Identifying Human Mobility via Trajectory Embeddings

Qiang Gao*, Fan Zhou*†, Kunpeng Zhang‡, Goce Trajcevski§, Xucheng Luo*, Fengli Zhang*
*University of Electronic Science and Technology of China, Chengdu, China
‡University of Maryland, College park
§Northwestern University, Evanston,
qianggao@std.uestc.edu.cn, fan.zhou@uestc.edu.cn, kzhang@rhsmith.umd.edu
goce@eecs.northwestern.edu, xucheng@uestc.edu.cn, fzhang@uestc.edu.cn

Abstract

Understanding human trajectory patterns is an important task in many location based social networks (LBSNs) applications, such as personalized recommendation and preference-based route planning. Most of the existing methods classify a trajectory (or its segments) based on spatio-temporal values and activities, into some predefined categories, e.g., walking or jogging. We tackle a novel trajectory classification problem: we identify and link trajectories to users who generate them in the LBSNs, a problem called Trajectory-User Linking (TUL). Solving the TUL problem is not a trivial task because: (1) the number of the classes (i.e., users) is much larger than the number of motion patterns in the common trajectory classification problems; and (2) the location based trajectory data, especially the check-ins, are often extremely sparse. To address these challenges, a Recurrent Neural Networks (RNN) based semi-supervised learning model, called TULER (TUL via Embedding and RNN) is proposed, which exploits the spatio-temporal data to capture the underlying semantics of user mobility patterns. Experiments conducted on real-world datasets demonstrate that TULER achieves better accuracy than the existing methods.

1 Introduction

Location based social networks (LBSNs), such as Instagram and Twitter, generate large volumes of data with geo-spatial tags. This enables an opportunity for better understanding and using of motion patterns in various applications [Zheng, 2015], such as: recommending POIs [Bhargava *et al.*, 2015] or next visit-location [Cheng *et al.*, 2013]; latent mobility patterns inference [Alharbi *et al.*, 2016] and influence maximization [Li *et al.*, 2016], etc.

Trajectory classification is the very initial step towards understanding user motion activities. For example, in the task of trajectory semantic inference, one should first label a sequence of spatio-temporal activities produced by a particular user as some possible patterns, e.g., still or moving, driving or walking, etc [Damiani and Güting, 2014]. Traditional trajectory classification studies mainly focus on recognizing the activity patterns and transportation modes of users, leveraging techniques such as Dynamic Bayesian Network (DBN), Hidden Markov Model (HMM) and Conditional Random Field (CRF) to incorporate historical visit locations and sequential patterns. Some existing literatures try to discover the features of individuals (or a community) based on the Latent Dirichlet Allocation (LDA) and Bayesian probabilistic graphical model for the purpose of personalized recommendations, such as tour planning [Chen *et al.*, 2016] and POIs recommendation [Bhargava *et al.*, 2015].

Despite the extensive work on mining user mobility behaviors [Dodge *et al.*, 2016; Pelekis and Theodoridis, 2014], little attention was paid to the problem of *linking trajectories* to *users* who generate them – which is important in many LBSN application-scenarios. For example, ride-sharing (bike, car) apps generate large volumes of trajectories – but the user identities are removed for the sake of privacy protection. However, correlating such trajectories with users, is also helpful in making better (i.e., more personalized and/or precise) recommendations. Moreover, it could help in identifying terrorists/criminals from sparse spatio-temporal data (e.g., the transient phone signals and check-ins).

This is what motivates us to study the *Trajectory-User Linking (TUL)* problem – which is a challenging task due to the facts that: (1) the typical number of mobile users is very large – far larger than the number of motion patterns used in the traditional trajectory classification studies; (2) the sparse trajectory data, e.g., the check-ins, may also involve noise and outliers that can affect the linking accuracy; (3) TUL problem differs from the traditional mobility pattern-recognition problems because it requires extracting and analyzing various features from trajectories [Yang *et al.*, 2015], which entails both the curse of dimensionality, as well as the invasion of user privacy.

In this paper, we propose a new method, *TULER (TUL via Embeddings and RNN)* for identifying and linking a large number of trajectories to their generating-users. More specifically, check-ins in trajectories are embedded into a low-dimensional space – similarly to approaches in word embedding, which have been demonstrated as capable of capturing word semantics in Natural Language Processing (NLP). The

†Corresponding author: Fan Zhou (fan.zhou@uestc.edu.cn)

motivation behind the check-in embedding is based on the observation that the frequency of location check-in follows a power-law distribution, similarly to the word frequency in NLP. After encoding trajectories using embedding in a *semi-supervised* way, TULER employs a sophisticated decoder model, i.e., the (stacked) LSTM [Hochreiter and Schmidhuber, 1997] or GRU [Chung *et al.*, 2014], that operates at the check-ins level to learn the underlying motion patterns of a particular user. Conditioned on both the latent motion patterns and users, TULER allows the models to simultaneously capture both *intra-* and *inter*-trajectory information – the essential intricate features required to discriminate and classify trajectories. To scale TULER to handle billions of trajectories and millions of users in LSBNs, and capitalize on the RNN inherent properties, instead of using the large input/output layers in the encoder/decoder to directly predict users, we divide trajectories and learn richer sub-trajectory patterns.

To the best of our knowledge, this is the first work to address the TUL problem and propose effective and efficient methods for solving it via semi-supervised model (TULER), leveraging both check-in embedding and Recurrent Neural Networks. In the rest of this paper, Sec. 2 reviews the related literature, and Sec. 3 gives a formal definition of TUL and the details of TULER. As presented in Sec. 4, our extensive experiments conducted on two real-world datasets show that TULER outperforms several state-of-the-art classification and trajectory similarity measuring algorithms, e.g., SVM, LDA and LCSS, on correlating trajectories to their users. Sec. 5 summarizes the paper and outlines directions for future work.

2 Related Work

Traditional trajectory pattern mining studies mainly focus on four topics: (i) detecting similarity: measuring the similarity or distance between two trajectories which is essential to cluster and query the trajectory data [Zheng, 2015]. Similarity measures often refer to Dynamic Time Warping, Longest Common Sub-Sequence (LCSS), Edit distance and Trajectory-Hausdorff Distance [Chen and Ng, 2004; Ding *et al.*, 2008]. (ii) discovering co-moving patterns: aiming at discovering the latent motion trends and gathering of crowds [Zheng *et al.*, 2013]. (iii) sequential and periodical pattern mining: finding (sub-)sequential and periodical motion patterns [Li *et al.*, 2012] which enables the travel recommendation [Chen *et al.*, 2016], life pattern understanding [Song *et al.*, 2010], trajectory search [Chen *et al.*, 2010] and next location prediction [Yang *et al.*, 2016]. Recently, a probabilistic graphical model based method called HuMoR is proposed in [Alharbi *et al.*, 2016], to learn the probability distribution of latent patterns from the trajectory data. Despite a large amount of works in semantic trajectory mining and classification, the TUL problem has never been formally defined and addressed.

Trajectory classification is one of the central tasks in understanding mobility patterns. Most existing classification works focus on labeling trajectories as different motion patterns, such as *Driving*, Biking, *Bus* and *Walking* in transportation classification [Zheng *et al.*, 2008] and *Occupied*, *Nonoccupied* and *Parked* in taxi status inference [Zhu *et al.*, 2012]. They mainly rely on extraction of spatio-temporal characteristics of trajectories. In contrast, TUL problem is more complex due to the large number of labels (users) and the interleaving sub-trajectories of different users.

Interest in natural language processing with RNNs has greatly increased in recent years, especially after introduction of memory units such as LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Chung *et al.*, 2014]. RNNs have been successfully applied to solve text classification problems [Lai *et al.*, 2015; Liu *et al.*, 2016a], where number of classes is small. Most of them are binary (e.g., IMDB[1] dataset) or have at most 20 classes (e.g., Fudan[2] dataset) – which is much less than the number of classes (users) in the TUL settings. In addition, in text classification tasks sufficient corpus is usually available, whereas TUL needs to address the sparsity issues for trajectory data.

3 Trajectory-User Linking

In this section, we first formally define the TUL problem, and then proceed with presenting the details of our proposed method.

Let $T_{u_i} = \{l_{i1}, l_{i2}, ..., l_{in}\}$ denote a trajectory generated by the user u_i during a time interval, where l_{ij} ($j \in [1, n]$) is a location point at time t_j for the user u_i, in a suitable coordinate system (e.g., longitude + latitude, or some $(x_{l_{ij}}, y_{l_{ij}})$. We refer to l_{ij} as a *check-in* in this paper. A trajectory $T_k = \{l_1, l_2, ..., l_m\}$ for which we do not know the ID of the user who generated it, is called *unlinked*. Suppose we have a number of unlinked trajectories $\mathcal{T} = \{T_1, ..., T_m\}$ produced by a set of users $\mathcal{U} = \{u_1, ..., u_n\}$ ($m \gg n$). The solution of TUL provides a mapping that will link unlinked trajectories to the users: $\mathcal{T} \mapsto \mathcal{U}$.

Given a set of unlinked trajectories, our solution (TULER) will first divide each trajectory into a set of consecutive sub-trajectories. We encode each sub-trajectory using trajectory embedding and we characterize each trajectory via trained RNN models and finally link them to users. The overview of TULER is illustrated in Figure 1.

3.1 Trajectory Segmentation

To reduce the computational complexity and capture richer semantics of moving patterns, we divide the original trajectory T_{u_i} into k consecutive sub-sequences $T^1_{u_i}, ..., T^k_{u_i}$. There exist various trajectory segmentation methods – e.g., based on semantic meaning and shape of trajectories [Zheng, 2015] – however, since this is not the core part of this work, in this paper we adopt the simple method used in [Liu *et al.*, 2016b].

3.2 Check-in Embedding

To mitigate the problem of the curse of dimensionality, we represent each check-in with a low-dimensional vector $\mathbf{v}_{l_i} \in \mathbb{R}^d$ instead of using traditional location representation method such as one-hot. Similar to word embeddings in natural language [Mikolov *et al.*, 2013], we obtain the check-in trajectory representations $\mathbf{T} \in \mathbb{R}^{|C| \times d}$ ($|C|$ is the number of

[1] http://ai.stanford.edu/amaas/data/sentiment/

[2] www.datatang.com/data/44139

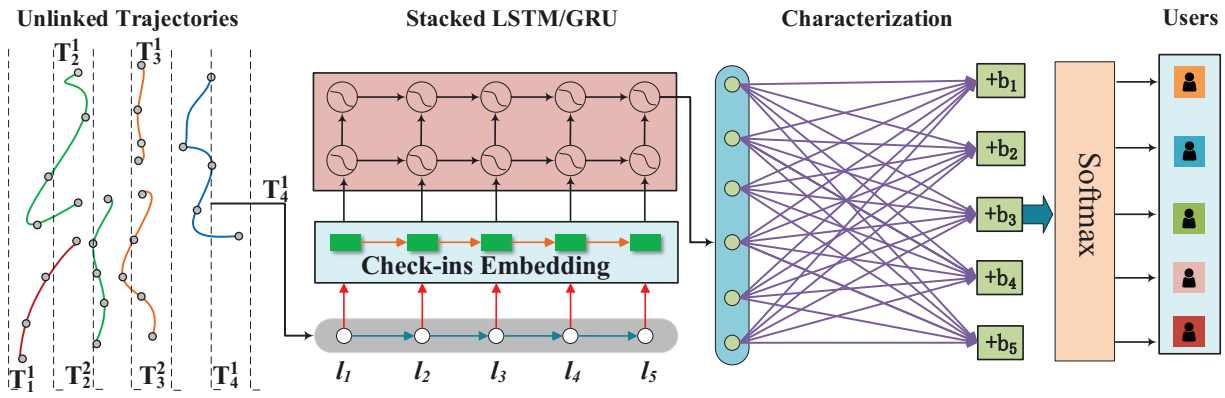
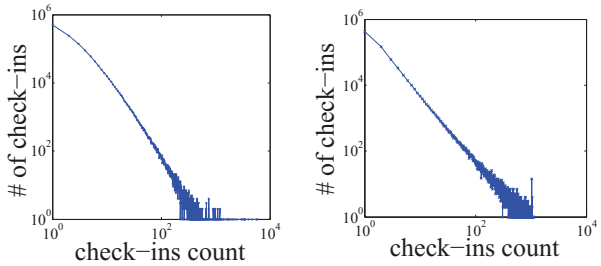


Figure 1: Overview of the TULER approach. TULER first uses trajectories to learn all check-in embeddings (low-dimension representation) $\mathbf{T} \in \mathbb{R}^{|C| \times d}$. Then combination of linked trajectory-user pairs and check-in embeddings will be used to train RNN model to characterize trajectories. An user for a given unlinked trajectory can finally be predicted.

check-ins in the dataset, d is the dimensionality in the lower space) by maximizing the probabilities of locations (check-ins) given their context in trajectories.

The reason we use check-in embedding is that the frequency of check-in locations in trajectories follows a power-law distribution, similarly to words in natural language. Figure 2 illustrates the distribution of check-ins in two real-world LSBN datasets from [Cho *et al.*, 2011]. Note that check-ins in all trajectories will be embedded into the latter RNN model, which addresses the overfitting problem when the amount of training instances is small – another motivation of using check-in embedding in our model.



(a) Gowalla (b) Brightkite

Figure 2: The frequency of check-in occurrence in two real-world trajectory datasets.

More specifically, the embedding of a location l_t is to predict its probability given its context locations $l_{t-c} : l_{t+c}$ through maximizing the log-likelihood $\sum_{t=1}^{m} \log p(l_t | l_{t-c} : l_{t+c})$, where c is the size of sliding window. The conditional probability $p(l_{t-c} : l_{t+c} | l_t)$ is defined by the softmax function as

$$p(l_{t+j} \mid l_t) = \frac{\exp\{\mathbf{v}(l_t)^{\mathrm{T}} \mathbf{v}'(l_{t+j})\}}{\sum_{l=1}^{|C|} \exp\{\mathbf{v}(l_t)^{\mathrm{T}} \mathbf{v}'(l)\}}$$

where $\mathbf{v}(l)$ and $\mathbf{v}'(l)$ are, respectively, the input and output vector representations of check-in l. We now can estimate the probability of a trajectory $T = \{l_1, l_2, ..., l_k\}$ by

$$p(\mathbf{v}(l)) = \prod_{i=1}^{m} p(\mathbf{v}(l_i) | C(l_i, l))$$

where $C(l_i, l)$ is the context of location l_i in a trajectory T, and probability $p(\mathbf{v}(l_i)|C(l_i, l)))$ is approximated by predicting $\mathbf{v}(l_i)$ with

$$\begin{aligned} p(\mathbf{v}(l_i)|C(l_i, l))) &= \prod_{l' \in C(l_i, l)} p(\mathbf{v}(l_i)|\mathbf{v}(l')) \\ &= \prod_{l' \in C(l_i, l)} \frac{\exp\{\mathbf{v}(l_i) \cdot \mathbf{v}(l')\}}{\sum_{l'' \in C} \exp\{\mathbf{v}(l'')\mathbf{v}(l')\}} \end{aligned}$$

3.3 Trajectory Characterization

Even if we split the original trajectories into short time-span sub-trajectories, there still exist many dense check-ins in some sub-trajectories. To process these long-term variable-length location sequences, TULER employs several variants of well-known RNN models, i.e., LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Chung *et al.*, 2014], as well as the stacked and bidirectional RNNs, to control the input and output of trajectory embeddings. For the sake of simplicity but without loss of generality, we discuss next in brief the LSTM and GRU model used in TULER.

TULER with LSTM

For the sub-trajectory $T = \{l_1, l_2, ..., l_k\}$, let h_{t-1}, h_t and $\tilde{h}_t$ denote the last, current and candidate embedding state, respectively. The LSTM model used in TULER is implemented as follows:

$$\begin{aligned} i_t &= \sigma(W_i \mathbf{v}^t(l_i) + U_i h_{t-1} + V_i c_{t-1} + b_i) \\ f_t &= \sigma(W_f \mathbf{v}^t(l_i) + U_f h_{t-1} + V_f c_{t-1} + b_f) \\ o_t &= \sigma(W_o \mathbf{v}^t(l_i) + U_o h_{t-1} + V_o c_t + b_o) \end{aligned} \quad (1)$$

where i_t, f_t, o_t and b_* are respectively the input gate, forget gate, output gate and bias vector, σ is a logistic sigmoid function, matrices W, U and V ($\in \mathbb{R}^{d\times d}$) are the different gate parameters, $\mathbf{v}^t(l_i)$ is the embedding of the check-in location l_i. The memory cell c_t is updated by partially replacing the existing memory unit with a new cell $\mathbf{c}_t$ as

$$c_t = f_t c_{t-1} + i_t \tanh(W_c \mathbf{v}(l_i) + U_c h_{t-1} + b_c) \quad (2)$$

The trajectory embedding is then updated by

$$h_t = o_t \odot \tanh(c_t) \quad (3)$$

where $\sigma(\cdot)$ and $tanh(\cdot)$ refer to the sigmoid and hyperbolic tangent function, and $\odot$ is the entry-wise product.

TULER with GRU

Similar to LSTM, TULER with GRU models the trajectory embedding using extra gating units, but without separated memory cells. Formally, we update the state of h_t by a linear interpolation between the last state h_{t-1} and the candidate state $\tilde{h}_t$ as

$$h_t = (1 - g_t)h_{t-1} + g_t \tilde{h}_t$$

where g_t is the update gate which decides how much the unit updates its activation by

$$g_t = \sigma(W_z \mathbf{v}^t(l_i) + U_z h_{t-1})$$

The candidate state $\tilde{h}_t$ is computed similarly to traditional RNN unit

$$\tilde{h}_t = \tanh(W \mathbf{v}^t(l_i) + U(s_t \odot h_{t-1}))$$

where s_t is a set of reset gates and is computed similarly to updating the gate

$$s_t = \sigma(W_s \mathbf{v}^t(l_i) + U_s h_{t-1})$$

Variants

Several variants of TULER are stacked LSTM/GRU and Bidirectional LSTM [Sutskever *et al.*, 2014]. In stacked LSTM/GRU, the hidden state of a unit in layer n is used as input to the unit in layer *n+1* at the same time step. The goal of multi-RNN stacking is to capture longer check-in dependency of a trajectory. However, the training time of stacked TULER increases exponentially with the number of layers. A Bidirectional LSTM can be used by running two LSTMs in parallel: one is on the sequential check-in embedding vectors, and the other is on the reverse embedding vectors. Apparently, this may substantially reduce the time required for training the model compared to the general and stacked LSTM/GRU based TULER. The performance of using different types of *deep* TULER will be investigated and discussed in the experiment section.

3.4 Trajectory-User Linking

To link trajectories to their users, the trajectory representations $\tilde{l}_{u_i}$ generated by the TULER are fed into softmax as

$$\begin{aligned} \tilde{l}_{u_i} &= \text{softmax}(W_{u^i} h_{u^i} + b_{u^i}) \\ &= \frac{\exp\{\mathbf{v}(l_i)^{\mathrm{T}} \kappa_i\}}{\sum_{j=1}^{|u|} \exp\{\mathbf{v}(l_i)^{\mathrm{T}} \kappa_j\}} \end{aligned}$$

where $\kappa = \{W, U, V, b\}$ is the parameter set needs to learn.

Now we learn the parameters κ w.r.t the objective function. Given a trajectory sequence $l_u = l_1, l_2, ..., l_m$ of a user u, we train the TULER to maximize the log-likelihood with respect to κ:

$$u(l_u) \mapsto \sum_{l_u \in \mathbb{U}} \log p(u|l_u, \kappa)$$

where u and $\mathbb{U}$ are respectively the ground-truth user of trajectory l_u and the training data. At each step, *stochastic gradient descent* is used to estimate the parameter set κ

$$\kappa \leftarrow \kappa + \alpha \frac{\partial \log p(u|l_u, \kappa)}{\partial \kappa}$$

where α is the learning rate. Finally, we aim at minimizing following cost function

$$\Phi(l_{u_i}, \tilde{l}_{u_i}) = -\sum_{i=1}^{|l|} \sum_{j=1}^{|u|} u \log(\tilde{l}_i^j)$$

where $\tilde{l}_i^j$ is the predicted trajectory embedding vector.

3.5 Optimization

To alleviate the problem of overfitting inherent to embedding and RNN, we apply a variational inference based dropout technique [Gal and Ghahramani, 2016] in TULER, which repeats the same dropout mask without deteriorating the performance.

During the trajectory embedding, we randomly drop some check-ins, e.g., trajectory sequence $l_1, l_3, l_5, l_3, l_7, ...$ might become $l_1, , l_5, , l_7, ...$ or $l_1, l_3, , l_3, l_7....$ That is, at a time step t, the same check-ins in the training data may be dropped and the corresponding row in the embedding matrix $\mathbf{T} \in \mathbb{R}^{|C|\times d}$ would be set to zero.

The dropout variant with respected to eq.(1) and eq.(2) of RNN in TULER here is defined similarly to [Gal and Ghahramani, 2016]

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ \tilde{c}_t \end{pmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\begin{pmatrix} W \\ U \\ V \end{pmatrix} \cdot \begin{pmatrix} \mathbf{v}^t(l_i) \odot \mathbf{r}_{\mathbf{v}} \\ h_{t-1} \odot \mathbf{r}_h \\ c_{t-1} \odot \mathbf{r}_c \end{pmatrix} + \begin{pmatrix} b_i \\ b_f \\ b_o \\ b_{\tilde{c}} \end{pmatrix} \right)$$

where $\mathbf{r}_*$ are random masks repeated at all time steps.

After two models: embedding and RNN are trained, a TULER model is constructed. The embedding layer of TULER encodes the semantics of check-ins and is initialized with the corresponding trained weights of W_e. The stacked or bidirectional layer of RNN and the softmax of the model are initialized randomly while all parameters are fine-tuned on the labeled data.

4 Evaluation

We now present our experiments, comparing TULER with several baseline methods on two public datasets. Source code, datasets and implementation details are available online at https://github.com/gcooq/TUL.

Table 1: Dataset description and statistics. U: the number of users; S/T: the number of trajectories for training and testing; C: the number of check-ins; $\overline{R}$: average length of trajectories (before segmentation); t_r: the range of the number of check-ins in sub-trajectories.

Dataset	U	S/T	C	$\overline{R}$	t_r
Gowalla	201	17,654/2,063	10,956	219	[1,131]
Brightkite	92	17,934/2,039	2,120	471	[1,184]

Table 2: Parameters used in TULER and baselines.

Parameters	We choose	Possible Choices
Dimensionality	250	100-300
Hidden size	300	250-1000
Learning rate	0.00095	0.00085-0.1
Dropout rate	0.5	0-1
Stacked TULER	2	≥ 2
LDA Matrix solver	SVD	SVD, LSQR, *etc*
SVM Kernel	Linear	Linear, RBF, *etc*

4.1 Datasets

To show the performance of TULER and the comparison with some existing methods, we conduct our experiments on two publicly available LBSN datasets: Gowalla and Brightkite [Cho *et al.*, 2011]. Both contain check-ins and users of the corresponding check-in trajectories (social connection links). We randomly select 201 and 92 users from Gowalla and Brightkite respectively. For each user, we concatenate all check-in locations to form a trajectory which will be further divided into sub-trajectories based on the time interval we define (i.e., 6 hours). Table 1 depicts the stats of two datasets.

4.2 Empirical Results

Before comparing the performance among various proposed algorithms and baselines, we pictorially show several trajectories from our dataset and their predicted users using TULER in Figure 3. In Figure 3(a) and 3(c), TULER successfully identified the trajectories produced by user No.119 and No.19 in two datasets, respectively. However, for the trajectories in Figure 3(b) and 3(d), TULER fails to link sub-trajectories to their users: user No.79 and No.97 in two datasets (maked as red $\times$), mainly because of the extremely sparse location sequences, e.g., they contain only 1 or 2 check-ins. This is a still an open problem for trajectory-user linking. That is, how can we identify the sparse check-in trajectories? A natural solution is to involve more extra attributes of the trajectories, e.g., timestamps, to reduce the complexity when searching trajectory users.

4.3 Baselines and Metrics for Comparison

We compare TULER with three state-of-the-art approaches from the field of trajectory similarity measurement and trajectory classification. In addition, TULER itself consists of five variants, i.e., one layer of RNN (TULER-LSTM and TULER-GRU), stacked RNNs (TULER-LSTM-S and TULER-GRU-S) and Bidirectional LSTM (Bi-TULER). The baselines are:

- **LCSS**: The Longest Common Sub-Sequence method [Ying *et al.*, 2011] is to match the longest common subsequence between two sequences using dynamic programming. LCSS is also a widely used representative method for measuring the trajectory similarity. We apply LCSS to our TUL problem by sequentially searching all trajectories in the training set for any given testing sub-trajectory to find the corresponding user.
- **LDA**: LDA-based (Linear Discriminant Analysis) methods have shown good performance in text classification. We employ Bag-of-Words (BoW) [Mikolov *et al.*, 2013] to embed the trajectory into one-hot vectors following the method for text embedding proposed in [Lai *et al.*, 2016], and Singular Value Decomposition (SVD) is used to decompose the within-class scatter matrix – SVD is especially suitable for the data with a large number of features such as the trajectory data. Note that other matrix solvers such as least squares and eigenvalue decomposition are also compared but omitted due to their lower performance than SVD in our experiments.
- **SVM**: In SVM implementation, linear kernel is used for solving TUL problem due to its better performance than other kernels such as RBF kernel and Gaussian kernel in our experiments. BoW is used to embed the trajectories.

TUL tries to predict *top-k* candidate users for each testing trajectory. In this paper, we use ACC@K and macro-F_1 to measure the performance, which are common metric in information retrieval area. Specifically, ACC@K is to evaluate the trajectory-user linking accuracy as

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

and macro-F_1 is the harmonic mean of the precision and recall:

$$\text{macro-}F_1 = \frac{2 \times P^* \times R^*}{P^* + R^*}.$$

where P^* and R^* are precision and recall averaged across all classes (users in TUL).

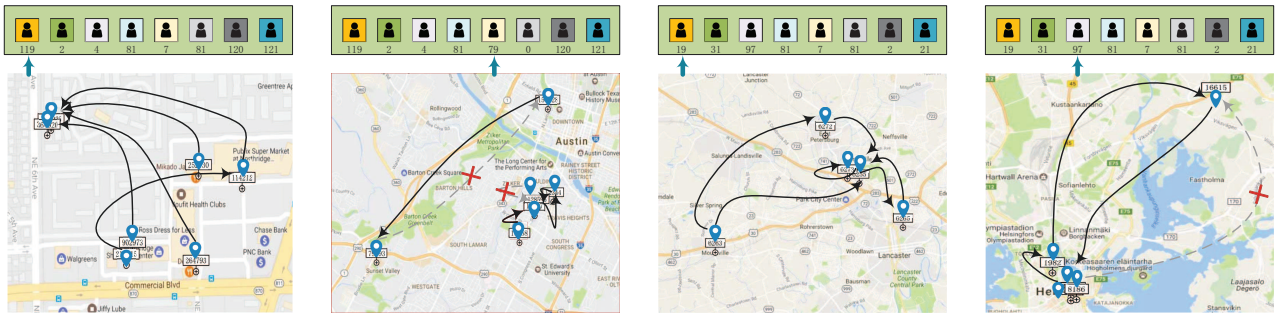
Performance comparison

Table 2 shows commonly used possible parameter values and the optimal choices tuned for both TULER and baseline approaches, which are used for the rest unless specified.

Table 3 and 4 respectively summarize the performance comparison among various TULER and baselines on two datasets, where the best method is shown in **bold**, and the second best is shown as underlined.

On Gowalla dataset, our model TULER with Bidirectional LSTM consistently outperforms other methods in terms of accuracy, while the TULER with one layer of LSTM achieves the best result with respect to the Macro-F_1 metric. Specifically, Bi-TULER yields 36.9%, 28.1% and 13.8% improvement compared to LCSS, LDA and SVM on ACC@5 metric.

Similar performance by TULER also holds on the Brightkite dataset. For example, the best and the second best results on the accuracy are achieved by TULER (TULER-LSTM and Bi-TULER respectively). Although LDA obtains



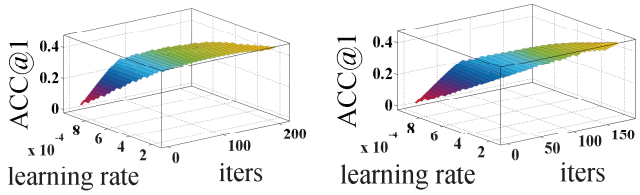
(a) Observation on Gowalla. (b) Observation on Gowalla. (c) Observation on Brightkite. (d) Observation on Brightkite.

Figure 3: Examples of using TULER to predict users of trajectories.

the highest Macro-F_1, TULER based methods achieve comparable results.

A counter-intuitive result is that stacked TULERs (such as stacked LSTM and GRU), primarily seeking to capture characteristics of longer trajectory sequences, fall behind the one layer TULER, as well as the Bi-TULER, although they all outperform baselines. One possible reason is that the trajectory segmentation in TULER has truncated the original long trajectories to short sub-sequences For longer trajectories, stacked TULER works the best – corresponding results are omitted for the simplification purpose.

Model robustness



(a) Gowalla. (b) Brightkite.

Figure 4: Parameter sensitivity of TULER-LSTM.

Some parameters like the number of iterations, learning rate might have significant impact on the model performance. Figure 4 shows that the accuracy of TULER is proportional to the number of iterations. In addition, a small number of learning rate (e.g., 0.95×10^{-3}) can obtain a higher classification accuracy.

5 Conclusions

We presented a new way to mine human mobility pattern – Trajectory-User Linking (TUL), which aims at identifying users of location-based trajectories (e.g., check-ins). We developed an RNN based model (coupling the check-ins) called TULER which, unlike traditional trajectory similarity measurement and classification models, is designed to capture the dependency of check-ins and to infer the latent patterns of users. TULER achieves the significant performance improvement, when compared to existing methods. At its current stage, TULER can be augmented with other social-networks information and, as part of our future work, we will investigate how to incorporate community moving patterns to further improve the performance of TULER.

Table 3: Performance comparison on the Gowalla dataset.

Method \ Metric	ACC@1	ACC@5	Macro-F_1
LCSS	32.65	46.13	27.02
LDA	37.86	49.28	34.08
SVM	41.25	55.50	34.32
TULER-LSTM	45.03	63.15	**35.77**
TULER-GRU	41.06	60.37	31.46
TULER-LSTM-S	41.68	57.03	32.43
TULER-GRU-S	40.10	59.08	32.37
Bi-TULER	**45.70**	**65.68**	35.56

Table 4: Performance comparison on the Brightkite dataset.

Method \ Metric	ACC@1	ACC@5	Macro-F_1
LCSS	30.12	39.13	23.02
LDA	40.50	53.38	**39.38**
SVM	42.07	61.46	36.59
TULER-LSTM	**45.00**	**64.64**	38.18
TULER-GRU	43.29	62.49	34.86
TULER-LSTM-S	43.19	61.56	38.71
TULER-GRU-S	41.38	60.68	38.71
Bi-TULER	44.91	63.91	38.20

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No.61602097, No.61672135 and No.61472064), NSF grants III 1213038 and CNS 1646107, ONR grant N00014-14-10215 and HERE grant 30046005, Sichuan Science-Technology Support Plan Program (No.2016GZ0065), and the Fundamental Research Funds for the Central Universities (No.ZYGX2015J072).

References

[Alharbi *et al.*, 2016] Basma Alharbi, Abdulhakim Qahtan, and Xiangliang Zhang. Minimizing user involvement for

learning human mobility patterns from location traces. In *AAAI*, 2016.

[Bhargava *et al.*, 2015] Preeti Bhargava, Thomas Phan, Jiayu Zhou, and Juhan Lee. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *ACM WWW*, 2015.

[Chen and Ng, 2004] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *VLDB*, 2004.

[Chen *et al.*, 2010] Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, and Xing Xie. Searching trajectories by locations: An efficiency study. In *ACM SIGMOD*, 2010.

[Chen *et al.*, 2016] Dawei Chen, Cheng Soon Ong, and Lexing Xie. Learning points and routes to recommend trajectories. In *ACM CIKM*, 2016.

[Cheng *et al.*, 2013] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. Where you like to go next: successive point-of-interest recommendation. In *IJCAI*, 2013.

[Cho *et al.*, 2011] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *ACM SIGKDD*, 2011.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, Kyung Hyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Eprint Arxiv*, 2014.

[Damiani and Güting, 2014] Maria Luisa Damiani and Ralf Hartmut Güting. Semantic trajectories and beyond (tutorial). In *IEEE 15th International Conference on Mobile Data Management, MDM 2014, Brisbane, Australia, July 14-18, 2014 - Volume 2*, pages 1–3, 2014.

[Ding *et al.*, 2008] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn J. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB*, 1(2), 2008.

[Dodge *et al.*, 2016] Somayeh Dodge, Robert Weibel, Sean Charles Ahearn, Maike Buchin, and Jennifer A. Miller. Analysis of movement data. *International Journal of Geographical Information Science*, 30(5):825–834, 2016.

[Gal and Ghahramani, 2016] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Lai *et al.*, 2015] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, 2015.

[Lai *et al.*, 2016] Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, 2016.

[Li *et al.*, 2012] Zhenhui Li, Jingjing Wang, and Jiawei Han. Mining event periodicity from incomplete observations. In *ACM SIGKDD*, 2012.

[Li *et al.*, 2016] Ji Li, Zhipeng Cai, Mingyuan Yan, and Yingshu Li. Using crowdsourced data in location-based social networks to explore influence maximization. In *IEEE INFOCOM*, 2016.

[Liu *et al.*, 2016a] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *IJCAI*, 2016.

[Liu *et al.*, 2016b] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: a recurrent model with spatial and temporal contexts. In *AAAI*, 2016.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Computer Science*, 2013.

[Pelekis and Theodoridis, 2014] Nikos Pelekis and Yannis Theodoridis. *Mobility Data Management and Exploration*. Springer, 2014.

[Song *et al.*, 2010] Chaoming Song, Zehui Qu, and Nicholas Blumm. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[Yang *et al.*, 2015] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015.

[Yang *et al.*, 2016] Cheng Yang, Maosong Sun, Wayne Xin Zhao, Zhiyuan Liu, and Edward Y Chang. A neural network approach to joint modeling social networks and mobile trajectories. *arXiv preprint arXiv:1606.08154*, 2016.

[Ying *et al.*, 2011] Jia Ching Ying, Wang Chien Lee, Tz Chiao Weng, and Vincent S. Tseng. Semantic trajectory mining for location prediction. In *ACM Sigspatial*, 2011.

[Zheng *et al.*, 2008] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei Ying Ma. Understanding mobility based on gps data. In *ACM UbiComp*, 2008.

[Zheng *et al.*, 2013] Yu Zheng, Nicholas Jing Yuan, Kai Zheng, and Shuo Shang. On discovery of gathering patterns from trajectories. *IEEE Transactions on Knowledge & Data Engineering*, 26(8):242–253, 2013.

[Zheng, 2015] Yu Zheng. Trajectory data mining: An overview. *Acm Transactions on Intelligent Systems & Technology*, 6(3):1–41, 2015.

[Zhu *et al.*, 2012] Yin Zhu, Yu Zheng, Liuhang Zhang, Darshan Santani, Xing Xie, and Qiang Yang. Inferring taxi status using gps trajectories. *Computer Science*, 2012.