction of long sequence information, we replace the RNN network with an LSTM network, which can learn long-term dependencies in training process. Based on this, we also make related experiments to discuss the effectiveness of the LSTM network.

Specifically, the comparative experimental results on both datasets are shown in Fig. 18. From the figure, we can see the superiority of LSTM network structure compared to RNN structure. In addition, we can find that LSTM reaches a smaller value in the SSM evaluation criteria than RNN as the dropping rate increases.

Fig. 19: Effectiveness of multi-level attention mechanism.

4.8 Effectiveness of multi-level attention mechanism

Different from our previous work, multi-level attention mechanisms are added to enhance the At2vec model. To demonstrate the impact of different levels of attention mechanisms on experimental results, we evaluate models with five distinct attentions. As shown in Fig. 19, we simplify the point level, feature level, point with feature level and all levels as P-attention, F-attention, PF-attention and All-attention respectively.

Generally speaking, the addition of attention mechanism is better than the original model with no attention mechanism on both datasets. When more levels of attention mechanisms are added, the experimental results were correspondingly improved and the All-attention mechanism reaches the best results in our model. Interestingly though, the trends of P-attention and F-attention methods in Fig. 19 are constantly changing with the increase of the dropping rate. As can be seen from the figure, P-attention is better at a lower dropping rate while F-attention is more superior at a higher one. What causes this result is that massive noise points are in the trajectory when the dropping rate is very low. Therefore, the point-level attention mechanism can screen out more important trajectory points to generate trajectory representations. At this stage, the point-level attention mechanism plays a major role. However, the increase of dropping rate causes the number of trajectory points to decrease rapidly. At this stage, the selection of features becomes more important.

Furthermore, as seen in Fig. 19(a), F-attention curve exceeds P-attention curve in the position between 0.1 and 0.2 of X-axis. However, the position is from 0.3 to 0.4 of X-axis in Fig. 19(b). In other words, in the NYC dataset, F-attention dominates while the P-attention is more important in the LA dataset. We explain this phenomenon according to Table 2. As shown in the table, each trajectory in NYC dataset contains fewer points than that in LA dataset averagely. Therefore, it is difficult to find key points in the NYC dataset for its high sparsity of trajectories. In this case, we know that feature selection is more important in the NYC dataset while screening of key points plays a key role in LA dataset.

4.9 Hyper Parameters Study

In the following experiment, we fix the dropping rate to 0.5 and use the SSM method to get the corresponding average ranking, which can show the effect of the number of hidden layers and cell size. Meanwhile, we analyze the effect of cell size on the final mean rank. There are two kinds of

cells in our model, corresponding to temporal and spatial characteristics.

We do several sets of experiments on the size of the time period and the spatial cell (results shown in the Fig. 20). As can be seen from the figure, the training result is the best when the time period is equal to 10 min and the size of the space cell is 300 meters. As we know, the dimension of the hidden layer determines the quality of the learned representation. High dimension of the hidden layer is generally more efficient, but there is a greater demand for the amount of training data. It can be seen from the figure that it is optimal when the value is 256, and gradually decreases after that.

5 RELATED WORK

5.1 Trajectory similarity computation

Trajectory similarity computation is a hot topic in the field of spatial database. The most classic LCSS [8] and DTW [9] are dynamic programming methods to find the amount of matching points among trajectories. DISSIM [25], EDR [24] and ERP [26] add spatial context to improve the accuracy of computation. Trajectory join is an important operation in trajectory similarity computation. Fang et al. [27] develop a MapReduce framework for the problem of KNN trajectory join. Specifically, the algorithms in [27] can obtain the top- k similar trajectories for two trajectory sets given a threshold and a time interval. The authors in [28] propose TS-Joins to evaluate trajectory join problem in spatial networks. They focus on efficient algorithms to enable parallel TS-Joins processing. Shang et al. [29] propose a weighted bi-graph cost model and design a graph orientation mechanism to co-ordinate the distributed trajectory join. Although the work mentioned above improves the efficiency and effectiveness of trajectory similarity computation, there are two general problems. One is trajectory heterogeneity problem, which has many negative influences on the precision of similarity computation. The other is that all these methods only analyze the trajectory similarity from the spatial and temporal domains, but ignore the importance of semantic information in the activity trajectories.

Many researchers point out that different sampling rates have a great influence on the accuracy of the experimental results. Su [12] and Ranu [11] et al. propose methods named APM and EDwP respectively to make the model sensitive to the low sampling period of the trajectory. The main idea of APM is to use the anchor points to align and complete the trajectory. Unlike Su's practice, EDwP adopts the projection and coverage between the two trajectories to complete the trajectory for unifying the sampling rates between two trajectories. Li et al. [13] raise a deep learning model, which attempts to learn a complementing vector of the trajectory. These methods are currently state-of-the-art for measuring trajectory similarity, but all of them only consider the spatial and temporal characteristics of the trajectory.

Our work differs from the above methods because the activity information of the trajectory is taken into consideration and embedded to a low dimensional vector for promoting the trajectory similarity computation. Although activity information in raw trajectories has been studied in user behavior similarity analysis [30], and similarity of

Fig. 20: Effect of different parameters.

user preferences on account of the routine activity [31], our focus is on trajectory similarity, rather than user behaviors/preferences. Other study of activity information is less relevant. Zheng et al. [32] regard some words containing activity information in the activity trajectory as keywords, so as to perform trajectory analysis based on keyword search. Zheng et al. [33] make KCKNN query based on road network by using activity trajectory data. Yao et al. [34] combine the recurrent neural network of deep learning with multiple features of semantic trajectory to predict user's next location. In summary, for activity trajectory, most of the existing studies aim to make queries or model user preference. However, in our work, we pay attention to the analysis of similarity of trajectory itself. More importantly, we take the keyword information of trajectory into account, so that the problem of low-sampling trajectory can be better solved.

5.2 Attention mechanism in Seq2Seq model

With the continuous development of deep learning, the Seq2Seq model has been widely used in the field of machine translation in natural language processing. Sutskever et al. [35] first propose sequence to sequence auto-encoder for machine translation. In order to capture the order information of the sequence, Cho et al. [19] develop RNN-based sequence to sequence model. Many researchers have also demonstrated the usefulness of sequence to sequence auto-encoders for generating representation for videos and sentences. Explicitly, Chung et al. [36] employ auto-encoders to model audio and embed it to vector. Moreover, Palangi et al. [37] apply the idea of sequence embedding to the field of IR (information retrieval).

The Seq2Seq framework is very classic, but it still contains many limitations. First, the semantic vector, which connects the encoder and the decoder may not fully represent the information of the entire sequence. Second, the information carried in the content first input to the network will be overwritten by the information input later. In order to solve the inherent problems of the model, Bahdanau et al. [38] first apply the attention mechanism in the Seq2Seq model for Neural Machine Translation tasks. Luong et al. [39] propose two kinds of attention mechanisms, named global mechanism and local mechanism, which make the attention mechanism expand in RNN. Yin et al. [40] explore the combination of attention mechanism and CNN structure.

Recently, BERT model proposed by Google is also based on their paper of attention mechanism [41]. In addition to the NLP tasks, the attention mechanism has also widely employed in many other fields of research [42] [43].

In our model, we utilize Seq2Seq model for trajectory similarity computation to solve the problem of trajectory heterogeneity in the spatial database field. Meanwhile, we introduce multi-level attention mechanisms to improve the Seq2Seq model.

6 CONCLUSION

We have studied the problem of trajectory similarity in this paper, focusing on solving the heterogeneity of trajectories. A novel framework referred to as At2vec has been proposed to model the similarity between activity trajectories. In order to solve the problem of different sampling rates, we extract features from three aspects and concatenate three semantic vectors. Combined with the deep learning Seq2Seq model, we obtain a robust vector containing these semantics. Furthermore, we introduce multi-level attention mechanisms to dynamically select the relevant points and discriminative features for trajectory similarity computation. Finally, we design a new loss function called STA to learn a consistent representation of trajectory. The performance of the At2vec has been demonstrated through extensive experiments on two large trajectory datasets.

ACKNOWLEDGMENTS

This paper is partially supported by Natural Science Foundation of China (Grant No. 61572336, No. 61572335, No. 61632016, No. 61772356, No. 61802344, No. 61602400, No. 61702227), and Natural Science Research Project of Jiangsu Higher Education Institution (No. 18KJA520010, No. 17KJA520003), and a Hong Kong Polytechnic University start-up fund (project no. 1.9B0V), and a Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions. This work is supported by the Open Program of Neusoft Corporation, item number SKLSAOP1801.

REFERENCES

- [1] C. Guo, B. Yang, J. Hu, and C. S. Jensen, "Learning to route with sparse trajectory sets," in *ICDE*, 2018, pp. 1073–1084.

- [2] J. Hu, B. Yang, C. Guo, and C. S. Jensen, "Risk-aware path selection with time-varying, uncertain travel costs: a time series approach," *VLDB J.*, vol. 27, no. 2, pp. 179–200, 2018.
- [3] C. Guo, B. Yang, J. Hu, C. S. Jensen, and L. Chen, "Context-aware, preference-based vehicle routing," *VLDB Journal*, online first, 2020.
- [4] M. Xiao, K. Ma, A. Liu, H. Zhao, Z. Li, K. Zheng, and X. Zhou, "SRA: secure reverse auction for task assignment in spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 4, pp. 782–796, 2020.
- [5] J. Zhu, W. Jiang, A. Liu, G. Liu, and L. Zhao, "Effective and efficient trajectory outlier detection based on time-dependent popular route," *World Wide Web*, vol. 20, no. 1, pp. 111–134, 2017.
- [6] C.-C. Hung, W.-C. Peng, and W.-C. Lee, "Clustering and aggregating clues of trajectories for mining trajectory patterns and routes," *The VLDB Journal/The International Journal on Very Large Data Bases*, vol. 24, no. 2, pp. 169–192, 2015.
- [7] A. Liu, K. Zheng, L. Li, G. Liu, L. Zhao, and X. Zhou, "Efficient secure similarity computation on encrypted trajectory data," in *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015, pp. 66–77.
- [8] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE, 2002, pp. 673–684.
- [9] B.-K. Yi, H. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Data Engineering, 1998. Proceedings., 14th International Conference on*. IEEE, 1998, pp. 201–208.
- [10] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012, pp. 1144–1155.
- [11] S. Ranu, P. Deepak, A. D. Telang, P. Deshpande, S. Raghavan *et al.*, "Indexing and matching trajectories under inconsistent sampling rates," in *2015 IEEE 31st International Conference on Data Engineering (ICDE)*. IEEE, 2015, pp. 999–1010.
- [12] H. Su, K. Zheng, J. Huang, H. Wang, and X. Zhou, "Calibrating trajectory data for spatio-temporal similarity analysis," *The VLDB Journal/The International Journal on Very Large Data Bases*, vol. 24, no. 1, pp. 93–116, 2015.
- [13] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 617–628.
- [14] Y. Zhang, A. Liu, G. Liu, Z. Li, and Q. Li, "Deep representation learning of activity trajectory similarity computation," in *2019 IEEE International Conference on Web Services (ICWS)*. IEEE, 2019, pp. 312–319.
- [15] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585.
- [16] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.
- [17] S. Wang and L. Cao, "Inferring implicit rules by learning explicit and hidden item dependency," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.
- [18] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: challenges, progress and prospects," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 6332–6338.
- [19] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
- [20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.
- [21] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
- [22] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language," in *Advances in Neural Information Processing Systems*, 2015, pp. 2773–2781.
- [23] R. H. Güting and M. Schneider, "Realm-based spatial data types: the rose algebra," *The VLDB Journal/The International Journal on Very Large Data Bases*, vol. 4, no. 2, pp. 243–286, 1995.
- [24] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 491–502.
- [25] E. Frentzos, K. Gratsias, and Y. Theodoridis, "Index-based most similar trajectory search," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, 2007, pp. 816–825.
- [26] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *Proceedings of the Thirtieth international conference on Very large data bases—Volume 30*. VLDB Endowment, 2004, pp. 792–803.
- [27] Y. Fang, R. Cheng, W. Tang, S. Maniu, and X. Yang, "Scalable algorithms for nearest-neighbor joins on big trajectory data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 785–800, 2015.
- [28] S. Shang, L. Chen, Z. Wei, C. S. Jensen, K. Zheng, and P. Kalnis, "Parallel trajectory similarity joins in spatial networks," *The VLDB Journal*, vol. 27, no. 3, pp. 395–420, 2018.
- [29] Z. Shang, G. Li, and Z. Bao, "Dita: Distributed in-memory trajectory analytics," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 725–740.
- [30] M.-J. Lee and C.-W. Chung, "A user similarity calculation based on the location for social network services," in *International Conference on Database Systems for Advanced Applications*. Springer, 2011, pp. 38–52.
- [31] M. Lv, L. Chen, and G. Chen, "Mining user similarity based on routine activities," *Information Sciences*, vol. 236, pp. 17–32, 2013.
- [32] B. Zheng, N. J. Yuan, K. Zheng, X. Xie, S. Sadiq, and X. Zhou, "Approximate keyword search in semantic trajectory database," in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 2015, pp. 975–986.
- [33] B. Zheng, K. Zheng, X. Xiao, H. Su, H. Yin, X. Zhou, and G. Li, "Keyword-aware continuous knn query on road networks," in *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 2016, pp. 871–882.
- [34] D. Yao, C. Zhang, J. Huang, and J. Bi, "Serm: A recurrent model for next location prediction in semantic trajectories," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 2411–2414.
- [35] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- [36] Y.-A. Chung, C.-C. Wu, C.-H. Shen, and H.-Y. Lee, "Unsupervised learning of audio segment representations using sequence-to-sequence recurrent neural networks," in *Proc. Interspeech*, 2016.
- [37] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 4, pp. 694–707, 2016.
- [38] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.
- [39] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.
- [40] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "Abcnn: Attention-based convolutional neural network for modeling sentence pairs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 259–272, 2016.
- [41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
- [42] S. Wang, L. Hu, L. Cao, X. Huang, D. Lian, and W. Liu, "Attention-based transactional context embedding for next-item recommendation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [43] 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 *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

An Liu is a Professor in the Department of Computer Science & Technology at Soochow University. He received his Ph. D. degree in computer science from both City University of Hong Kong (CityU) and University of Science and Technology of China (USTC) in 2009. His research interests include spatial databases, crowdsourcing, recommender systems, data security and privacy, and cloud/service computing.

Lei Zhao is a professor in the School of Computer Science and Technology at Soochow University, Suzhou. He received his Ph.D. degree in computer science from Soochow University, Suzhou, in 2006. His research focuses on graph databases, social media analysis, query outsourcing, parallel and distributed computing. His recent research is to analyze large graph databases in an effective, efficient, and secure way.

Yifan Zhang is working toward the master degree in the Research Center on Advanced Data Analytics (ADA), Soochow University, China. He has published his work on trajectory similarity at IEEE ICWS. His research interests include spatial database, recommender systems, and deep learning.

Xiangliang Zhang is an Associate Professor and directs the Machine Intelligence and Knowledge Engineering (MINE) Laboratory in King Abdullahs University of Science and Technology (KAUST). She earned her Ph.D degree in computer science with great honors from INRIA-University Paris-Sud 11, France, in 2010. Her main research interests and experiences are in diverse areas of machine learning and data mining.

Guanfeng Liu is currently a Lecturer in the Department of Computing, Macquarie University, Sydney, Australia. He received his PhD degree in Computer Science from Macquarie University in 2013. His research interests include graph database, trust computing, and social computing.

Yanan Zhang is working toward the graduate degree at Soochow University, China. She is specifically working on the POI recommendation system for improving interested place ranking. Her research interests include machine learning, trajectory prediction, information security.

Zhixu Li is an associate professor in the Department of Computer Science and Technology at Soochow University, Suzhou. He received his Ph.D. degree in computer science from the University of Queensland in 2013, and his B.S. and M.S. degree in computer science from Renmin University of China, Beijing in 2006 and 2009 respectively. His research interests include data cleaning, information extraction and retrieval, machine learning, deep learning, knowledge graph and crowdsourcing.

Qing Li is a Chair Professor at the Department of Computing, the Hong Kong Polytechnic University. He received his B.Eng. from Hunan University (Changsha), and M.Sc. and Ph.D. degrees from the University of Southern California (Los Angeles), all in computer science. His research interests include multi-modal data management, conceptual data modeling, social media, Web services, and e-learning systems. He has authored/co-authored over 400 publications in these areas. He is actively involved in the research community and has served as an associate editor of a number of major technical journals including IEEE Transactions on Knowledge and Data Engineering (TKDE), ACM Transactions on Internet Technology (TOIT), Data Science and Engineering (DSE), World Wide Web (WWW), and Journal of Web Engineering, in addition to being a Conference and Program Chair/Co-Chair of numerous major international conferences. He also sits in the Steering Committees of DASFAA, ER, ACM RecSys, IEEE U-MEDIA, and ICWL. Prof. Li is a Fellow of IEE/IET (UK), and a distinguished member of CCF (China).

Xiaofang Zhou is a professor of computer science with The University of Queensland. He is the head of the Data and Knowledge Engineering Research Division. He is a specially appointed adjunct professor under the Chinese National Qianren Scheme hosted by the Renmin University of China (2010-2013), and by Soochow University where he has led the Research Center on Advanced Data Analytics (ADA) since July 2013. He has been working in the area of spatial and multimedia databases, data quality, high performance query processing, Web information systems, and bioinformatics, and has co-authored more than 250 research papers with many published in top journals and conferences. He is a fellow of the IEEE.