Trajectory-User Link with Attention Recurrent Networks

Conference Paper · January 2021

DOI: 10.1109/ICPR48806.2021.9412453

Yong-Jun Xu
Chinese Academy of Sciences
243 PUBLICATIONS **4,188** CITATIONS
SEE PROFILE

Fei Wang
Chinese Academy of Sciences
67 PUBLICATIONS **2,355** CITATIONS
SEE PROFILE

Lin Wu
Chinese Academy of Sciences
27 PUBLICATIONS **498** CITATIONS
SEE PROFILE

Tangwen Qian
Chinese Academy of Sciences
35 PUBLICATIONS **586** CITATIONS
SEE PROFILE

2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Trajectory-User Link with Attention Recurrent Networks

Tao Sun*†, Yongjun Xu*, Fei Wang*, Lin Wu*, Tangwen Qian*†, Zezhi Shao*†
*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
†University of Chinese Academy of Sciences, Beijing, China
Email: *{suntao, xyj, wangfei, wulin, qiantangwen, shaozezhi16}@ict.ac.cn

Abstract—**The prevalent adoptions of GPS-enabled devices have witnessed an explosion of various location-based services which produces a huge amount of trajectories monitoring the individuals' movements. In this paper, we tackle Trajectory-User Link (TUL) problem, which identifies humans' movement patterns and links trajectories to the users who generated them. Existing solutions on TUL problem employ recurrent neural networks and variational autoencoder methods, which face the bottlenecks in the case of excessively long trajectories and fragmentary users' movements. However, these are common characteristics of trajectory data in reality, leading to performance degradation of the existing models. In this paper, we propose an end-to-end attention recurrent neural learning framework, called TULAR (Trajectory-User Link with Attention Recurrent Networks), which focus on selected parts of the source trajectories when linking. TULAR introduce the Trajectory Semantic Vector (TSV) via unsupervised location representation learning and recurrent neural networks, by which to reckon the weight of parts of source trajectory. Further, we employ three attention scores for the weight measurements. Experiments are conducted on two real world datasets and compared with several existing methods, and the results show that TULAR yields a new state-of-the-art performance. Source code is public available at GitHub: https://github.com/taos123/TULAR.**

I. Introduction

With the developments of tracking techniques (i.e., GPS), the movement trajectories of urban entities (e.g., taxi, bus, passenger) are collected automatically and become widely available. Big urban trajectory data analysis plays an essential role in many smart city applications, such as traffic management [1], [2], events predictions [3], [4] and navigation service. Discovering human moving behavior is an important part of trajectory data analysis, having attracted wide attention from both academia and industry, which is extensively applied in location based services (LBS) and location based social networks (LBSN) [5], [6]. Human movement analysis enables a better understanding and using in various applications such as: mobility pattern discovering [7], [8], next visit-location recommendation [9]–[11] and routine planning [12].

In this paper, we discuss TUL problem (Trajectory-User Link) , an essential task in human moving behavior analysis, aiming to identify and link the trajectories to users who generate them [13]. TUL can be applied in a lot of trajectory data mining domains such as merging two trajectory datasets, collected from different sources, where users' labels are inconsistent; Recognizing target identities in anonymous trajectories such as radar system. The main challenges of TUL lie on how to model both the complex spatial and temporal information of trajectories and how to discriminate resemble moving patterns of different users. The latter can put another way that the difference of trajectories of the same user at different times are probably bigger than the trajectories of different accompanied users. However, the common cluster analysis is that objects in the same group are more similar to each other than to those in other groups. Those following problems will also bring troubles to TUL: the number of users is much larger than categories in the general trajectory classification and the trajectory datasets are very spare acquired in real world.

Trajectory similarity based methods such as LCSS (Longest Common Sub-Sequence), EDR (Edit Distance on Real Sequence) and DTW (Dynamic Time Warping) are one kind of TUL solutions. Those methods are general trajectory classification methods but face problems when the data magnitude increase. Deep neural networks based models have achieved a better performance in recent years. TULER [13] and TULVAE [14] are proposed for TUL problem solutions, which employ deep recurrent neural networks for modeling the trajectory information, achieving the best performance. TULER employs sequence embedding methods to map check-in locations into vector space, called check-in embedding, which is inspired by word2vec method in natural language process [15], [16]. Check-in embedding benefits from acquiring a better semantic representation for locations than representation by longitude and latitude. DNN based models use RNNs (e.g., LSTM [17] or GRU [18]) to model the sequence of check-in and distinguish different users of trajectories. However, TULER and TULVAE face two deficiencies: (1) The check-in embedding process enhances the semantic information for locations meanwhile maps the different location in the similar vectors, which results in hardly distinguish the accompanied patterns. (2) As the growing of trajectory length, it will be harder to model whole trajectory information for recurrent neural networks.

In this paper, we propose a novel Trajectory-User Link method, called **TULAR**, which solves TUL problem with Attention Recurrent Neural Networks. We propose Trajectory Semantic Vector (TSV) in TULAR, via trajectory embedding and recurrent neural networks model, which maps the variable-length source trajectories to fixed-length vectors in feature space. In addition, we put to use three different attention scores to focus on selected parts of source trajectory by supervised learning. Recalculated TSV is employed for ultimate trajectory-user linking. Comparing with existing work, our main contributions are:

- A novel Trajectory-User Link method, TULAR, is proposed, which improves both accuracy and efficiency for TUL problem. TULAR is an end-to-end trajectory identifying neural network framework, with superb scalability.
- TULAR introduces TSV, a representation learning method, mapping the variable-length trajectories to fixed-length vectors in feature space. Trajectories of the same user are likely more similar to each other in feature space, which is untenable in primitive geospatial space.
- Three different trajectory embedding methods and three different attention score measures are used in this paper. A lots of experiments are conducted to demonstrate our improvements, using four real world datasets and compared with several state-of-art methods.
- We arrange an easy-to-use project and make the source code public available at GitHub: https://github.com/taos123/TULAR.

The rest of paper is assigned as follow: In section 2 we introduce the relation works about TUL problem and attention recurrent networks. We introduce preliminaries in section 3. Technical detail of TULAR method is discussed in section 4. In section 5, we present the detail of experiments and show the results. In section 6 we conclude our work.

II. Related Works

Trajectory-User link (TUL) problem is a tough trajectory classification problem because the number of the classes is much large than common trajectory classification. Although it has been formally defined and addressed until the year of 2017, many works have been done for it. The traditional approaches, such as DTW, LCSS and Trjectory-Hausdorff Distance [19], find the similarity between different trajectories. In this way, similar trajectories are classified into same user. In recent year, deep neural networks have been confirmed a better performance in the TUL problem [13], [14]. Those methods employ embedding method to enhance the locations semantic information. Meanwhile, RNNs (e.g., LSTM, GRU and their variant) are employed to model the sequence characteristics of trajectory. More information is considered into the model in [14], which learns the human movement patterns in a neural generative architecture with stochastic latent variables than span hidden statues in RNN.

Attention is human vision mechanism, which focus the restricted resource to acquire more valuable information and it is widely used in deep neural networks. Attention recurrent networks are first employed under the encoder-decoder architectures, also called sequence to sequence [20] (seq2seq) model, which contains two parts of RNNs model and are widely employed in nature language process and other sequence process. Attention mechanism is selectively focusing on parts of the source sequence. The core of attention is to calculate the attention scores. Attention recurrent networks are used to improve performance of NLP task such as neural machine translation (NMT) [21], automatic summarization [22], sentiment analysis [23], [24], etc. Attention mechanism have been used in location prediction tasks [10], predicting the future locations of users based on the historical movements. Yet attention mechanism has not been used in trajectory-user link problems. In this paper, we propose the trajectory semantic vector, which is calculated by users' trajectories under attention mechanism. We will show the details in the following sections.

III. Problem Statement

Before we introduce TULAR, we need to expound the unambiguous definition about TUL problem. In addition, we will also introduce notations used in this paper.

a) Trajectory: Trajectory is a sequence of geo-points which records the movements of human or other targets such as animals, hurricane and vehicles. A general trajectory has three key elements: who, when and where, with the following forms $T = \{< u, t_1, p_1 >, < u, t_2, p_2 >, \cdots < u, t_n, p_n >\}$, where u is the targets identity recording "who", t is the timestamp recording "when" and p is position information recording "where". In some cases such as radar system, the target identities are unknown, which are called anonymous trajectories. In real world datasets, the position can be a coordinate of latitude and longitude or POI (point of interests) such as park, restaurant, etc. Trajectories consisted by POIs also called check-in based trajectories. In this paper, we use check-in based trajectory datasets as the basic inputs. In the real world datasets, the sampling rate is unsettled, and the length of trajectory is variable.

b) Trajectory-User Link: TUL is a task to identify anonymous trajectories and link them to users who generate them. Let $\mathcal{T} = T_1, T_2, \cdots, T_m$ denotes the set of trajectories where users' identities are unknown and $\mathcal{U} = U_1, U_2, \cdots, U_n$ denotes the set of users. In the real world datasets, m is much larger than n in general. The linking task is to find a map function $f(T)$, satisfied the follow Equation (1).

$$\min_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \|f(T) - T_r\|, f(T) \in \mathcal{U}, T_r \in \mathcal{U} \tag{1}$$

Where $\|\cdot\|$ denotes difference evaluation operator, T_r is the real user of trajectory T and $\mathcal{F}$ is hypothesis space of TUL problem.

IV. Method

In this section, we introduce TULAR, a framework for TUL task. The overview of TULAR is shown in Figure 1. Trajectory data preprocess and segmentation are not the key points in this paper and we follow preprocess operations as literature [13]. We will focus on TSV evaluation and attention recurrent neural networks for linking in the follow section.

A. Trajectory Semantic Vector Evaluation

There are two challenges in trajectory mining process: (1) the length of trajectories are unfixed; (2) the information of locations in trajectories are rarely considered. We propose

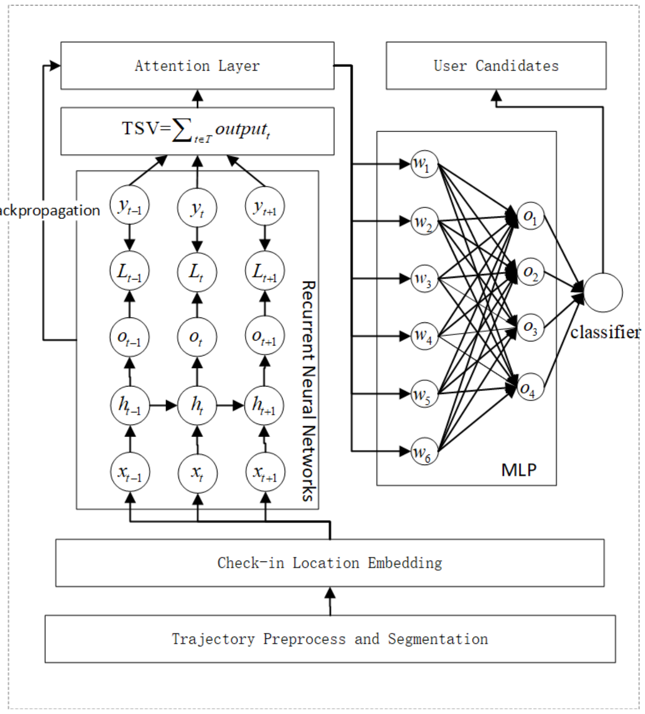
Attention Layer
User Candidates
TSV=$\sum_{t\in T} output_t$
backpropagation
Recurrent Neural Networks
classifier
MLP
Check-in Location Embedding
Trajectory Preprocess and Segmentation

Fig. 1: The structure of TULAR. There are four main parts of TULAR structure: trajectory preprocess and segmentation, check-in embedding, attention recurrent neural networks and multi-layer perceptron for linking. The first three parts constituted TSV evaluation. The last part determine who generated the trajectories.

trajectory semantic vector (TSV), which is a fix-length vector, mapped from source trajectory into the N-dimension Euclidean space. We employ location embedding, an unsupervised learning method which learns a map function from geographical space into vector space so that the same locations are in the near vectors. Then a recurrent neural networks model is used to encode the sequences of location embedding to get the TSV.

Trajectory is a sophisticated data type for every location constituting the trajectory containing strong spatial semantic information. Those locations are corresponding to the real geographical position in the world, such as restaurant, emporium, residence, etc. The sequence of locations reflect the temporal semantic information which reflects the user's visiting preference, habits and patterns. As a result, extracting the appropriate feature from original trajectory is an essential process in the trajectory data mining domains. Then we extract those information from source trajectories as the format $< p_i, C(p_i, p) >$, where the $C(p_i, p)$ is the context of location p_i. It is obvious that the locations with same semantic information are probably with the homologous context.

Check-in locations represented by longitude and latitude are inadequacy for neural network model. Following [13], we represent each check-in locations with a fixed-dimensional vector v, which called check-in embedding. Similar with word2vec in the nature language process, check-in embedding learns the vector representation from the check-in context in the trajectory. The benefits of check-in embedding are to map the similar semantic check-in locations into nearby vectors in the Euclid space. To get check-in locations vector representation, we need maximize the conditional probability $p(v(p_i))$, which is defined by Equation 2.

$$p(v(p_i)|C(p_i,p)) = \prod_{p' \in C(p_i,p)} p(v(p_i)|v(p')) = \prod_{p' \in C(p_i,p)} \frac{\exp\{v(p_i)\cdot v(p')\}}{\sum_{p''\in C(p_i,p)} \exp\{v(p'')\cdot v(p')\}} \tag{2}$$

Where $v(p_i)$ represents the check-in location vector representation and $C(p_i, p)$ is the context of p_i.

However, there is another problem in the check-in location embedding. The original trajectories are recorded by days, which may create the cases that dividing unrelated check-in locations into an integrated trajectory. To address this problem, we repartition original trajectories into sub-sequences by the time intervals (e.g., 6 hours [13]).

After embedding check-in location into vectors, we need to consider another problem in trajectory. We employ a RNN based model to encode the trajectory to a vector. RNN is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. It allowed models to exhibit temporal dynamic behaviors. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. Three RNN variants including: long short-term memory (LSTM), gated recurrent units (GRU) and bidirectional recurrent neural networks (BRNN) are employed in TULAR.

First we employ LSTM for TSV evaluation. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTMs are developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. For the sub-trajectory $T = \{l_1, l_2, \cdots, l_k\}$,and let h_{t-1} denote the last state, h_t denotes the current state and $\widetilde{h_t}$ denotes the candidate state. We can use recursion formula as follow.

$$f_t = \sigma_g(W_f v(p_t) + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i v(p_t) + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o v(p_t) + U_o h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c v(p_t) + U_c h_{t-1} + b_c)$$

$$h_t = o_t \circ \sigma_h(c_t)$$

In the training process, the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\circ$ denotes the Hadamard product (element-wise product). The subscript t indexes the time step. $v(p_t)$ is input vector to the LSTM units. f_t is forget gate's activation vector. i_t is update gate's activation vector. o_t is output gate's activation vector. h_t is hidden state vector also known as output vector of LSTM units. c_t is cell state vector. W, U and b are weight matrices and bias vector parameters which need to be learned during training. σ_g is a sigmoid function. σ_c is a hyperbolic tangent. σ_h is hyperbolic tangent. In this way, we can get the every time the RNN output. And then, we use mean of the outputs of every time to evaluate the TSV as follow.

$$TSV = \frac{1}{n}\sum_{t=1}^{n} output_t \tag{3}$$

Then we employ GRU for TSV evaluation. Gated recurrent units (GRU) are a gating mechanism in recurrent neural networks, which is proposed in literature [25]. The GRU is like a long short-term memory (LSTM) with forget gate but has fewer parameters than LSTM, as it lacks an output gate. GRUs have been shown to exhibit even better performance on certain smaller datasets.

$$z_t = \sigma_g\left(W_z v\left(p_t\right) + U_z h_{t-1} + b_z\right)$$

$$r_t = \sigma_g\left(W_r v\left(p_t\right) + U_r h_{t-1} + b_r\right)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h\left(W_h v\left(p_t\right) + U_h\left(r_t \circ h_{t-1}\right) + b_h\right)$$

Initially, for $t = 0$, the output vector is $h_t = 0$. Where the $v(p_t)$ is the input vector, h_t is output vector, z_t is update gate vector, r_t is the reset vector and W, U and b are the parameters need to learn.σ_g is a sigmoid function. σ_h is a hyperbolic tangent. Similar with LSTM, the TSV of sub-trajectory T is evaluated by the mean of hidden outputs of GRU of every time nodes, which is show in Equation 3.

Bidirectional recurrent neural networks is a neural structure which connects two hidden layers of opposite directions to the same output. With this form of generative deep learning, the output layer can get information from past (backwards) and future (forward) states simultaneously. BRNNs were introduced to increase the amount of input information available to the network. Standard recurrent neural network (RNNs) also have restrictions as the future input information cannot be reached from the current state. On the contrary, BRNNs do not require their input data to be fixed. Moreover, their future input information is reachable from the current state. BRNN are especially useful when the context of the input is needed. For example, in handwriting recognition, the performance can be enhanced by knowledge of the letters located before and after the current letter. We choose LSTM units as BRNN units, and the output is shown in Equation 4.

$$output = concat\left(output_{forward}, output_{back}\right) \tag{4}$$

TSV under BRNN is also evaluated by equation 3.

B. Attention Recurrent Neural Networks

In this section, we introduce how to use attention recurrent neural networks in TUL problem. In the above section, we introduce TSV and we want to use it to link the trajectories to users. The previous works employed a softmax function for classification, where the loss function $\mathcal{L}$ can be evaluated as Equation 5.

$$\mathcal{L} = cross\_entropy\left(user, softmax\left(TSV\right)\right) \tag{5}$$

However, there are two problem when using TSV for linking directly: (1) The original trajectories will contain noise points when the trajectories is too long, but the noise points will be given the same weight as the normal points. (2) Evaluating the mean of outputs of every time points will reduce the impact of some key points, which may determine the linking results in the original trajectories.

As it remembered in the vanilla seq2seq model, we pass the last source state from the encoder to the decoder when starting the decoding process. It works well for short and medium-length sentences. However, for long sentences, the single fixed-size hidden state becomes an information bottleneck. Instead of discarding all of the hidden states computed in the source RNN, the attention mechanism provides an approach that allows the decoder to peek at them (treating them as a dynamic memory of the source information). By doing so, the attention mechanism improves the translation of longer sentences. Nowadays, attention mechanisms are the de-facto standard and have been successfully applied to many other tasks, including image caption generation, speech recognition, and text summarization.

Combining with the above point of view, we need to build an attention based model to re-evaluate TSV. We assign a weight to the outputs of each time point. Let a_{ts} denote the weight of each time point, which gives the different importance of every points. Then the new TSV can be evaluated as Equation 6.

$$TSV_{attention} = \sum_{s} a_{ts}\bar{h}_s \tag{6}$$

In the above equation, the key of $TSV_{attention}$ is to evaluate the weight, which is also called the attention score. We calculate the relevancy between the trajectory semantic encoding and check-in locations. As a result, the key point of attention recurrent neural networks is how to evaluate the weight a_{ts}. We reference Luong's [26] multiplicative style and Bahdanau's [21] additive style based score function.

Under the Luong's multiplicative attention style, which called multi-based attention score function, the weight of nodes can be evaluated as follow

$$score\left(h_t, \bar{h}_s\right) = {h_t}^T W \bar{h}_s$$

where W is the parameter need to learn, as the same dimension as input vector.

Under Bahdanau's additive attention style [21], the weight of nodes can be evaluated as follow

$$score\left(h_t, \bar{h}_s\right) = \tanh\left(W_1 h_t + W_2 \bar{h}_s\right)$$

where W_1 and W_2 are the parameter need to learn in the training process. The $tanh()$ is active function.

In addition, we employ a simple dot based style. The weight of nodes can be evaluated as follow.

$$score\left(h_t, \bar{h}_s\right) = {h_t}^T \cdot \bar{h}_s$$

Finally, in case of the computing overflow, we need to normalize the weight of nodes. And the normalized weight of nodes can be evaluated as follow

$$a_{ts} = \frac{\exp\left(score\left(h_t, \bar{h}_s\right)\right)}{\sum_{s'=1}^{S} \exp\left(score\left(h_t, \bar{h}_{s'}\right)\right)}$$

After introduction of attention based TSV evaluation. We need to link the TSV to users where the ultimate goal is to solve TUL problem.

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to refer to any feedforward artificial neural networks, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation). A MLP consists of three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It is able to distinguish data that is not linearly separable.

TUL is a multi-classification problem in essence, so we employ softmax function to link trajectories to their users [13]. For trajectory semantic vector TSV, let $MLP(TSV)$ denotes the outputs of MLP, which is shown as follow.

$$MLP\left(TSV\right) = tanh(W \cdot TSV + b)$$

Let V_u denotes the one hot representation of users. In the final, we rewrite the loss function $\mathcal{L}$ as the Equation 7.

$$\mathcal{L} = cross\_entropy\left(V_u, softmax\left(MLP\left(TSV\right)\right)\right) \quad (7)$$

C. Model Details and Parameter Setting

In this section, we introduce the details of TULAR including check-in location embedding model and attention recurrent neural network model.

An open python library Gensim[1] is applied in our model for check-in location embedding. There are three important parameters in embedding training. Let $vector\ size$ denotes the dimensionality of the location embedding vectors, $window$ denotes the length of the context and $learning\ iteration$ denotes the iterations over the check-in location embedding models. The value of $vector\ size$ and $window$ depend on the size of trajectory training corpus, and we find the model performs best when $vector\ size$ is taken as 250. When we raise the value of $learning\ iteration$, we can get a more stable values of $vector\ size$. We find that if $learning\ iteration$ is greater than 100, $vector\ size$ will converge. We choose skip-gram [15] as the training algorithm.

[1]https://radimrehurek.com/gensim/

TUL problem can be regard as a multi-classification problem as every users regarded as one categories. As a result, we use a Multilayer Perceptron (MLP) to classify TSV. A softmax function is used to map the non-normalized output of MLP to a probability distribution over predicted output classes. We use the cross-entropy loss as TULAR loss function and we use Adam [27] as the optimization method, because it is able to calculate the learning rate of each parameter adaptively.

In addition, in order to reduce overfitting in recurrent neural networks, we employ a regularization technique dropout in our method, which is a very efficient way of performing model averaging with neural networks by preventing complex co-adaptations on training data. We set $droput\ rate$ to 0.5. Let $hidden\ size$ denotes the number of hidden layers in the recurrent neural networks and $attention\ size$ denotes the size of TSV. We adopt grid search strategy to choose the optimal parameters of $hidden\ size$ and $attention\ size$. We set the initial learning rate to 0.00095 and reduce it by 0.0001 step by step. We tried a very large number of learning iteration and find that after 30 iterations of training, the network achieves convergence. The specific parameter values of TULAR are shown in Table I.

TABLE I: TULAR Parameters Details

Parameter	Available Range	Recommended Value
vector size	$[200, 300]$	250
window	≥ 20	20
learning iteration	≥ 20	100
hidden size	$[100, 500]$	300
learning rate	$[0.00015, 0.00095]$	0.00095
dropout rate	$[0, 1]$	0.5
attention size	$[500, 1000]$	600
stacked TULAR	≥ 2	2
training iteration	≥ 30	30

The first column are important parameters in TULAR training. The second column are available ranges of parameter that we have tested. The last column are the specific values we choose in this paper for the best performance.

V. Experiments

In this section, we discuss TULAR performance in real world datasets. There are three aspects which we take into consideration: overall performance, training convergence comparison and training time. Before comparing the performance among various proposed methods, we need to introduce the details of datasets, baselines and metrics.

A. Datasets

We conduct our experiments in the public trajectory data [28]: Gowalla[2] and Brightkite[3]. Both of them are collected from location-based social networking website where users share their locations by check-in record. The data are recorded information as $\{user\_id, timestamp, latitude, longitude, location\_id\}$. We randomly select different number of users in Gowalla and Brightkite which is in the same as [13] in order to be consistent with the benchmarks. The overview of the datasets is shown in Table II.

TABLE II: Datasets description and statistics

Datasets	$\|U\|$	$\|T\|$	$\|C\|$	$\|Ave\|$
Gowalla	201	19968	1958	99.24
	112	9920	6683	88.57
Brightkite	34	9920	652	291.76
	92	19904	471	216.34

$|U|$ is the number of users in the datasets. $|T|$ denotes the number of trajectories. We can see $|T| \gg |U|$. $|C|$ is the number of check-in locations. $|Ave|$ represents the average check-in locations number in one trajectory.

B. Baseline

We compare TULAR with several existing approaches. The introductions of those methods are shown as follow.

- **TULER** [13]. TULER uses the check-in embedding to enhance the check-in locations information and employs RNNs to model the sequence features. TULER employs LSTM, GRU and Bi-LSTM as RNN models which called: TULER-LSTM, TULER-GRU, TULER-LSTM-S, TULER-GRU-S and Bi-TULER. We employ a open source of TULER in github[4].

- **TULVAE** [14]. TULVAE learns the human movements in a neural generative architecture with stochastic latent variables than span hidden states in RNN. TULVAE includes HTULER-L, HTULER-G, HTULER-B and TULVAE. TULVAE was the state-of-the-art method for TUL problem. We employ a open source of TULVAE in github[5].

C. Metrics

We use the $Acc@K$ to measure models performance, which is the common metrics in information retrieval domains.

$$Acc@K = \frac{\#correctly\ identified\ trajectories\ @K}{\#trajectories}$$

where the $\#correctly\ identified\ trajectories\ @K$ is the correct users at top K candidates and $\#trajectories$ is the total number of un-linked trajectories. In addition, because TUL is a multi-classification task, we also need to consider macro-R, macro-P and macro-f1. The macro-R is the mean of recall value of every classification and macro-P is the mean of precision value of every classification. The macro-f1 is defined as follow.

$$macro-F1 = 2 \times \frac{macro-P \times macro-R}{macro-P + macro-R}$$

D. Results

a) Overall: The experiments are conducted on TULAR with three types of RNN variants and three attention scores. In totally, there are nine variants of TULAR, including TULAR-LSTM-M, TULAR-LSTM-A, TULAR-LSTM-D, TULAR-GRU-M, TULAR-GRU-A, TULAR-GRU-D, TULAR-BRNN-M, TULAR-BRNN-A, TULAR-BRNN-D, where M, A, D represent multiplicative style attention, additive style attention and dot style attention. Table III and Table IV show the performance comparisons on two different datasets. The results show that TULAR achieves the best performance than existing methods in terms of accuracy@1, accuracy@5 and macro-F1 metrics. TULAR with BRNN achieves the best results on Gowalla datasets. While, TULAR with LSTM achieves the best results on Brightkite datasets. TULAR yields 10.59%, 8.78%, 16.12% improvements compared to TULVAE (state-of-the-art method) for Acc@1, Acc@5 and macro-F1 metrics, and yields 7.57%, 7.10%, 12.99% improvements compared to TULVAE for Acc@1, Acc@5 and macro-F1 metrics. TULAR with BRNN and dot attention model achieves the best results in terms of accuracy@1 on Brightkite |92|, yielding 21.72%, 18.10% and 32.42% improvements compared to TULVAE for Acc@1, Acc@5 and macro-F1 metrics.

As we can see from Tabel II, the average of trajectories length in Brightkite is much larger than the length in Gowalla. As a result, the performance of RNN based model is greatly reduced on Brightkite, even if the user's number of Brightkite is less than the number of Gowalla. On the contrary, TULAR achieves a better performance on Brightkite, which can be explained that attention mechanism amplifies certain parts of source trajectories.

b) Effect of Attention Mechanism: We compare the effect of attention mechanism on training convergence under different RNN models. Accuracy@1 in 30 iterations on Gowalla |201| with LSTM, GRU and BRNN are shown in the Figure 2 first line. Accuracy@1 in 30 iterations on Brightkite |92| with LSTM, GRU and BRNN are shown in the Figure 2 second line. It can be seen that TULAR with multi-based attention is the lowest accuracy at the beginning of training. The reason of this results is that multi-based attention contains most parameters than the remaining two. Because of our random initialization parameters, it takes longer time for parameter learning.

c) Training Time: Besides, we analyze the effect of attention mechanism on training time. Because the training time is rely depend on datasets, we select one of the datasets to show the training time comparison result. Figure 3 shows the training time consumption comparison on Gowalla |201|. As we can see, attention mechanism does bring extra time cost due

[2]Gowalla: http://snap.stanford.edu/data/loc-Gowalla.html
[3]Brightkite: http://snap.stanford.edu/data/loc-Brightkite.html
[4]TULER: https://github.com/gcooq/TUL
[5]TULVAE: https://github.com/AI-World/IJCAI-TULVAE

TABLE III: TULAR Performance Comparison on Gowalla

Metric / Method	Acc@1	Acc@5	macro-P	macro-R	macro-F1	Acc@1	Acc@5	macro-P	macro-R	macro-F1
	$\|U\| = 112$					$\|U\| = 201$				
TULER-LSTM	41.79	57.89	33.61	31.33	32.43	41.24	56.88	31.70	28.60	30.07
TULER-GRU	42.61	57.95	35.22	32.69	33.91	40.85	57.31	29.52	27.80	28.64
TULER-LSTM-S	42.11	58.01	33.49	31.97	32.71	41.22	57.70	29.34	28.68	29.01
TULER-GRU-S	41.35	58.45	32.51	31.79	32.15	41.07	57.49	29.08	27.17	28.09
Bi-TULER	42.67	59.54	37.55	33.04	32.15	41.95	57.58	32.15	31.66	31.90
HTULER-L	43.89	60.90	35.95	34.32	35.12	43.40	60.25	34.43	33.63	34.02
HTULER-G	43.33	60.74	37.71	34.47	36.01	42.88	59.41	32.72	32.54	32.63
HTULER-B	44.21	62.28	36.48	33.51	34.93	44.50	60.93	34.89	34.46	34.67
TULVAE	44.35	64.46	40.28	32.89	36.21	45.40	62.39	36.13	34.71	35.41
TULAR-LSTM-M	48.74	70.02	45.45	40.73	42.95	47.47	67.61	41.20	38.69	39.91
TULAR-LSTM-A	48.42	70.02	45.42	40.06	42.57	47.79	67.70	40.26	39.22	39.73
TULAR-LSTM-D	48.00	68.65	47.17	41.38	44.08	47.26	67.08	38.65	38.72	38.69
TULAR-GRU-M	46.22	66.87	43.45	38.13	40.61	45.79	64.35	36.89	36.98	36.93
TULAR-GRU-A	46.22	69.28	40.58	40.16	40.37	46.26	66.29	36.86	37.96	37.40
TULAR-GRU-D	47.48	67.92	40.90	38.46	39.64	45.11	64.30	37.36	37.35	37.36
TULAR-BRNN-M	48.63	69.49	46.18	40.72	43.28	48.00	67.29	40.22	39.78	40.00
TULAR-BRNN-A	48.21	70.85	43.85	41.31	42.41	**48.84**	**66.82**	**40.38**	**39.64**	**40.01**
TULAR-BRNN-D	**49.05**	**70.12**	**43.26**	**40.91**	**42.05**	48.21	67.40	41.51	40.24	40.87

TULAR with BRNN and dot attention achieves the best results in terms of accuracy@1, yielding 10.59%, 8.78%, 16.12% improvements compared to TULVAE for Acc@1, Acc@5 and macro-F1 metrics on Gowalla |112|. TULAR with BRNN and additive attention achieves the best results in terms of accuracy@1, yielding 7.57%, 7.10%, 12.99% improvements compared to TULVAE for Acc@1, Acc@5 and macro-F1 metrics on Gowalla |112|.

TABLE IV: TULAR Performance Comparison on Brightkite

Metric / Method	Acc@1	Acc@5	macro-P	macro-R	macro-F1	Acc@1	Acc@5	macro-P	macro-R	macro-F1
	$\|U\| = 34$					$\|U\| = 92$				
TULER-LSTM	48.26	67.39	49.90	47.20	48.51	43.01	59.84	38.45	35.81	37.08
TULER-GRU	47.84	67.42	48.88	46.87	47.85	44.03	61.36	38.86	36.47	37.62
TULER-LSTM-S	47.88	67.38	48.81	47.03	47.62	44.23	61.00	38.02	36.33	37.16
TULER-GRU-S	48.08	68.23	48.87	46.74	47.78	43.93	61.85	37.93	36.01	36.94
Bi-TULER	48.13	68.17	49.15	47.06	48.08	43.54	60.68	38.20	36.47	37.31
HTULER-L	49.44	71.13	51.51	47.31	49.32	45.26	63.55	41.61	38.13	39.79
HTULER-G	49.12	70.81	51.85	46.88	49.24	44.50	63.17	41.10	37.51	39.22
HTULER-B	49.78	70.69	52.45	47.98	48.90	45.30	63.93	41.82	39.32	38.60
TULVAE	49.82	71.71	51.26	46.43	48.72	45.98	64.84	43.15	39.65	41.32
TULAR-LSTM-M	52.86	74.11	51.19	49.40	50.28	**58.45**	**76.58**	**56.56**	**52.99**	**54.72**
TULAR-LSTM-A	**53.85**	**75.69**	**50.84**	**48.32**	**49.55**	56.32	75.41	56.09	51.06	53.46
TULAR-LSTM-D	53.16	75.59	48.69	48.62	48.66	58.04	75.11	56.33	52.75	54.48
TULAR-GRU-M	52.17	73.22	47.17	46.60	46.88	55.66	74.20	56.21	51.11	53.54
TULAR-GRU-A	52.56	72.92	52.97	49.08	50.95	55.61	74.35	54.17	50.72	52.39
TULAR-GRU-D	53.06	72.82	49.34	47.44	48.37	56.42	73.69	57.20	52.92	54.98
TULAR-BRNN-M	53.35	75.00	50.06	49.02	49.54	57.38	75.46	58.66	51.66	54.94
TULAR-BRNN-A	53.75	76.67	50.28	47.62	48.91	57.08	75.21	56.63	51.15	53.75
TULAR-BRNN-D	53.65	73.61	49.23	48.63	48.93	57.84	76.02	58.55	53.32	55.81

TULAR with BRNN and dot attention achieves the best results in terms of accuracy@1, yielding 8.08%, 5.55% and 1.70% improvements compared to TULVAE for Acc@1, Acc@5 and macro-F1 metrics on Brightkite |34|. TULAR with BRNN and dot attention achieves the best results in terms of accuracy@1 on Brightkite |92|, yielding 21.72%, 18.10% and 32.42% improvements compared to TULVAE for Acc@1, Acc@5 and macro-F1 metrics.

to drawing into new parameters. Multi based attention takes the most time, dot-based attention takes the least time and add-based attention in between. However, the consumption of time is almost the same among the three attention mechanism.

VI. CONCLUSION

In this paper, we proposed TULAR, an attention based recurrent neural networks framework for Trajectory-User Link problem. We discovered that RNN based model could not distinguish trajectories correctly when the trajectories are out of length. A conception of trajectory semantic vector had been proposed in the TULAR, which was used in measure the effectiveness of the parts of original trajectories for user linking. A lot of experiments have been conducted in real world datasets for TULAR performance.

There were two aspects that we thought can improve TUL performance: (1) Geospatial information should be taken into consideration to reduce the number of user candidates. (2) Users' behavior preference should be taken into consideration when linking trajectories to users. Those thoughts will be confirmed in the future works.

ACKNOWLEDGMENT

This work is partially supported by NSFC No.61902376 and NSFC No. 61602447.

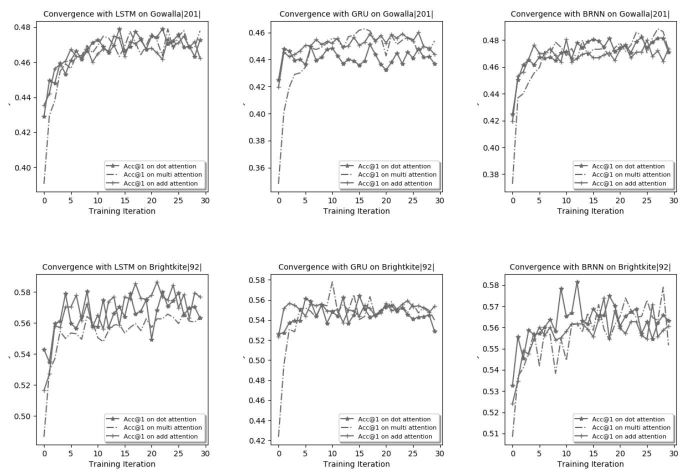

Fig. 2: Convergence comparison with different attention scores. Accuracy@1 in 30 iterations on Gowalla |201| with LSTM, GRU and BRNN are shown in the first line. Accuracy@1 in 30 iterations on Brightkite |92| with LSTM, GRU and BRNN are shown in the second line.

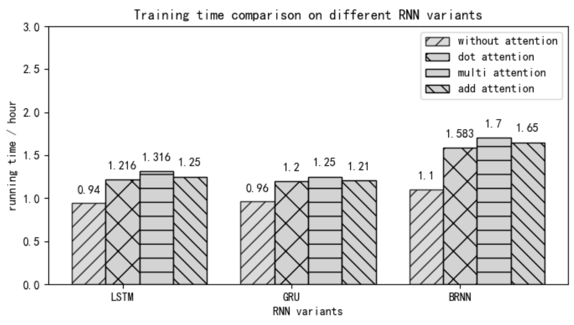

Fig. 3: Training time comparison on different RNN variants. The ordinate represent the training time within 30 iterations. Those experiments were conducted on one GeForce GTX 1070 GPU with i7-6770 HQ CPU and 16G memory.

References

[1] T. Stollenwerk, B. O'Gorman, D. Venturelli, S. Mandrà, and R. Biswas, "Quantum annealing applied to de-conflicting optimal trajectories for air traffic management," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, 2017.

[2] L. Qiang and K. D. Kim, "Autonomous and connected intersection crossing traffic management using discrete-time occupancies trajectory," *Applied Intelligence*, no. 4, 2017.

[3] A. Vahedian, X. Zhou, L. Tong, Y. Li, and J. Luo, "Forecasting gathering events through continuous destination prediction on big trajectory data," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2017, pp. 1–10.

[4] H. Xiong, A. Vahedian, X. Zhou, Y. Li, and J. Luo, "Predicting traffic congestion propagation patterns: a propagation graph approach," in *Proceedings of the 11th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, 2018, pp. 60–69.

[5] Y. Zheng, "Tutorial on location-based social networks," in *Proceedings of the 21st international conference on World wide web, WWW*, vol. 12, no. 5, 2012.

[6] Z. Yu, "Trajectory data mining: An overview," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 3, pp. 1–41, 2015.

[7] H. Jeung, M. L. Yiu, and C. S. Jensen, "Trajectory pattern mining," in *Computing with spatial trajectories*. Springer, 2011, pp. 143–177.

[8] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1081–1094, 2008.

[9] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "Wherenext: a location predictor on trajectory pattern mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 637–646.

[10] 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*. International World Wide Web Conferences Steering Committee, 2018, pp. 1459–1468.

[11] J. J.-C. Ying, W.-C. Lee, T.-C. Weng, and V. S. Tseng, "Semantic trajectory mining for location prediction," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2011, pp. 34–43.

[12] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, "Route planning in transportation networks," in *Algorithm engineering*. Springer, 2016, pp. 19–80.

[13] Q. Gao, F. Zhou, K. Zhang, G. Trajcevski, X. Luo, and F. Zhang, "Identifying human mobility via trajectory embeddings." in *IJCAI*, vol. 17, 2017, pp. 1689–1695.

[14] F. Zhou, Q. Gao, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Trajectory-user linking via variational autoencoder." in *IJCAI*, 2018, pp. 3212–3218.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] 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.

[19] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.

[20] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," *Advances in NIPS*, 2014.

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[22] A. Nenkova, K. McKeown *et al.*, "Automatic summarization," *Foundations and Trends® in Information Retrieval*, vol. 5, no. 2–3, pp. 103–233, 2011.

[23] B. Pang, L. Lee *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[24] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, 2011, pp. 30–38.

[25] 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.

[26] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1082–1090.