Semi-supervised Trajectory Understanding with POI Attention for End-to-End Trip Recommendation

FAN ZHOU and HANTAO WU, University of Electronic Science and Technology of China, China
GOCE TRAJCEVSKI and ASHFAQ KHOKHAR, Iowa State University, USA
KUNPENG ZHANG, University of Maryland, College Park, USA

Trip planning/recommendation is an important task for a plethora of applications in urban settings (e.g., tourism, transportation, social outings), relying on services provided by Location-Based Social Networks (LBSN). To provide greater context-awareness in trajectory planning, LBSNs combine historical trajectories of users for generating various hand-crafted features—e.g., geo-tags of photos taken by tourists and textual characteristics derived from reviews. Those features are used to learn tourists' preferences, which are then used to generate a travel plan recommendation. However, many such features are extracted based on prior knowledge or empirical analysis specific to particular datasets, rendering the corresponding solutions not to be generalizable to diverse data sources. Thus, one important question for managing mobility is how to learn an accurate tour planning model based solely on POI visits or user check-ins and without the efforts of hand-crafted feature engineering. Inspired by recent successes of deep learning in sequence learning, we develop a solution to the tour planning problem based on the semi-supervised learning paradigm. An important aspect of our solution is that it does not involve any feature engineering. Specifically, we propose the Trip Recommendation method via trajectory Encoder and Decoder—a novel end-to-end approach encoding historical trajectories into vectors, while capturing both the intrinsic characteristics of individual POIs and the transition patterns among POIs. We also incorporate historical attention mechanism in our sequence-to-sequence trip recommendation task to improve the effectiveness. Experiments conducted on multiple publicly available LBSN datasets demonstrate significantly superior performance of our method.

CCS Concepts: • **Information systems** → **Location based services**; *Temporal data*; *Recommender systems*; • **Computer systems organization** → *Neural networks*;

Additional Key Words and Phrases: Trip recommendation, semi-supervised learning, encoder-decoder, recurrent neural networks, attention mechanism

ACM Reference format:
Fan Zhou, Hantao Wu, Goce Trajcevski, Ashfaq Khokhar, and Kunpeng Zhang. 2020. Semi-supervised Trajectory Understanding with POI Attention for End-to-End Trip Recommendation. *ACM Trans. Spatial Algorithms Syst.* 6, 2, Article 13 (February 2020), 25 pages.
https://doi.org/10.1145/3378890

This work was supported by National Natural Science Foundation of China (Grant No. 61602097), NSF Grants No. III 1213038 and No. CNS 1646107, and ONR Grant No. N00014-14-10215.

Authors' addresses: F. Zhou and H. Wu, University of Electronic Science and Technology of China, No. 4, Section 2, North Jianshe Rd., Chengdu, Sichuan, 610054, China; emails: fan.zhou@uestc.edu.cn, marlboro.wu@foxmail.com; G. Trajcevski (corresponding author), A. Khokhar, Iowa State University, Ames, IA; emails: gocet25@iastate.edu, ashfaq@iastate.edu; K. Zhang, University of Maryland, College Park, College Park, MD; email: kzhang@rhsmith.umd.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.

2374-0353/2020/02-ART13 $15.00
https://doi.org/10.1145/3378890

1 INTRODUCTION

Location Based Social Networks (LBSN), e.g., Twitter, Instagram, and Foursquare, generate massive amounts of user mobility data with time-stamped geo-tagging on a daily basis. Mining and learning useful spatio-temporal information from such data is desirable in many applications of Location-Based Services (LBS) [63] that are of high relevance in urban mobility management. But few examples are identifying human moving patterns [46, 74], urban traffic management [85], crime prediction [24], vibrant community identification [70], and trajectory-based queries [4]. Location-based data generated by various other sources, such as sensors, GPS, IoT devices, and so on, are also widely used in several LBS applications, such as traffic flow pattern detection [88]. Even the "transactional trajectories" formed by historical activities (e.g., ATM withdrawals, in-store or online purchases, etc.) are also valuable in understanding human behaviors in the context of identifying similar mobility behaviors among an evolving group of customers or groups of employees that have similar careers [19].

Trajectory recommendation [8] or tour planning [41] is one of the most popular and computationally challenging tasks in LBS [40]. The majority of the existing works that also involve semantic-based contexts in addition to the mere location-in-time data focus on deriving (i.e., handcrafted) various specific features for trip recommendation, such as user trajectory patterns, popularity of POIs, preferences of individuals, trip constraints, and so on. For example, Lu et al. [48] leverage travel clues recovered from on-line geo-tagged photos to suggest customized travel route plans. Lim et al. [41] build a personalized tour recommendation using POI popularity and user interest/preferences extracted from geo-tagged photos of visited scenery. Chen et al. [8] recommend a sequence of POIs with a probabilistic transition model using various features learned from user past behavior and statistics with respect to POIs, including POI ranking, category, and popularity. More recently, Wen et al. [72] propose a travel recommendation method by extracting keywords representing POIs from user historical mobility records and social interactions.

Modeling relationships among POIs and users is a difficult task and has been a major cause for various deficiencies inherent in the existing trajectory planning approaches. There are several reasons that are at the core of such deficiencies: (1) To our knowledge, there exist no systematic guides as to how to select and extract features from heterogeneous data, much less to consider their various combinations—for instance, how to choose between geo-tagged photos and social texts for trip planning. Even if one could measure their performance before making the choice, how can we decide which feature extraction algorithms to use from the multitude of available ones? (2) Different features have different impact on the performance, sometimes even leading to effects completely opposite from the expectations based on similar use-cases. (3) Combining multiple features may potentially improve the accuracy of trip planning—however, it is not straightforward to quantify the contribution of various features nor to generalize a particular approach.

Traditional trip recommendation systems often model the POI transition relying on Markov Chains (MC)—a promising foundation that adequately captures temporal relations in data—as exemplified by the Factorizing Personalized Markov Chains (FPMC) [11, 57] and ranking-based MC transition [8]. However, these models are built upon a strong assumption of independence among non-adjacent POIs—which, in turn, limits their performance on capturing long-term dependencies of POIs. As an example, consider a description of a semantic trajectory (cf. Reference [53]) given by: *museum* → *cafe* → *historical_site* → · · · → *park* → *culture_center*. In it, the long-term visiting pattern such as *museum* → *cafe* → *historical_cite* cannot be straightforwardly captured by MC-based methods when generating the next POI of *culture_center*. In addition, for some patterns (e.g., repeated check-ins) that could be important in particular application settings, it is also not straightforward how to capture them by the use of MC-based methods.

To overcome the kinds of deficiencies outlined above, we propose a novel recurrent neural networks-based tour/trajectory planning approach—*Trip Recommendation* via trajectory *Encoder and Decoder (TRED)*. The encoder network encodes the trajectory features to a hidden representation, and the decoder network takes the representation and sequentially predicts the trajectory pattern. The proposed solution works in an end-to-end manner without additional overheads of feature engineering (i.e., requiring domain knowledge to extract important features) for the POI and user profile and adapts to learn long-term dependencies and possible regular moving patterns in POIs. By leveraging unlabeled data via the autoencoder network training, TRED also efficiently alleviates the data sparsity issue when only using labeled data. In addition, the performance of TRED is improved by introducing: (1) a trajectory attention mechanism—i.e., augmenting the decoder to properly incorporate more influential POIs (cf. Section 3.2) and (2) a reconstruction operation in the final beam search inferring step (cf. Section 4.2) to further increase the efficiency.

The main contributions of this work are summarized as follows:

- We propose a data-driven method to address the trip recommendation problem, where a semi-supervised trajectory-to-trajectory learning model is developed to capture the latent semantics of moving patterns in the LBSN data.
- We propose a deep learning–based approach for trip recommendation in an end-to-end manner that does not require hand-crafted features of POIs and user activities. Our method can be easily generalized to different LBSN applications and is especially suitable for the datasets without additional features, e.g., POI popularity, POI category, queuing time, and so on. This property also eases the burden of computation in various feature-based matrix factorizing and multiplication.
- Our proposed method can preserve both previous and subsequent transition patterns among POIs and thus overcome the long-term dependency problem. By introducing the idea of attention on historical trajectory, our model significantly improves the prediction performance when generating planned POI sequences.
- We have conducted extensive experimental evaluations on several real-world as well as synthetic datasets, and the results demonstrate that our proposed method is effective and efficient. It achieves more correct recommendation (on the average, 22% in terms of F_1 score) and more accurate orderness (on the average, 38% in terms of pairs-F_1 score) when compared to the state-of-the-art baselines [8, 41].

We note that certain aspects (i.e., trajectory embedding and autoencoders) have been used in our earlier works [21, 92]—addressing the problem of trajectory to user linkage (TUL) and overcoming the data sparsity issue. This article, however, has qualitatively novel contributions: Rather than identifying human mobility patterns from spatio-temporal data, we address the trip planning problem with a unique approach for capturing semantic aspects of the trajectories while avoiding feature(s) engineering. Specifically, we address the trip recommendation problem by learning the trajectory context in a sequence to sequence manner with an added novelty of proposing a trajectory attention model for learning the transition patterns and the importance of historical check-ins, as well as an efficient way for alleviating the suboptimality problem when inferring the trip.

In the rest of this article, we review the related work in Section 2, and in Section 3 we formalize the problem of trajectory modeling and trip recommendation along with its mapping on neural networks, and we also describe the overall processing framework (encoding/decoding and attention). In Section 4, we present the details of the learning methodology and trip planning with trajectory attention. Experimental evaluations illustrating the accuracy and efficiency of our proposed method are presented in Section 5, and Section 6 concludes the article.

2 RELATED WORK

There is a large body of works addressing location recommendation and trip planning problems and, in the following, we overview the three most related categories.

2.1 POI Recommendation

Collaborative Filtering (CF) is widely used for recommending POIs, including user-based CF [80], time-aware CF [82], social influence-based CF [79], geographical CF [20], and information coverage-based CF [9]. Matrix Factorization (MF) is another line of CF method that factorizes a user-POI interest matrix to explore user preferences. Cheng et al. fuse MF with geographical and social influence for POI recommendation in LBSNs [10]. Lian et al. [38] augment users' and POIs' latent factors in the factorization model with activity area vectors of users and influence area vectors of POIs to deal with the challenge of matrix sparsity. Liu et al. [42] propose a geographical probabilistic factor analysis model leveraging the Bayesian non-negative matrix factorization. Users' interest [36, 37], check-in behavior [81], user preference rankings [13], metric embedding [18], POI categories [28], and POI characteristics [30] are also exploited for POI recommendation. However, these methods require a number of hand-crafted features to construct the preference matrix. They are rather inefficient due to frequently involving matrix factorization operation. Sequential influence among POIs has been incorporated into POI recommendation, mainly using MC for quantifying POI transition and predicting the next location. FPMC [57] extends MC via factorization of the probability transition matrix. Cheng et al. [11] take into account user movement constraints and propose a location constrained FPMC method for location prediction. However, MC-based models make a strong independence assumption among POIs except the adjacent two, which fails to capture long-term dependencies of POIs. An experimental evaluation [44] has been conducted to evaluate 12 representative POI recommendation models and reports many interesting findings, including geographical information [37, 38] and implicit feedback of user preference [45], which are the most important factors for improving the performance of POI recommendation. Recently, deep learning has been widely leveraged for POI recommendation. For example, Yang et al. [77] develop a context embedding framework combined with CF for POI recommendation. Zhao et al. [89] present a geo-temporal sequential embedding rank model for POI recommendation. In addition, while most of the existing works focus on recommendations for individual users, techniques to provide recommendations to groups of users are scarce. Ayala-Gomez et al. [2] propose a GeoGroup-Recommender (GGR), a class of hybrid recommender systems that combine the group geographical preferences using Kernel Density Estimation, category and location features, and group check-ins outperforming a large number of other recommender systems [2]. As the availability of social information (e.g., friendships), Allison et al. [6] proposed social Poisson factorization, a Bayesian model that incorporates the users latent preferences for items with the latent influences of their friends for personalized recommendation. Our approach provides a unique way to entangle the intrinsic POI-related attributes with the transitions of the locations among (i.e., the sequence of visiting of) the POIs.

2.2 Trip Planning

Trip planning [22, 23] leverages the spatio-temporal check-in data for recommending a sequence of POIs, and most of the existing works use heuristic combination of location and route [47, 48, 90] to model the planning problem. Wei et al. construct popular routes from uncertain trajectories [71]. Kurashima et al. [34] plan trips based on user interest and frequently traveled routes using the Markov model. Yuan et al. [82] exploit time-aware CF to recommend POIs for a given user at a specified time in a day. TripBuilder [5] is a framework for personalized tour planning, modeled as an instance of the generalized maximum coverage problem. TripRouter [30] is a heuristic approach that guides the search toward the destination location and uses a backward checking mechanism to boost the constructed time-sensitive routes, where popularity, visiting order, visiting time and transit time are combined to model the POIs. TripPlanner [7] combines LBSN and taxi GPS footprints for interactive and traffic-aware trip planning. Existing trip planning algorithms are feature based that require carefully selecting and comparing feature combinations and thus are hard to generalize to different scenarios.

Recently, Lim et al. [41] derived a relative measure of time-based user interest using duration of a visit. The tour recommendation was formulated as an orienteering problem [27] that maximizes a global reward based on POI popularity and users' interests, while adhering to the budget constrained by the travel-time and personalized visit duration. Zhang et al. [83, 84] tailor personal preference from user behavior and leverage constraints (e.g., POI availability and uncertain traveling time) to prune the state space in solving the orienteering problem. Chen et al. [8] simultaneously use both POIs and routes from historical behavior and trajectories, and leverage MC to model POI→POI transition. Lim et al. [39] study recommending personalized itineraries while minimizing queuing times in popular and interesting attractions. This is a NP-hard problem that includes time-dependent queuing times and is solved by an adapted Monte Carlo tree search algorithm that considers attraction popularity, user interest, and queuing times as reward. Similarly to their earlier work, this approach is feature driven and fails to capture intrinsic moving patterns in trajectories, although a state-of-the-art performance on trip planning was demonstrated. In contrast, our proposed method TRED in this article is the first data-driven trip planning algorithm without any further feature engineering—neither POIs nor routes—and it can be easily incorporated into existing framework, which opens a new perspective on modeling the trip planning problem.

2.3 RNN-based Trajectory Modeling

Recurrent neural networks (RNN) have been successfully applied in many sequential data, such as machine translation [3], click prediction [87] and text classification [35]. ST-RNN [43] models spatio-temporal data using RNN for next location prediction. The problem of identifying human trajectory patterns has been investigated in Reference [21] using various RNN-based models. Wu et al. [75] make full advantage of the strength of RNN to capture variable-length sequence and meanwhile to address constraints of topological structure on trajectory modeling. In computer vision, Alahi et al. learn general human movement to predict their future trajectories [1].

Our proposed model is different from these works in that: (1) we model the geo-location learning problem in LBSN with the encoder-decoder mechanism, (2) we propose a semi-supervised learning approach for capturing sequential behaviors and patterns of trajectories in an end-to-end manner, and (3) we demonstrate that the proposed data-driven method can efficiently tackle the trip planning problem.

We note that related problems (e.g., sequence route planning, pattern queries, etc.) have been studied by the researchers from spatio-temporal data management community [16, 17, 58, 64, 68]. More recently, the variants of group patterns and sequence routes have been explored [59, 61], along with extensions that include hierarchical semantic similarity of the points of stopping/visiting [62]. However, these works are orthogonal to the problem(s) that we pursue: namely, they are focused on efficient query processing from given datasets, whereas we are focusing on the problem of learning about the relationships in sequentiality of POIs visits from users' check-ins.

3 MOTION MODELING AND BASIC PROCESSING FRAMEWORK

Given a set of trajectories $\mathbf{T} = \{T_1, T_2, \ldots, T_n\}$, where T_* is a sequence of POIs generated by a particular user and it can be denoted as $T_* = \{l_1, l_2, \ldots, l_m\}$, let $\mathbf{T}^r$ denote the training data consisting of a set of variable-length trajectories generated by certain users in a particular area (e.g., in a city or a scenic spot). The goal of trajectory recommendation is to plan a trip with length L: $\tau = \{l_1, l_2, \ldots, l_L\}$, where l_i denotes a location or POI, and the start ($l_1 = l_s$) and end ($l_L = l_e$) points are specified as parameters in the recommendation algorithm.

Similarly to word embedding in natural language [50] and other POI embedding techniques [18, 21], we obtain POI vector representations $\mathbf{P} \in \mathbb{R}^{|C| \times d}$ ($|C|$ is the number of unique POIs in the dataset, d is the dimensionality in the lower space) by maximizing the probabilities of locations (check-ins) given their context in trajectories.

We model the occurrence probability of a sequence of m locations/POIs $l_1, \ldots, l_m$ in a particular trajectory using all previous locations (rather than a window of w previous locations), as follows:

$$P(l_1, \ldots, l_m) = \prod_{i=1}^{m} P(l_i | l_1, \ldots, l_{i-1}). \tag{1}$$

In essence, this is an autoregressive type of a problem—and a rather natural approach is to model it via RNNs. Specifically, RNNs consist of layers of neuron units, and they rely on the respective cells and controlled gates to capture (and correspondingly update) the user's dynamic preferences as given in the sequence of user's check-ins. Thus, at a time-step t, the output of the hidden state $\mathbf{h}_{t-1}$ of RNNs from the previous step, along with the POI vector $\mathbf{v}_t$, are used to compute the current state $\mathbf{h}_t$, for which we rely on the following formula:

$$\mathbf{h}_t = \sigma(\mathbf{W}^{hh}\mathbf{h}_{t-1} + \mathbf{W}^{ih}\mathbf{v}_t), \tag{2}$$

where matrix $\mathbf{W}^{hh}$ conditions the output $\mathbf{h}_{t-1}$ of a non-linear activation function $\sigma(\cdot)$ at previous time-step $t-1$ (e.g., ReLU and Sigmoid), and matrix $\mathbf{W}^{ih}$ parameterizes the connection from input vector $\mathbf{v}_t$ to the hidden layer.

Given the current state $\mathbf{h}_t$ and the input POI vector $\mathbf{v}_t$, the next POI is predicted based on the probability distribution at time t,

$$\hat{\mathbf{y}}_t = \text{softmax}(\mathbf{W}^{ho}\mathbf{h}_t), \tag{3}$$

where matrix $\mathbf{W}^{ho}$ parameterizes the hidden-to-output connections, and softmax function produces the normalized probabilities over the output value $\mathbf{o}_t = \mathbf{W}^{ho}\mathbf{h}_t$. The output values $\mathbf{o}_1, \ldots, \mathbf{o}_m$ are therefore a mapping from input vectors $l_1, \ldots, l_m$. Now we can compute the loss of such mapping as the sum of losses over the entire POI sequence and time steps m, similarly to sentence modeling in NLP [25]:

$$\mathcal{L}(\mathrm{y}, \hat{\mathrm{y}}) = -\frac{1}{m}\sum_{t=1}^{m}\sum_{k=1}^{|C|} \mathrm{y}_{t,k} \ln(\hat{\mathrm{y}}_{t,k}) + (1 - \mathrm{y}_{t,k}) \ln(1 - \hat{\mathrm{y}}_{t,k}), \tag{4}$$

where the runtime of training this model with Backward Propagation Through Time (BPTT) is $O(m)$ [73].

3.1 Basic Methodologies

We now introduce the overall framework of our proposed solution, as illustrated in Figure 1. It consists of the following major components: POI embedding for capturing latent semantics of check-ins, trajectory pretraining with parameters initialized to make learning more efficient, encoder (various recurrent neural network models can serve such a role), and decoder to reconstruct the trajectory for trip planning. We then discuss in detail each component of TRED.

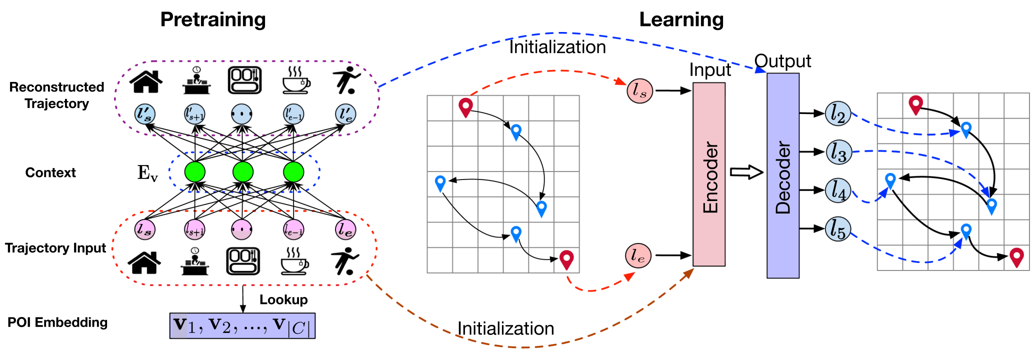


Fig. 1. The overall framework of TRED with training and planning. The training uses an autoencoder to characterize latent semantics of trajectories in an unsupervised manner. The recommendation can be made through the decoder for a given tuple $< l_s, l_e, L >$, where l_s is the start POI and l_e is the end POI, and L is the length.

3.1.1 POI Embedding. To encode latent features of POIs in trajectories, we represent each POI l_i with a low-dimensional vector $\mathbf{v}_i \in \mathbb{R}^d$ instead of using traditional location representation method such as one-hot. Like word embedding in natural language processing [50], we obtain the POI representations $\mathbf{P} \in \mathbb{R}^{|C| \times d}$ (again, $|C|$ denotes the number of POIs in the dataset and d is the dimensionality in the lower space) by maximizing the probabilities of POIs given their context in trajectories.

Essentially, the embedding of a POI l_i is calculated by maximizing the log-likelihood of all trajectories: $\sum_{t=1}^{n} \sum_{i=1}^{m} \log p(l_i | l_{i+j})$, where $-w \le j \le w$, w is the size of sliding window, m is the trajectory length, and n is the number of trajectories. The conditional probability $p(l_i | l_{i+j})$ is defined by the softmax function as

$$p(l_i | l_{i+j}) = \frac{\exp\{\mathbf{v}_i \mathbf{v}'_{i+j}\}}{\sum_{l=1}^{|C|} \exp\{\mathbf{v}_i \mathbf{v}'_l\}}, -w \le j \le w, \tag{5}$$

where $\mathbf{v}_i$ and $\mathbf{v}'_i$ are, respectively, the input and output vector representations of the POI l_i. We now can estimate the probability of a trajectory $T_* = \{l_1, l_2, \ldots, l_m\}$ by

$$p(T_*) = \prod_{i=1}^{m} p(\mathbf{v}_i | \mathcal{N}(l_i)), \tag{6}$$

where $\mathcal{N}(l_i)$ is the context of location l_i in a trajectory T_*, and the value for the probability $p(\mathbf{v}_i | \mathcal{N}(l_i))$ is approximated with

$$p(\mathbf{v}_i | \mathcal{N}(l_i)) = \prod_{l' \in \mathcal{N}(l_i)} p(\mathbf{v}_i | \mathbf{v}_{l'}) = \prod_{l' \in \mathcal{N}(l_i)} \frac{\exp\{\mathbf{v}_i \cdot \mathbf{v}_{l'}\}}{\sum_{l'' \in C} \exp\{\mathbf{v}_{l''} \cdot \mathbf{v}_{l'}\}}. \tag{7}$$

3.1.2 Trajectory Encoding. To capture the context information of POIs, TRED uses RNN—an effective sequence modeling method—as an encoder to code trajectories. The encoder reads the input—a trajectory T_* denoted by a sequence of POI embedding vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ and from which to generate a context vector $\mathbf{E}_{\mathbf{v}_t}$. At each time step t, given the input POI sequence vectors $\mathbf{v}_t$ and previous hidden state $\mathbf{h}_{t-1}$, the encoder computes the hidden state as $\mathbf{h}_t = \text{RNN}(\mathbf{v}_t, \mathbf{h}_{t-1})$. The output of the final hidden layer of the RNN is recognized as the variable-length encoding vector $\mathbf{E}_{\mathbf{v}_t}$.

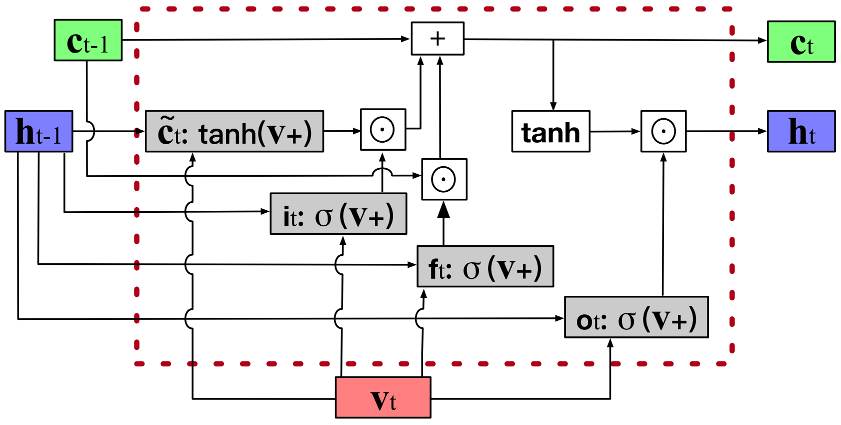


Fig. 2. Illustration of the encoder.

The implementation of RNN encoder can vary—e.g., a *Long Short-Term Memory* (LSTM) [29], *Gated Recurrent Units* (GRU) [12], or bi-directional LSTM/GRU [3]—all of which are able to learn context information with long-range temporal dependencies.

3.1.3 LSTM Encoder. When applying the LSTM as the cell for encoding, the current hidden state $\mathbf{h}_t = \text{LSTM}(\mathbf{v}_t, \mathbf{h}_{t-1})$ is updated by:

$$\begin{aligned}\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{v}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)\\ \mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{v}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)\\ \mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{v}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o),\end{aligned}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, and $\mathbf{b}_*$ are respectively the input gate, forget gate, output gate, and bias vectors; σ is the logistic sigmoid function; matrices $\mathbf{W}$ and $\mathbf{U}$ ($\in \mathbb{R}^{d\times d}$) are the different gate parameters; and $\mathbf{v}_t$ is the embedding vector of the POI l_t. The memory cell $\mathbf{c}_t$ is updated by replacing the existing memory unit with a new cell as:

$$\begin{aligned}\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c\mathbf{v}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)\\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,\end{aligned} \tag{8}$$

where $\tanh(\cdot)$ refers to the hyperbolic tangent function and $\odot$ is the component-wise multiplication. The new memory $\mathbf{c}_t$ is produced by comprehensively considering the forget gate $\mathbf{f}_t$ (and accordingly ignoring the past memory $\mathbf{c}_{t-1}$) and input gate $\mathbf{i}_t$ (and accordingly generating a new memory $\tilde{\mathbf{c}}_t$). Figure 2 shows the computation graph of LSTM encoder.

The encoding vector $\mathbf{E}_{\mathbf{v}_t}$, the output of the final hidden layer, is updated as

$$\mathbf{E}_{\mathbf{v}_t} = \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{9}$$

3.1.4 GRU Encoder. GRU is a simpler LSTM variant that has nonetheless been proven effective. Similarly to LSTM, the hidden state in GRU encoding $\mathbf{h}_t = \text{GRU}(\mathbf{v}_t, \mathbf{h}_{t-1})$ is updated by:

$$\begin{aligned}\mathbf{z}_t &= \sigma(\mathbf{W}_z\mathbf{v}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z)\\ \mathbf{r}_t &= \sigma(\mathbf{W}_r\mathbf{v}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r)\\ \tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_h\mathbf{v}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h),\end{aligned}$$

where reset gate $\mathbf{r}_t$ determines the importance weight of $\mathbf{h}_{t-1}$, and the new memory $\tilde{\mathbf{h}}_t$ is the combination of a new input vector $\mathbf{v}_t$ with the previous hidden state $\mathbf{h}_{t-1}$. Finally, the encoding vector of the last hidden state is generated by considering $\mathbf{h}_{t-1}$ and $\tilde{\mathbf{h}}_t$ as

$$\mathbf{E}_{\mathbf{v}_t} = \mathbf{h}_t = \mathbf{z}_t\tilde{\mathbf{h}}_t + (1 - \mathbf{z}_t)\mathbf{h}_{t-1}. \tag{10}$$

Compared to the LSTM, GRU encoder performs less parameterized affine transform and thus is more memory and computation efficient.

3.1.5 Bi-directional Encoder. In this article, we choose the bi-directional encoder where the trajectories are fed into two RNN layers (LSTM or GRU) in different directions [26]. The hidden states are concatenated to get the final context vector.

Bi-directional RNNs have recently been used for representation learning for sequential data and pre-training of natural language models [15, 54]. The main motivation for adopting them in our work is that they enable encoding the context around a pivot-POI (which, in turn, is subsequently used in the attention mechanism).

The forward RNN reads the trajectory (e.g., $l_1, l_2, \ldots, l_m$) and calculates a sequence of forward hidden states $(\overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_m)$. The backward RNN reads the trajectory in a reverse order (e.g., $l_m, l_{m-1}, \ldots, l_1$) and generates a sequence of backward hidden states $(\overleftarrow{\mathbf{h}}_m, \overleftarrow{\mathbf{h}}_{m-1}, \ldots, \overleftarrow{\mathbf{h}}_1)$. Thus, we can obtain an representation for each POI by concatenating the forward and backward hidden states, i.e., $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i]$.

We note that we provide two implementations of the bi-directional TRED—LSTM based and GRU based. They are respectively denoted as TRED-L and TRED-G, and we provide experimental observations regarding their tradeoffs between effectiveness vs. efficiency in Section 6.

3.1.6 Trajectory Decoding. The decoder is also a recurrent neural network trained to output a trajectory $l_1, \ldots, l_L$ by predicting the next POI l_i given the hidden state $\mathbf{h}'_i$ and encoding vector (a.k.a. context vector) $\mathbf{E}_{\mathbf{v}_t}$. Unlike the encoder, the output ith POI l_i and its hidden state $\mathbf{h}'_i$ are conditioned on both previous POI l_{i-1} and hidden state $\mathbf{h}'_{i-1}$:

$$l_i = \mathrm{RNN}(\mathbf{h}'_{i-1}, l_{i-1}, \mathbf{E}_{\mathbf{v}_t}). \tag{11}$$

Thus, the probability of POI sequence $l_1, \ldots, l_L$ with the conditional distribution of the next POI can be computed as:

$$p(l_1, \ldots, l_L) = \prod_{i=1}^{L} p(l_i | l_1, l_2, \ldots, l_{i-1}, \mathbf{E}_{\mathbf{v}_t}) = \prod_{i=1}^{L} \pi(\mathbf{h}'_i, l_{i-1}, \mathbf{E}_{\mathbf{v}_t}), \tag{12}$$

where π is an activation function (e.g., softmax) and the length of recommended POIs L is a user-specified parameter. Note that L may vary from m—the length of the input sequence $\mathbf{v}_1, \ldots, \mathbf{v}_m$—to 1.

3.2 Trajectory Attention

With the above processes of trajectory encoding-decoding, one can readily produce a sequence of POIs with highest probability at each time step. However, we ignore an important characteristic of the POIs in the trajectory recommendation—namely, the importance of POIs is different from each other when generating a particular trip, i.e., some POIs are more influential when compared to others [60]. To overcome this problem, we introduce the idea of trajectory attention—and adaptation of the approach recently used in neural machine translation (NMT) [52]. The basic idea of attention mechanism in sequence-to-sequence NMT model is motivated by the observation that instead of attempting to learn a fixed-length representation for each sentence, one can focus on a specific area of the whole input and refer to these vectors at each decoding step [3]. In a similar spirit, we can reference a particular area of the input rather than the entire trajectory at the decoding step.

In our trip recommendation problem, by introducing attention on the input trajectories, the process of generating planned trip would focus on those POIs that are more influential. When recommending a trajectory, TRED produces a sequence of POIs conditioned on the hidden state $\mathbf{h}'_i$ and the context vector $\mathbf{E}_{\mathbf{v}_t}$. Toward that goal, we add a non-linear function in the decoder RNN to track the state of generating planned POI by computing an attention vector α_i for each input POI. This vector tells the decoder how much it should focus on a particular input POI—the larger the value in α_i, the more influence a POI will play the role of generating the outputs [67].

Since we do not have prior knowledge of which POI in the input trajectory, we leverage the global attention proposed in Reference [49] to involve all the hidden states of the encoder when deriving the context vector as $\mathbf{E}_{\mathbf{v}_t} = \sum_{t=1}^{m} \alpha_{i,t}\mathbf{h}_t$, where $\mathbf{h}_t$ is the tth hidden state of the encoder and m is the length of the input trajectory. Specifically, the attention vector $\alpha_{i,t}$ is computed by comparing the current (decoder) hidden state $\mathbf{h}'_i$ with each source (encoder) hidden state $\mathbf{h}_t$ as

$$\alpha_{i,t} = \frac{\exp(\eta(\mathbf{h}'_i, \mathbf{h}_t))}{\sum_{t=1}^{m} \exp(\eta(\mathbf{h}'_i, \mathbf{h}_t))}, \tag{13}$$

where $\eta(\cdot)$ refers to a score function (or alignment model in Reference [3]) and can be arbitrary function that represents the confidence that the model focuses on a particular input POI encoding vector at the each decoding step. We note that a number of works have proposed various methods for improving the performance of attention, e.g., coverage-based attention [67], monotonic attention [55], conditional random fields-based attention [32], stacked attention layers [69], among others. However, optimal attention mechanism selection is beyond the scope of this work. We use the dot product as score function $\eta(\mathbf{h}'_i, \mathbf{h}_t) = \mathbf{h}_t^{\top}\mathbf{h}'_i$, which is based on the RNN hidden state $\mathbf{h}'_i$ and the POIs of the input trajectory. The alignment model is then fed into the decoder RNN for training with all the other components of the TRED framework.

4 MODEL LEARNING

We now turn the attention to the training process. For each trajectory T_i in $\mathbf{T}^r$, the input is a tuple $\tau_s =< l_1, l_L, L >$ (starting POI, ending POI, and the length of the trajectory), and the predicted output sequence is $\tau_o = l_2, \ldots, l_{L-1}$ (all intermediate POIs). Our objective is to learn a supervised model mapping $\tau_s \mapsto \tau_o$ using the above-described trajectory encoding-decoding method. The training process seeks to optimize parameters $\vartheta*$ for the purpose of encoding $< l_1, l_L, L >$ and decoding it into the target POI sequence $l_2, \ldots, l_{L-1}$. Thus, the objective is to maximize likelihood estimation:

$$\vartheta* = \arg\max_{\vartheta} \prod_{(\tau_s,\tau_o)\in\mathbf{T}^r} P(\tau_o|\tau_s;\vartheta) = \arg\max_{\vartheta} \prod_{(\tau_s,\tau_o)\in\mathbf{T}^r} \prod_{i=4}^{L} P(\tau_o|l_2,\ldots,l_{i-2},\tau_s;\vartheta). \tag{14}$$

4.1 Pre-training

Before training the trajectory recommendation model, we first pre-train an unsupervised model using a sequence autoencoder, which encodes each input trajectory into a vector and then predicts the input trajectory itself at the stage of decoding. This pre-trained model can be used as the initialization for training. Although the objectives of the two models are different, they share a common structure through latent variables. The basic motivation behind this pre-training process is to make our TRED model training converge faster and more stable [14]. In addition, we find this

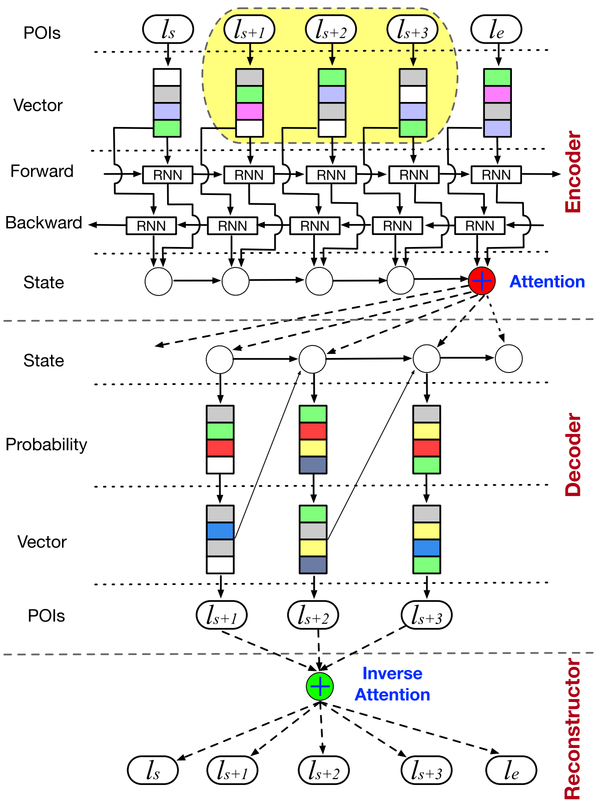


Fig. 3. Illustration of training TRED and pretraining with autoencoder. Note that ignoring the middle POIs (highlighted in yellow ellipse) of input would get the training procedure of TRED. TRED first embeds all the POIs into a low-dimension representation and uses an autoencoder to pretrain the trajectories in unsupervised manner. Then the start-end POI pair (l_s, l_e) combined with tour length L are used to train TRED model to characterize trajectories. A trip can finally be planned for a given tuple $\langle l_s, l_e, L\rangle$.

process can, to some extent, alleviate overfitting usually involved in autoencoder training, which has also been observed in machine translation [56]. The overview of pre-training with autoencoder is illustrated in Figure 3.

The effectiveness of using autoencoder lies in the "memory" function of the neural networks. That is, the pre-trained model may have already "remembered" the visiting order and sequential patterns of training trajectories, which is the possible reason for good and stable performance in initializing sequence-to-sequence-based models [14]. Another merit of this pre-training is that it is an unsupervised manner and thus can be used to leverage large amount of unlabeled data—normally considered to be an efficient way of improving the generalization capability of recurrent networks and thus useful especially when the labeled data are limited.

The objective for this pre-trained encoder is to capture the intrinsic features of moving patterns and underlying structure of trajectories by learning both linear and non-linear feature representations in data. The encoding vector, usually smaller than the original trajectory in terms of dimensionality, maintains meaningful latent attributes of the trajectory, which can be used to reconstruct the outputs when planning the trip. This autoencoder model is proved to be effective and, more importantly, significantly reduce and even eliminate the heavy task of feature engineering.

4.2 Training and Optimizing the Trip Planning

After pre-training the autoencoder model, we use the encoder and decoder to initialize the model for training trip planning. The training detail is similar to the autoencoder except that (1) the input of trip planning consists only the start and end POIs and (2) the output contains the rest POIs in the original trajectories. That is, removing the shadow area highlighted in yellow (encoder) in Figure 3 is the training process of TRED.

Since the training objective is to optimize probability $P(\tau_o|\tau_s;\vartheta)$, decoding of a tuple τ_s, ideally, should be the maximum conditional probability $P(l_2,\ldots,l_{L-1}|\tau_s)$. Nevertheless, one cannot compute the probability of sequence τ_o for every candidate POI and find the maximum one, due to the combinatorial problem of searching all possible $P(l_2,\ldots,l_{L-1})$ POI combinations, especially for the larger value of L and $|C|$. Beam search is a widely used heuristic algorithm in neural machine translation [3, 56] that shrinks search space and significantly reduces the computation complexity by exploiting the sequential-factorization of $P(\tau_o|\tau_s)$ to maintain a small number of partial hypotheses and find an approximation. However, beam search may easily drop to local optimum problem, since a locally optimal choice does not necessarily lead to a maximal conditional probability in a complete planning trip $P(l_2,\ldots,l_{L-1})$.

We note that in TRED, we leverage the *reconstruction* idea from Reference [66] in neural machine translation to alleviate the suboptimal problem of trip inference. The basic idea behind the reconstruction is an added *reconstructor* imposes a constraint that the machine translation model should be able to reconstruct the input source sentence from the target-side hidden layers, which encourages the decoder to embed complete information of the source side [66]. Recall that we pre-train trajectories with an autoencoder, where we use the output of hidden layer at the decoder to reconstruct the original trajectory POI by POI. That is, the reconstructor leverages an inverse context vector $\mathbf{E}_{\mathbf{v}_j}$ to reconstruct the input POI sequence, and the training objective is therefore:

$$\mathcal{J}(\vartheta,\beta) = \arg\max_{\vartheta} \sum_{(\tau_s,\tau_o)\in\mathbf{T}^r} \log P(\tau_o|\tau_s;\vartheta) + \arg\max_{\beta} \sum_{(\tau_s,\tau_o)\in\mathbf{T}} \lambda \log P(\tau_s|\tau_o;\beta), \tag{15}$$

where ϑ and β are model parameters in encoder-decoder and reconstructor, respectively, and the hyper-parameter λ regularizes the performance of encoder-decoder with the reconstruction. Note that the log-likelihood in the first term maximizes the conditional probability in training data $\mathbf{T}^r$ while in the pre-training (the second term) leveraging both training data and testing data $\mathbf{T}^e$. Once the model is trained, we can use the likelihood score $Q_k + \lambda P_k$ associated with the candidate trip τ'_k and its corresponding hidden state in decoder $\mathbf{h}'_k$, where likelihood score Q_k and auxiliary reconstruction score P_k are respectively produced by decoder candidates and reconstructor candidates to conduct the beam search to plan a trip that approximately maximize the score. In testing, reconstruction works as a reranking technique to select a better trip from the k-best candidates generated by the decoder [66]. Algorithm 1 demonstrates the pseudo-code of the TRED training.

We observe that in the proposed model TRED we mainly focus on learning the users' visiting patterns from historical trips. While not explicitly modeling certain constraints (e.g., travel time and distance between successive POIs), TRED can be straightforwardly regularized with such constraints. For example, it is not hard to add a regularizer term to penalize the recommended POIs that are too far away from the previous one. However, despite this, as we will show in the next section, TRED can capture the constraints in a data-driven manner. Thus, for example, TRED performs competitively in terms of POI popularity and trip interest of users, compared to the baselines that have explicitly optimized based on these constraints.

ALGORITHM 1: Overview of the TRED Training.

```
Input: $l_s$: start POI, $l_e$: end POI, $L$: trip length, $\mathbf{T} = \mathbf{T}^r + \mathbf{T}^e$: trajectories, $k$: beam search size.
Embed POIs into vector $\mathbf{v}_i, i \in [1, m]$.
repeat
    /* Pretraining                                                    */
    foreach $T_i \in \mathbf{T}$ do
        Encode $T_i$ to obtain context vector $\mathbf{E}_{\mathbf{v}_t}$;
        Compute attention vector $\alpha_{i,t}$;
        Decode $\mathbf{E}_{\mathbf{v}_t}$ to obtain $\tau_o = l_2, \ldots, l_{L-1}$;
        Produce candidate tuples $< \tau_1, \mathbf{h}_1, P_1 >, \ldots, < \tau_k, \mathbf{h}_k, P_k >$;
        Reconstruct $\tau_o$ to obtain $T_i$;
        Produce candidate tuples $< \tau'_1, \mathbf{h}'_1, P_1 + \lambda Q_1 >, \ldots, < \tau'_k, \mathbf{h}'_k, P_k + \lambda Q_k >$.
    end
    Output: reconstructor parameters $\beta$ and regularizer paramter $\lambda$.
    /* Training                                                       */
    foreach $\tau_s =< l_s, l_e, L >\in \mathbf{T}^r$ do
        Encode $\tau_s$ to obtain context vector $\mathbf{E}_{\mathbf{v}_t}$;
        Compute attention vector $\alpha_{i,t}$;
        Decode $\mathbf{E}_{\mathbf{v}_t}$ to obtain $\tau_o = l_2, \ldots, l_{L-1}$.
    end
    Output: autoencoder parameters $\vartheta$.
    /* Reconstruction                                                 */
    foreach $T_i \in \mathbf{T}^e$ do
        Produce candidate tuples $< \tau_1, \mathbf{h}_1, P_1 >, \ldots, < \tau_k, \mathbf{h}_k, P_k >$;
        Reconstruct $\tau_o$ to obtain $T_i$;
        Produce candidate tuples $< \tau'_1, \mathbf{h}'_1, P_1 + \lambda Q_1 >, \ldots, < \tau'_k, \mathbf{h}'_k, P_k + \lambda Q_k >$.
    end
until converge
```

5 EXPERIMENTS

In this section, we present our experimental observations by comparing TRED with several baseline methods on three public datasets in terms of both effectiveness and scalability of those methods.

5.1 Datasets

The datasets used in our experiments are shown in Table 1. The trajectories in Toronto, Osaka, Glasgow, and Edinburgh are extracted from Flickr photos and videos [65] by Lim et. al. [41], while the Melbourne data are as built in Reference [8]. The Foursquare dataset [78] contains 573,703 check-ins in Tokyo collected for about 10 months (from April 12, 2012, to February 16, 2013). Each check-in is associated with a timestamp, GPS coordinates and some semantics (e.g., fine-grained venue-categories). We obtain different sets of Foursquare@N (N = 100, 200, ...) by randomly selecting N users and their historical trajectories.

We note that the Flickr and Foursquare datasets have relatively fewer POIs/users—however, this is caused by the nature of trip (sequence) planning problem that, having an ordering, is more difficult than a single point recommendation. In other words, one should not only accurately recommend a sequence of POIs but also consider their ranking order. More importantly, these datasets are typically used for evaluating the trip recommendation methods in the state-of-the-art works

Table 1. Descriptives of Datasets (# Denotes the Number of)

Dataset	#POIs	#Trajectories	#Users
Flickr@Edinburgh	29	5,028	1,454
Flickr@Glasgow	29	2,227	601
Flickr@Melbourne	87	5,106	1,000
Flickr@Osaka	29	1,115	450
Flickr@Toronto	30	6,057	1,395
Foursquare@100	30	3,667	100
Foursquare@200	30	6,414	200
Foursquare@400	30	11,254	400
Foursquare@800	30	22,064	800
Geolife	4,796	24,943	179

[8, 18, 39, 41]. To evaluate the performance of models on more general scenarios, we also conduct experiments on a larger and dense GPS dataset.

Geolife dataset [91] was collected by the Microsoft Geolife project over a period of 5+ years (from April 2007 to August 2012). Since trajectories in the original Geolife datasets only have GPS coordinates (longitude and latitude), we cluster all points based on their locations to obtain 4,796 POIs—that is, we save the two digits after the decimal point of longitude and latitude. While the Geolife data may not be the most suitable for trip recommendation, we note that they contain richer human mobility information, especially daily moving patterns, which can be used to evaluate the ability of modeling sequential patterns. For all datasets, we use leave-one-out cross validation to evaluate different trajectory recommendation algorithms, following Reference [8]—i.e., when testing on a trajectory, all other trajectories are used for training.

5.2 Baselines and Metrics

We compare the two variants of TRED, i.e., LSTM-based (TRED-L) and GRU-based encoder-decoder (TRED-G)—both implemented with bi-directional RNNs—with several state-of-the-art approaches. In the following, we present the baselines and define the evaluation metrics.

5.2.1 *Baselines.*

The baselines for trip recommendation consist of the following:

- **Random**: This is a naïve approach that chooses POIs at random (i.e., POI l_i different from l_{i-1} and l_s, l_e, $\forall i$) to construct a trajectory with a desired length.
- **Popularity** [18]: This method is essentially recommending the most popular and unvisited POI at each time.
- **PersTour and PersTour-L** [41]: The tour recommendation problem is modeled using a formulation of the orienteering problem and considers user trip constraints such as time limits and the need to start and end at specific POIs. PersTour explores POI features with a time budget, while PersTour-L is a variant by replacing the time budget with a constrained trajectory length.
- **POIRank** [8]: POIRank recommends a trajectory by first ranking POIs with rankSVM and then connecting them according to ranking scores.
- **Markov and Markov-Rank** [8]: Markov-based method considers the POI-to-POI transition probabilities and recommends a trajectory by maximizing the transition likelihood. Markov-Rank is a method for learning both POI ranking and Markov transition. The ranking of POIs is learned by rankSVM with linear kernel and L_2 loss. Trajectories recommended by Markov and Markov-Rank are trained using the maximum likelihood approach.

- **Path and Path-Rank** [8]: Path and Path-Rank are methods for eliminating sub-tours in Markov and Markov-Rank by finding the best path using an Integer Linear Program with sub-tour elimination constraints adapted from the Traveling Salesman Problem.
- **PersQueue** [39]: This most recent personalized tour recommendation approach recommends personalized itineraries that aims to maximize attraction popularity and user interest preferences and minimize queuing times, while adhering to a time constraint for completing the itinerary. It adopts Monte Carlo Tree Search (MCTS) for the queuing time aware tour recommendation, where the reward reflects the POI popularity, user interest and queuing time associated with each itinerary. Due to lack of explicitly queuing time of POIs, we use the time interval between two adjacent POIs as the implicit queuing time for each POI.

5.2.2 *Evaluation Metrics.* We use the standard F_1 and pairs-F_1 scores to evaluate the trip recommendation performance.

$\mathbf{F_1}$ score. Following Reference [41], we use the F_1 score of a recommended trajectory as one evaluation metric. It is defined as the harmonic mean of Precision and Recall of POIs in a trajectory:

$$F_1 = \frac{2 \times P \times R}{P + R}, \tag{16}$$

where P and R are respectively the trajectory Precision and Recall. Precision determines the proportion of POIs in a user's real-life trajectory that were also recommended in the planned trip, while Recall is the proportion of POIs planed in the results that were also in a user's ground-truth trajectory.

pairs-$\mathbf{F_1}$ score. pairs-F_1 is a new metric proposed in Reference [8], considering both POI correctness and visiting order by measuring the F_1 score of every pair of POIs, whether they are adjacent or not in a trajectory as

$$\text{pairs-}F_1 = \frac{2 \times P' \times R'}{P' + R'}, \tag{17}$$

where P' and R' denote the Precision and Recall of ordered POI pairs, respectively. The value of pairs-F_1 is between 0 and 1. The higher the value, the better the recommended results—value 1 means that both POIs and their visiting order in the planned trajectory are exactly the same as the ground truth.

Popularity score. The overall popularity based on all POIs in a recommended trip τ [41], defined as $\text{Pop}(\tau) = \sum_{l \in \tau} \text{Pop}(l)$, where $\text{Pop}(l)$ measures the number of times POI l has been visited in the data.

Trip Interest. The total interest (for a user u) based on all POIs in a recommended trip τ, defined as $\text{Int}(\tau) = \sum_{l \in \tau} \text{Cat}(l)$, where $\text{Cat}(l)$ is the interest of a user u in POI category.

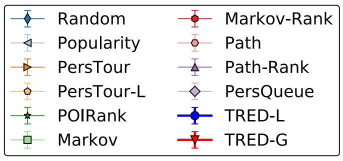
5.2.3 *Experimental Setup.* The experiments are conducted on a machine with 2 Inter Xeon E5 @2.20 GHz CPU, 64G RAM and one Nvidia GTX 1080Ti GPU. We evaluate all algorithms with 5 times 10-fold cross validation. Table 2 shows the optimal parameter settings tuned for both TRED-L and TRED-G, which are implemented using tensorflow and GPU for computational acceleration. All reported results are following this setting.

5.3 Performance on Trip Recommendation

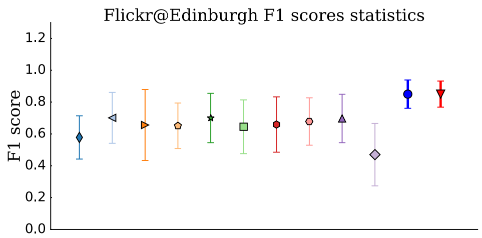
Figures 4 and 5 compare the performance of various trip recommendation algorithms on Flickr data in terms of F_1 and pairs-F_1 scores respectively. The results demonstrate that TRED outperforms all previous algorithms, both heuristics and feature-based learning methods, in almost all scenarios—except that PTour achieves slightly higher pairs-F_1 score on Glasgow data than TRED. In fact, the performance of TRED can be easily tuned better by adding more layers—however, we prefer to

Table 2. Optimal Parameter Settings Tuned for TRED-L and TRED-G

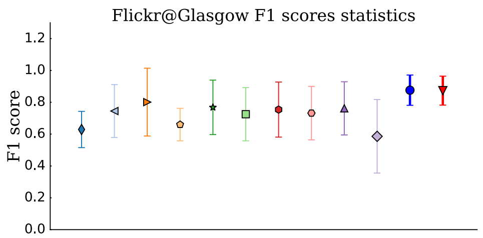
Parameters	Values
Number of layers	2
Number of hidden units	29
Dropout rate	0.5
Learning rate	0.5
Learning rate decay factor	0.99
Batch size	64



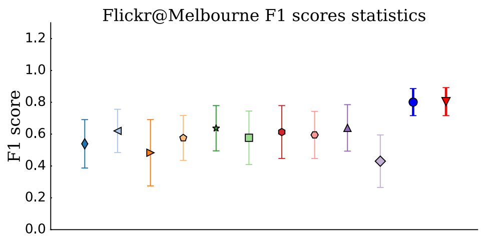
(a) Legend.



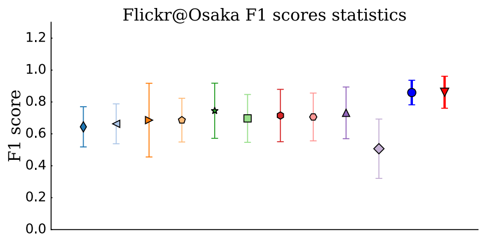
(b) F_1 on Edinburgh.



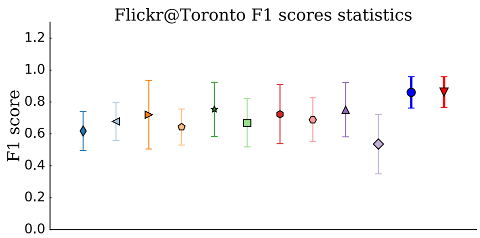
(c) F_1 on Glasgow.



(d) F_1 on Melbourne.



(e) F_1 on Osaka.

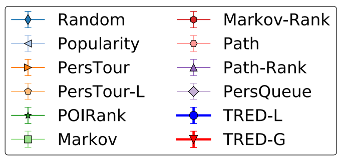


(f) F_1 on Toronto.

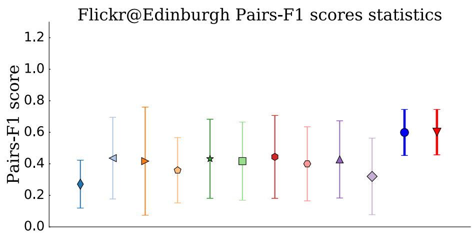
Fig. 4. F_1 Comparisons among different algorithms on Flickr data. The table with the exact values used this figure is reported in the Appendix.

keep all the reported results using the same experimental settings. Another reason behind this choice is that we are interested in investigating the ability of deep learning–based methods on the trip planning problem while minimizing the complexity of neural networks models.

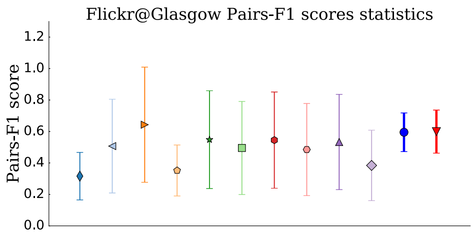
Tables 3 and 4, respectively, summarize the F_1 and pairs-F_1 performance comparison between TRED and several baselines on Geolife and Foursquare datasets where the best method is shown in bold, and the second best is shown as underlined. Note that the values before and after "±" are respectively the mean and standard deviation of POIs successfully predicted (or reconstructed) by



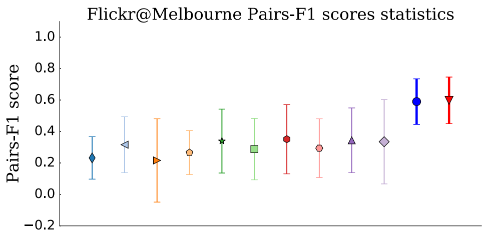
(a) Legend.



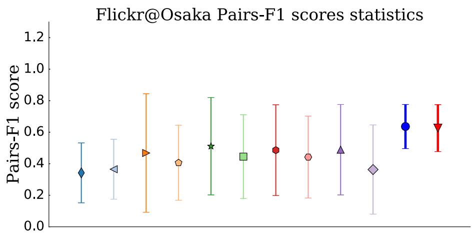
(b) Pairs-F_1 on Edinburgh.



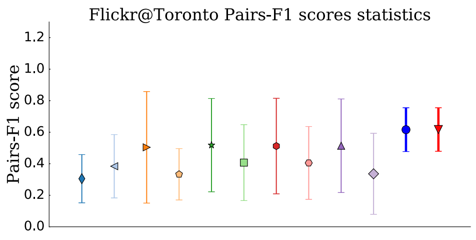
(c) pairs-F_1 on Glasgow.



(d) Pairs-F_1 on Melbourne.



(e) Pairs-pairs-F_1 on Osaka.



(f) pairs-F_1 on Toronto.

Fig. 5. Pairs-F_1 comparisons among different algorithms on Flickr data. The table with the exact values used this figure is reported in the Appendix.

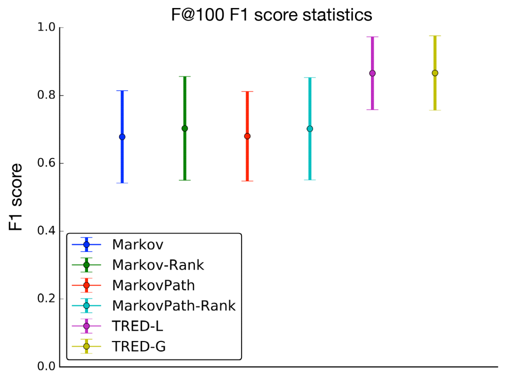
Table 3. F_1 Comparisons among Different Algorithms on Geolife and Foursquare Data

	Geolife	**F@100**	**F@200**	**F@400**	**F@800**
Random	0.116 ± 0.121	0.622 ± 0.103	0.630 ± 0.105	0.626 ± 0.103	0.617 ± 0.109
Popularity	0.176 ± 0.124	0.664 ± 0.120	0.687 ± 0.133	0.682 ± 0.126	0.688 ± 0.143
POIRank	0.231 ± 0.132	0.710 ± 0.155	0.748 ± 0.172	0.743 ± 0.166	0.716 ± 0.159
TRED-L	**0.459 ± 0.248**	0.866 ± 0.107	0.871 ± 0.109	**0.874 ± 0.108**	0.883 ± 0.114
TRED-G	0.431 ± 0.233	**0.866 ± 0.110**	**0.873 ± 0.106**	0.873 ± 0.108	**0.887 ± 0.110**

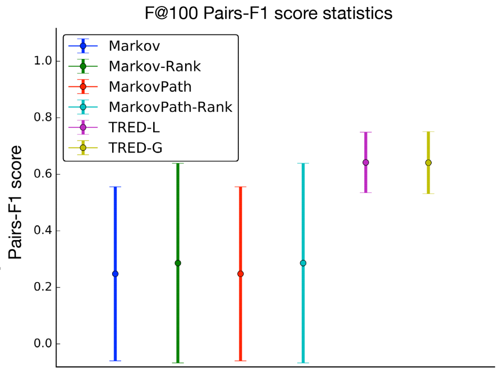
all methods. The reason of only focusing on these methods is that the number of POIs and trajectories in these two datasets are too large to train the Markov-based POI-POI transition model and the feature-based user preference models used in previous works [8, 18], as well as the MCTS-based queuing time aware model [39]—usually they require days and even weeks to train models in our experiment settings. Therefore, we only compare the performance of TRED with Markov-based methods on Foursquare@100 dataset. As illustrated in Figure 6, TRED again exhibits significant improvement on trip recommendation in both metrics.

Table 4. Pairs-F_1 Comparisons among Different Algorithms on Geolife and Foursquare Data

	Geolife	**F@100**	**F@200**	**F@400**	**F@800**
Random	0.011 ± 0.033	0.258 ± 0.153	0.275 ± 0.164	0.270 ± 0.156	0.259 ± 0.156
Popularity	0.000 ± 0.000	0.192 ± 0.269	0.255 ± 0.313	0.239 ± 0.297	0.259 ± 0.323
POIRank	0.000 ± 0.000	0.251 ± 0.351	0.353 ± 0.410	0.330 ± 0.395	0.285 ± 0.354
TRED-L	**0.204 ± 0.229**	**0.642 ± 0.166**	0.658 ± 0.179	0.646 ± 0.168	**0.653 ± 0.171**
TRED-G	0.199 ± 0.158	0.641 ± 0.166	**0.663 ± 0.176**	**0.648 ± 0.167**	0.651 ± 0.167



(a) F_1 on Foursquare@100.



(b) pairs-F_1 on Foursquare@100.

Fig. 6. Performance on Foursquare data with previous 100 users' trajectories (F@100).

From the experimental results we observe that:

- Learning-based methods are more accurate than heuristic methods on trip recommendation.
- The performance of Markov chains-based POI-POI transition models are restricted by the number of POIs, i.e., the more POIs, the less accuracy they achieve. For example, the performance of methods proposed in References [8, 18] deteriorates as the number of POIs increase (e.g., they all behave worse in the case of Melbourne data that has more POIs). However, the number of trajectories may also affect the performance of the Markov-based models. For instance, when the number of POIs are equal (e.g., Osaka, Glasgow, Toronto and Edinburgh data), more trajectories and lower accuracy are achieved by these methods.
- In contrast, both of our methods (TRED-L and TRED-G) act in a rather stable manner as the number of POIs and trajectories increases. An extreme case is the Geolife data that has particularly larger number of POIs, where the performance of TRED drops to lower value. However, all the other trip recommendation methods fail to work especially in terms of the pairs-F_1 score.
- Surprisingly, while MCTS is successful in many reinforcement learning tasks such as Alpha Go, PersQueue [39] fails in our implementation. The main reason is that PersQueue is very sensitive to data—while performing well on datasets that explicitly contain the queuing time associated with attractions (as reported in Reference [39]), it fails to most datasets where the queuing time is only implicitly given.
- As our motivation for this work, TRED outperforms previous works in terms of learning the sequential information of trajectories—especially, the performance of TRED on

Table 5. Popularity Comparisons on Flickr Data

	Edinburgh	**Glasgow**	**Osaka**	**Toronto**
Random	0.656 ± 0.025	0.483 ± 0.048	0.433 ± 0.055	0.581 ± 0.032
Popularity	1.775 ± 0.039	1.399 ± 0.075	0.837 ± 0.062	1.566 ± 0.050
PersTour	2.012 ± 0.043	1.601 ± 0.089	1.144 ± 0.093	1.960 ± 0.064
PersTour-L	**2.016 ± 0.042**	1.562 ± 0.089	1.126 ± 0.095	**2.053 ± 0.063**
TRED-L	2.014 ± 0.040	**1.665 ± 0.092**	**1.168 ± 0.094**	2.022 ± 0.060
TRED-G	2.009 ± 0.043	1.582 ± 0.084	1.148 ± 0.098	2.046 ± 0.066

Table 6. User Interest Comparisons on Flickr Data

	Edinburgh	**Glasgow**	**Osaka**	**Toronto**
Random	0.526 ± 0.033	0.229 ± 0.041	0.305 ± 0.089	0.467 ± 0.037
Popularity	0.577 ± 0.033	0.217 ± 0.049	0.223 ± 0.066	0.443 ± 0.029
PersTour	**1.579 ± 0.069**	**0.625 ± 0.084**	**1.171 ± 0.206**	**1.223 ± 0.061**
PersTour-L	1.383 ± 0.068	0.563 ± 0.091	1.151 ± 0.213	1.088 ± 0.060
TRED-L	1.402 ± 0.074	0.611 ± 0.084	1.160 ± 0.202	1.085 ± 0.068
TRED-G	1.398 ± 0.072	0.620 ± 0.082	1.166 ± 0.200	1.080 ± 0.065

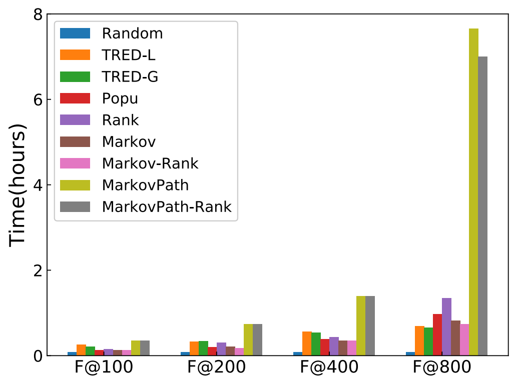
pairs-F_1 prove the effectiveness of RNN-based model on maintaining the order of POIs in trajectories.

- Finally, we compared the trip popularity and user interest (over the POI category) in a recommended trip. We re-iterate that we do not explicitly model the POI popularity and users' interests on the POI category in this work. Table 5 shows the results on POI popularity and, as can be seen, our two models exhibit competitive performance compared to PersTour and PersTour-L—which, however, are completely based on user interest and POI visit frequency. This is a natural result of the pre-training step used in our model that, in fact, encodes the POI frequency in an implicit manner. Combined with the beam search with reconstruction trick enables us to successfully balance the transition probability that is mainly captured by RNN modules and POI popularity when recommending a trip. As for the user interest over the POI category (Table 6), our model performs slightly worse than PersTour and PersTour-L, which incorporate the duration time of POIs—which, arguably, reflects the interest of users over a POI [41]—when recommending a trip.

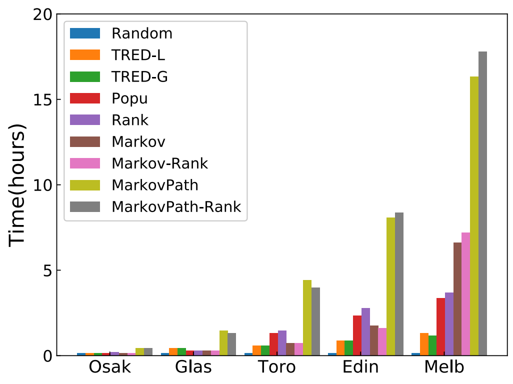
5.4 Efficiency Comparisons

Figure 7(a) and (b) illustrates the computation time taken by the different approaches used in this section and on different datasets. We observe that as the size of data (i.e., number of trajectories and users) increases, the advantage of TRED in terms of efficiency becomes more significant, especially for larger size datasets such as Foursquare@800 and Flickr@Melbourne. In other words, it demonstrates that our method scales much better than the MC-based POI transition methods—which have the overheads of calculating all the pairwise POIs transitions. Note that we do not include the results of PersQueue, because the execution took an extremely long time to converge—i.e., more than a week in our implementation.

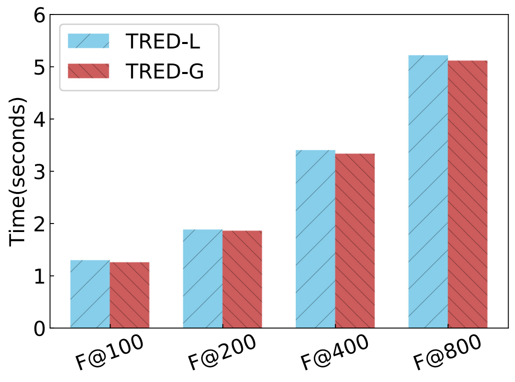
As for recommending trips after training the model (i.e., applicability for real-time recommendations), we observe that a trip can be planed quite efficiently with TRED. Figure 7(c) and (d) shows the testing time of the two proposed models, from where we can observe that the time for planning



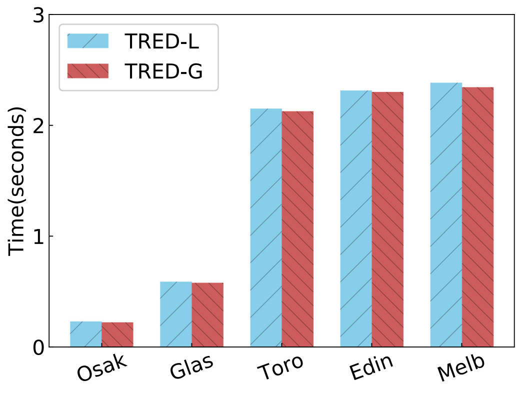
(a) Training time on Foursquare data.



(b) Training time on Flickr data.



(c) Testing time on Foursquare data.



(d) Testing time on Flickr data.

Fig. 7. Comparison of training and testing time.

trips is linearly related to the data size. In addition, TRED-L requires slightly greater execution time than TRED-G on both training and testing, because more parameters need to be learned by LSTM.

6 CONCLUSIONS AND FUTURE WORK

We introduced a novel framework, TRED, for generating an end-to-end trip recommendation, with consideration of intermediate stop-points that capture various POIs. TRED is based on integrating RNNs and sequence-to-sequence learning to exploit existing POI transition patterns. This, in turn, enables us to automatically learn intrinsic patterns among users and POIs that can be subsequently used for planning a trajectory that satisfies both (1) end-locations and (2) sequentiality of intermediate points. One major advantage of TRED is that it does not require labor-intensive feature engineering and thus can be easily generalized to various LBSNs applications. We evaluated the benefits of TRED on several publicly available spatio-temporal datasets. The results show that it outperforms state-of-the-art baselines in recommendation accuracy, while also being efficient in terms of execution time. One of the benefits of a deep learning–based trip recommendation is that it provides valuable insights in correlating contexts that may have influence on users' moving patterns. This, in turn, may be a source of promising guidelines for further investigations of mobility patterns and enable customization for a broader range of applications dealing with modeling trajectory distributions—which is part of the challenges that we plan to address in the future.

The immediate extensions of our work are focusing on jointly exploiting the potential impacts of two complementary contexts: (1) *Temporal*: We are planning to incorporate both trip-time constraints and stay-time constraints in POIs, as well as other temporal kinds of costs. In addition to the existing works (e.g., Reference [39]), we are planning to leverage the body of literature on the time-window variants of Vehicle Routing Problems (VRP), with specific focus on the impact of the travel-time constraints [51, 86]. (2) *Spatial*: In addition to incorporating the impacts of contexts such as uncertainty [31] and semantic similarity of spatial attributes [76], we will also address the problem of pruning the impact of POI locations that are exceeding certain distance (respectively, travel-time) thresholds and cater to variable-length trip recommendations. To this end, we plan to investigate the benefits of the concept of learned trees [33].

A APPENDIX

Exact values used in Figures 4 and 5:

Table 7. F_1 Comparisons among Different Algorithms on Flickr Data

	Edinburgh	**Glasgow**	**Melbourne**	**Osaka**	**Toronto**
Random	0.578 ± 0.136	0.629 ± 0.114	0.539 ± 0.152	0.644 ± 0.126	0.618 ± 0.122
Popularity	0.701 ± 0.160	0.745 ± 0.166	0.620 ± 0.136	0.663 ± 0.125	0.678 ± 0.121
PersTour	0.656 ± 0.223	0.801 ± 0.213	0.483 ± 0.208	0.686 ± 0.231	0.720 ± 0.215
PersTour-L	0.651 ± 0.143	0.660 ± 0.102	0.576 ± 0.141	0.686 ± 0.137	0.643 ± 0.113
POIRank	0.700 ± 0.155	0.768 ± 0.171	0.637 ± 0.142	0.745 ± 0.173	0.754 ± 0.170
Markov	0.645 ± 0.169	0.725 ± 0.167	0.577 ± 0.168	0.697 ± 0.150	0.669 ± 0.151
Markov-Rank	0.659 ± 0.174	0.754 ± 0.173	0.613 ± 0.166	0.715 ± 0.164	0.723 ± 0.185
Path	0.678 ± 0.149	0.732 ± 0.168	0.595 ± 0.148	0.706 ± 0.150	0.688 ± 0.138
Path-Rank	0.697 ± 0.152	0.762 ± 0.167	0.639 ± 0.146	0.732 ± 0.162	0.751 ± 0.170
PersQueue	0.470 ± 0.196	0.586 ± 0.231	0.430 ± 0.165	0.507 ± 0.186	0.536 ± 0.187
TRED-L	0.850 ± 0.089	**0.876 ± 0.095**	0.801 ± 0.085	0.859 ± 0.077	0.860 ± 0.098
TRED-G	**0.850 ± 0.082**	0.873 ± 0.091	**0.804 ± 0.088**	**0.861 ± 0.100**	**0.862 ± 0.096**

Table 8. Pairs- F_1 Comparisons among Different Algorithms on Flickr Data

	Edinburgh	**Glasgow**	**Melbourne**	**Osaka**	**Toronto**
Random	0.271 ± 0.152	0.316 ± 0.151	0.232 ± 0.135	0.342 ± 0.190	0.305 ± 0.153
Popularity	0.436 ± 0.259	0.507 ± 0.298	0.316 ± 0.178	0.365 ± 0.190	0.384 ± 0.201
PersTour	0.417 ± 0.343	**0.643 ± 0.366**	0.216 ± 0.265	0.468 ± 0.376	0.504 ± 0.354
PersTour-L	0.359 ± 0.207	0.352 ± 0.162	0.266 ± 0.140	0.406 ± 0.238	0.333 ± 0.163
POIRank	0.432 ± 0.251	0.548 ± 0.311	0.339 ± 0.203	0.511 ± 0.309	0.518 ± 0.296
Markov	0.417 ± 0.248	0.495 ± 0.296	0.288 ± 0.195	0.445 ± 0.266	0.407 ± 0.241
Markov-Rank	0.444 ± 0.263	0.545 ± 0.306	0.351 ± 0.220	0.486 ± 0.288	0.512 ± 0.303
Path	0.400 ± 0.235	0.485 ± 0.293	0.294 ± 0.187	0.442 ± 0.260	0.405 ± 0.231
Path-Rank	0.428 ± 0.245	0.533 ± 0.303	0.344 ± 0.206	0.489 ± 0.287	0.514 ± 0.297
PersQueue	0.320 ± 0.243	0.384 ± 0.224	0.335 ± 0.268	0.363 ± 0.283	0.336 ± 0.257
TRED-L	0.599 ± 0.146	0.595 ± 0.123	0.590 ± 0.145	**0.636 ± 0.140**	0.616 ± 0.139
TRED-G	**0.601 ± 0.144**	0.599 ± 0.137	**0.598 ± 0.148**	0.626 ± 0.149	**0.617 ± 0.138**

REFERENCES

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Fei Fei Li, and Silvio Savarese. 2016. Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16).*

[2] Frederick Ayala-Gómez, Bálint Daróczy, Michael Mathioudakis, András Benczúr, and Aristides Gionis. 2017. Where could we go?: Recommendations for groups in location-based social networks. In *Proceedings of the 2017 ACM on Web Science Conference (WebSci'17).* 93–102.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR'15).*

[4] Petko Bakalov, Eamonn Keogh, and Vassilis J. Tsotras. 2007. TS2-tree - an efficient similarity based organization for trajectory data. In *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems.* 58:1–58:4.

[5] Igo Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. 2013. Where shall we go today? Planning touristic tours with tripbuilder. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'13).*

[6] Allison J. B. Chaney, David M. Blei, and Tina Eliassi-Rad. 2015. A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'15).* 43–50.

[7] Chao Chen, Daqing Zhang, Bin Guo, Xiaojuan Ma, Gang Pan, and Zhaohui Wu. 2015. TripPlanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints. *IEEE Trans. Intell. Transport. Syst.* 16, 3 (2015), 1259–1273.

[8] Dawei Chen, Cheng Soon Ong, and Lexing Xie. 2016. Learning points and routes to recommend trajectories. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'16).*

[9] Xuefeng Chen, Yifeng Zeng, Gao Cong, Shengchao Qin, Yanping Xiang, and Yuanshun Dai. 2015. On information coverage for location category based point-of-interest recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[10] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. 2012. Fused matrix factorization with geographical and social influence in location-based social networks. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[11] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'13).*

[12] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.35 55* (2014).

[13] Chaoran Cui, Jialie Shen, Liqiang Nie, Richang Hong, and Jun Ma. 2017. Augmented collaborative filtering for sparseness reduction in personalized POI recommendation. *ACM Trans. Intell. Syst. Technol.* 8, 5 (2017), 1–23.

[14] Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proceedings of the Neural Information Processing Systems (NIPS'15).*

[15] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova arXiv. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'19).*

[16] Cédric du Mouza, Philippe Rigaux, and Michel Scholl. 2007. Parameterized pattern queries. *Data Knowl. Eng.* 63, 2 (2007).

[17] Jochen Eisner and Stefan Funke. 2012. Sequenced route queries: Getting things done on the way back home. In *Proceedings of the ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL'17).*

[18] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized ranking metric embedding for next new poi recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'15).*

[19] Krempl G., Siddiqui Z. F., and Spiliopoulou M. 2011. Online clustering of high-dimensional trajectories under concept drift. In *Proceedings of the The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD'11).*

[20] Huiji Gao, Jiliang Tang, and Huan Liu. 2012. gSCorr: Modeling geo-social correlations for new check-ins on location-based social networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'12).*

[21] Qiang Gao, Fan Zhou, Kunpeng Zhang, and Goce Trajcevski. 2017. Identifying human mobility via trajectory embeddings. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'17).*

[22] Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Grammati Pantziou, and Nikolaos Vathis. 2017. Scenic route planning for tourists. *Pers. Ubiq. Comput.* 21, 1 (2017), 137–155.

[23] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. 2014. A survey on algorithmic approaches for solving tourist trip design problems. *J. Heurist.* 20, 3 (2014), 291–328.

[24] Matthew S. Gerber. 2014. Predicting crime using Twitter and kernel density estimation. *Decis. Support Syst.* 61, 1 (2014), 115–125.

[25] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep Learning*, Vol. 1. MIT Press, Cambridge, MA.

[26] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

[27] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. *Eur. J. Operat. Res.* 255, 2 (2016), 315–332.

[28] Jing He, Xin Li, and Lejian Liao. 2017. Category-aware next point-of-interest recommendation via listwise bayesian personalized ranking. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'17).*

[29] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neur. Comput.* 9, 8 (1997), 1735–1780.

[30] Hsun Ping Hsieh and Cheng Te Li. 2014. Mining and planning time-aware routes from check-in data. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'14).*

[31] Jiafeng Hu, Reynold Cheng, Zhipeng Huang, Yixang Fang, and Siqiang Luo. 2017. On embedding uncertain graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management.* ACM, 157–166.

[32] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR'17).*

[33] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD'18).* 489–504.

[34] Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. 2010. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'10).*

[35] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[36] Huayu Li, Yong Ge, Defu Lian, and Hao Liu. 2017. Learning user's intrinsic and extrinsic interests for point-of-interest recommendation: A unified approach. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'17).*

[37] Xutao Li, Gao Cong, Xiao Li Li, Tuan Anh Nguyen Pham, and Shonali Krishnaswamy. 2015. Rank-GeoFM: A ranking based geographical factorization method for point of interest recommendation. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR'15).*

[38] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 2014. GeoMF: Joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (SIGKDD'14).*

[39] Kwan Hui Lim, Jeffrey Chan, Shanika Karunasekera, and Christopher Leckie. 2017. Personalized itinerary recommendation with queuing time awareness. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR'17).*

[40] Kwan Hui Lim, Jeffrey Chan, Shanika Karunasekera, and Christopher Leckie. 2018. Tour recommendation and trip planning using location-based social media: A survey. *Knowl. Inf. Syst.* 60, 3 (2018), 1–29.

[41] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2015. Personalized tour recommendation based on user interests and points of interest visit durations. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'15).*

[42] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (SIGKDD'13).*

[43] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[44] Yiding Liu, Tuan-Anh Pham, Gao Cong, and Quan Yuan. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment* 10, 10 (2017), 1010–1021.

[45] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'14).*

[46] Corrado Loglisci, Donato Malerba, and Apostolos N. Papadopoulos. 2014. Mining trajectory data for discovering communities of moving objects. In *Proceedings of the Workshops of the International Conference on Extending Database and the International Conference on Database Theory (EDBT/ICDT'14).* 301–308.

[47] Hsueh Chan Lu, Ching Yu Chen, and Vincent S. Tseng. 2012. Personalized trip recommendation with multiple constraints by mining user check-in behaviors. In *Proceedings of the ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL'12).*

[48] Xin Lu, Changhu Wang, Jiang Ming Yang, Yanwei Pang, and Lei Zhang. 2010. Photo2Trip: Generating travel routes from geo-tagged photos for trip planning. In *Proceedings of the ACM Multimedia Conference (MM'10)*.

[49] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*.

[50] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[51] Liu Na, Shao Xueyan, and Qi Mingliang. 2012. A Bi-objective evacuation routing engineering model with secondary evacuation expected costs. *Syst. Eng. Proc.* 5 (2012), 1–7.

[52] Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR* (2017).

[53] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady L. Andrienko, Natalia V. Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, José Antônio Fernandes de Macêdo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. 2013. Semantic trajectories modeling and analysis. *ACM Comput. Surv.* 45, 4 (2013), 42:1–42:32.

[54] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[55] Colin Raffel, Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the International Conference on Machine Learning (ICML'17)*.

[56] Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2016. Unsupervised pretraining for sequence to sequence learning. *CoRR* abs/1611.02683 (2016).

[57] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the International World Wide Web Conference (WWW'10)*.

[58] Dimitris Sacharidis, Panagiotis Bouros, and Theodoros Chondrogiannis. 2017. Finding the most preferred path. In *Proceedings of the ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL'17)*. 5:1–5:10.

[59] Mahmoud Attia Sakr and Ralf Hartmut Güting. 2014. Group spatiotemporal pattern queries. *GeoInformatica* 18, 4 (2014), 699–746.

[60] Muhammad Aamir Saleem, Rohit Kumar, Toon Calders, Xike Xie, and Torben Bach Pedersen. 2017. Location influence in location-based social networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. 621–630.

[61] Samiha Samrose, Tanzima Hashem, Sukarna Barua, Mohammed Eunus Ali, Mohammad Hafiz Uddin, and Md. Iftekhar Mahmud. 2015. Efficient computation of group optimal sequenced routes in road networks. In *Proceedings of the IEEE International Conference on Mobile Data Management (MDM'15)*. 122–127.

[62] Yuya Sasaki, Yoshiharu Ishikawa, Yasuhiro Fujiwara, and Makoto Onizuka. 2018. Sequenced route query with semantic hierarchy. In *Proceedings of the Extended Database Technology Conference (EDBT'18)*. 37–48.

[63] Jochen H. Schiller and Agnès Voisard (Eds.). 2004. *Location-Based Services*. Morgan Kaufmann.

[64] Mehdi Sharifzadeh and Cyrus Shahabi. 2008. Processing optimal sequenced route queries using voronoi diagrams. *GeoInformatica* 12, 4 (2008), 411–433.

[65] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2015. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817* (2015).

[66] Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2017).

[67] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *CoRR* (2016).

[68] Fabio Valdés and Ralf Hartmut Güting. 2014. Index-supported pattern matching on symbolic trajectories. In *Proceedings of the ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL'14)*.

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Neural Information Processing Systems (NIPS'17)*.

[70] Pengyang Wang, Jiawei Zhang, Guannan Liu, Yanjie Fu, and Charu Aggarwal. 2018. Ensemble-Spotting: Ranking urban vibrancy via poi embedding with multi-view spatial graphs. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. 351–359.

[71] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (SIGKDD'12)*.

[72] Yu-Ting Wen, Jinyoung Yeo, Wen-Chih Peng, and Seung-Won Hwang. 2017. Efficient keyword-aware representative travel route recommendation. In *Proceedings of the IEEE Transactions on Knowledge and Data Engineering (TKDE'17)*.

[73] D. R. G. H. R. Williams and Geoffrey Hinton. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–538.

[74] Matthew J. Williams, Roger M. Whitaker, and Stuart M. Allen. 2016. There and back again: Detecting regularity in human encounter communities. *IEEE Trans. Mobile Comput.* 16, 6 (2016), 1744–1757.

[75] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. 2017. Modeling trajectories with recurrent neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'17).*

[76] Bo Yan, Krzysztof Janowicz, Gengchen Mai, and Song Gao. 2017. From itdl to place2vec: Reasoning about place type similarity and relatedness by learning embeddings from augmented spatial contexts. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 35.

[77] Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. 2017. Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (SIGKDD'17).*

[78] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. 2015. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *IEEE Trans. Syst. Man Cybernet.: Syst.* 45, 1 (2015), 129–142.

[79] Mao Ye, Xingjie Liu, and Wang-Chien Lee. 2012. Exploring social influence for recommendation: A generative model approach. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR'12).*

[80] Mao Ye, Peifeng Yin, Wang Chien Lee, and Dik Lun Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR'11).*

[81] Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia Sadiq. 2016. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. *ACM Trans. Intell. Syst. Technol.* 35, 2 (2016), 1–44.

[82] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-aware point-of-interest recommendation. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR'13).*

[83] Chenyi Zhang, Hongwei Liang, Ke Wang 0001, and Jianling Sun. 2015. Personalized trip recommendation with POI availability and uncertain traveling time. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'15).*

[84] Chenyi Zhang, Hongwei Liang, and Ke Wang. 2016. Trip recommendation meets real-world constraints: POI availability, diversity, and traveling time uncertainty. *ACM Trans. Intell. Syst. Technol.* 35, 1 (2016), 1–28.

[85] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[86] Y. Zhang and X. D. Chen. 2014. An optimization model for the vehicle routing problem in multi-product frozen food delivery. *J. Appl. Res. Technol.* 12, 2 (2014), 239–250.

[87] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[88] Zhenhua Zhang, Qing He, Hanghang Tong, Jizhan Gou, and Xiaoling Li. 2016. Spatial-temporal traffic flow pattern identification and anomaly detection with dictionary-based compression theory in a large-scale urban network. *Transport. Res. Part C: Emerg. Technol.* 71 (2016), 284–302.

[89] Shenglin Zhao, Tong Zhao, Irwin King, and Michael R. Lyu. 2017. Geo-Teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In *Proceedings of the International World Wide Web Conference (WWW'17).*

[90] Weimin Zheng, Zhixue Liao, and Jing Qin. 2017. Using a four-step heuristic algorithm to design personalized day tour route within a tourist attraction. *Tour. Manage.* 62 (2017), 335–349.

[91] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the International World Wide Web Conference (WWW'09).*

[92] Fan Zhou, Qiang Gao, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2018. Trajectory-user linking via variational AutoEncoder. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'18).* 3212–3218.

Received December 2018; revised December 2019; accepted December 2019