Spatiotemporal Representation Learning for Driving Behavior Analysis: A Joint Perspective of Peer and Temporal Dependencies

Pengyang Wang, Xiaolin Li, Yu Zheng, Charu Aggarwal, Yanjie Fu

Abstract—Driving is a complex activity that requires multi-level skilled operations (e.g., acceleration, braking, turning). Analyzing driving behaviors can help us assess driver performances, improve traffic safety, and, ultimately, promote the development of intelligent and resilient transportation systems. While some efforts have been made for analyzing driving behaviors, existing methods can be improved via representation learning by jointly exploring the peer and temporal dependencies of driving behaviors. To that end, in this paper, we develop a **P**eer and **T**emporal-**A**ware **R**epresentation **L**earning based framework (PTARL) for driving behavior analysis with GPS trajectory data. Specifically, we first detect the driving operations and states of each driver from their GPS traces. Then, we derive a sequence of multi-view driving state transition graphs from the driving state sequences, in order to characterize a driver's driving behaviors that vary over time. In addition, we develop a peer and temporal-aware representation learning method to learn a sequence of time-varying yet relational vectorized representations from the driving state transition graphs. The proposed method can simultaneously model both the graph-graph peer dependency and the current-past temporal dependency in a unified optimization framework. Also, we provide two effective solutions for the optimization problem: (i)a joint optimization solution of representation learning and prediction; and (ii) a step-by-step solution of representation learning and prediction. Besides, we explore two strategies to fuse the learned representations from multi-view transition graphs: (i) simple alignment and (ii) collective fusion. Moreover, we apply the developed framework to the two applications of quantitative transportation safety: (i) scoring of driving performances, and (ii)detection of dangerous regions. Finally, we present extensive experimental results with big trajectory data to demonstrate the enhanced performances of the proposed method for quantitative transportation safety.

Index Terms—Representation Learning, Peer Dependency, Temporal Dependency. Driving Behavior Analysis.

1 Introduction

Driving behavior is a complex activity that requires multi-level skilled operations, such as acceleration, deceleration, keeping constant speed, turning left, turning right, and moving straight. Analyzing driving behavior can help us assess driver performances, enhance traffic safety, and, ultimately, promote the development of intelligent and resilient transportation systems, in order to enable many important applications, such as monitoring drivers, vehicles, and roads, providing early warning and driving assistance, and enhancing driving comfort and energy saving. In this paper, we study the problem of learning to represent driving behaviors with applications to quantitative transportation safety.

Prior studies in driving behaviors analysis can be categorized into: (i) descriptive analysis, in which transportation experts define measurements (e.g., harsh or frequent acceleration/braking, sharp turn, acceleration before turn) based on transportation theory to describe driving behaviors [4]; (ii) predictive analysis, in which researchers mine the patterns from driving data and apply machine learning models (e.g., SVM, naive Bayesian, etc.) to predict risky scores [36]; (iii) causal analysis, in which researchers identify the causal factors of driving behaviors and explain how these factors influence road safety [20]. Moreover, there are studies utilizing CAN data to quantify driving behaviors [13], [15]. However, previous studies have some limits. For example, some studies are based on biased and expensive data sources, e.g., self-reported survey. Some studies empirically define descriptive variables with the help of domain experts, and, thereby, are lack of generalization capability when dealing with big and noisy driving data. Some studies mainly focus on analyzing coarse-grained (e.g., user-group-level or region-level) driving behaviors, rather than individual-level driving behaviors. Besides, CAN data may cause privacy issues which can be avoided by analyzing the ubiquitous GPS data. CAN data is more accurate to quantify driving operations, like speed and directions. It can record the vehicle status and can be read through specific facilities. However, since it needs equipments to read the data, the accessibility to CAN data may cause privacy issue that leads to the difficulty of collecting data. For example, many drivers are not willing to release CAN data to the insurance company. Unlike CAN data, due to the pervasiveness of GPS sensors (e.g., mobile phones), GPS data can be easily obtained from location-based apps (e.g., google maps, yelp) that are granted permission for collecting data by users themselves. On the other hand, there are many public GPS dataset on

- *Yanjie Fu and Pengyang Wang are with University of Central Florida. Email: {yanjiefoo, merlinwang555}@gmail.com.*
- *Xiaolin Li is with Nanjing University. Emails: lixl@nju.edu.cn.*
- *Yu Zheng is with Urban Computing Business Unit, JD Finance. Emails: msyuzheng@outlook.com.*
- *Charu Aggarwal is with IBM T. J. Watson Research Center. Emails: charu@us.ibm.com.*
- *Xiaolin Li and Yanjie Fu are co-contact authors.*

recording driver behaviors. All the personal information has been encrypted that there will be any privacy issues. In a nutshell, due to the easier availability and well protected personal information, we propose to choose GPS data for the task of driving behavior analysis.

Indeed, the increasingly pervasiveness of GPS sensors has accumulated large-scale driving behavior data. And the emergence of representation learning techniques provides great potential for automated behavior profiling. It is naturally promising to combine high-resolution widely-available GPS trajectories and representation learning for driving behavior analysis. However, three unique challenges arise in achieving this goal.

The first challenge is that GPS traces (e.g., time, latitude, longitude) encode the driving operations, states, and styles in a semantically-implicit way, which jeopardizes the applicability of representation learning. Therefore, it highly necessitates a novel method to transform GPS traces into an appropriate structure that can effectively characterize driving activities and corresponding spatio-temporal dynamics. To address the challenge, we analogize driving behaviors as a sequence of state transition graphs, and develop a three-step characterization method. To begin with, we identify two types of driving operations: (i) speed-related (i.e., acceleration, deceleration, constant speed) and (ii) direction-related (i.e., turning left, turning right, move straight) from a GPS trajectory, and obtain a sequence of driving operations for each driver. Later, we define a driving state as a two-tuple combination that includes a speed operation status and a direction operation status, and extract a sequence of driving states for each driver. Lastly, to reduce the possible impacts of outliers, which might be generated by small sensor data errors, we derive multi-view driving state transition graphs (i.e., the transition probability and transition duration of driving states) to characterize driving behaviors, and obtain a sequence of driving state transition graphs as the inputs of representation learning.

Second, after analyzing large-scale driving data, we identify two dependencies of driving state transition graphs: (i) *peer dependency*: if two driving state transition graphs are structurally similar, then the embeddings of the two graphs are similar in the latent feature space; (ii) *temporal dependency*: the embedding of a driving state transition graph not just depends on the driving operations at the current time period, but also has correlation with the previous ones. It thus is important to model the coupling of both the peer and temporal dependencies in representation learning. Therefore, we develop a **P**eer and **T**emporal-**A**ware **R**epresentation **L**earning framework (PTARL) that can jointly model the graph-graph peer dependency across drivers, as well as the current-past temporal dependency within a driver, in representation learning. The proposed method can learn a sequence of time-varying yet relational vectorized representations from the driving state transition graphs, using a widely-available GPS data source and with very limited knowledge of surrounding conditions.

Third, it is challenging to choose an appropriate optimization strategy. From the perspective of representation learning, the objective is to obtain the optimal representations of driving behavior; from the perspective of driving performance assessment , the objective is to predict driving scores as accurate as possible. Indeed, there are two optimization strategies: (i) a joint optimization strategy, in which we jointly minimize the total loss of both the presentation learning task and the regression task; (ii) a step-by-step strategy, in which we first minimize the loss of the representation learning task, and then minimize the loss of the regression task. A a result, based on the two strategies, we develop two variants of PTARL.

Along these lines, in this paper, we develop a peer and temporal-aware representation learning based analytic framework for driving behaviors analysis using GPS traces. Specifically, we first construct a sequence of multi-view driving state transition graphs from GPS traces to characterize the dynamic driving behaviors of each driver. Besides, we identify the graph-graph peer and current-past temporal dependencies of driving behaviors, and incorporate the modeling of the peer and temporal dependencies into a unified Auto-Encoder based optimization framework. Also, we develop two different strategies for the optimization problem: (i) a joint optimization strategy and (ii) a step-by-step strategy. As applications, we exploit the learned representations of driving behaviors for driving performance assessment and risky region detection. Finally, we present extensive experiments to demonstrate the enhanced performances of the proposed method with real-world vehicle GPS traces.

2 Problem Statement and Framework Overview

TABLE 1: Summary of notations.

Symbol	Discription
ϕ_t	The latitude at time t
λ_t	The longitude at time t
G_i^τ	The driving behavior transition graph sequence of driver i at the time slot τ
$\mathbf{x}_i^\tau$	The original original vector representation of G_i^τ
$\mathbf{z}_i^\tau$	The learned representation for the driver i at the time slot τ
$(\mathbf{y}_i^o)^\tau$	The latent feature representations of the driver i at hidden layers o at the time τ in the encode process
$(\hat{\mathbf{y}_i^o})^\tau$	The latent feature representations of the driver i at hidden layers o at the time τ in the decode process
$\mathbf{W}$ $\mathbf{b}$	Weights and biases in the encode process
$\hat{\mathbf{W}}$ $\hat{\mathbf{b}}$	Weights and biases in the decode process process
$\mathcal{H}(\star)$	Loss function
A, B	The hyperparameters to control the weight of the representation learning loss and regression loss

We first introduce some important definitions and the problem statement, and then present an overview of the proposed framework.

2.1 Definitions and Problem Statement

Definition 1. Driving Operation. *Driving operations are defined as a set of activities and steps that a driver operates when driving a vehicle, according to the driver's personal judgment, experience and skills. Since a moving object can be characterized by speed and direction, we similarly categorize driving operations into (i) speed-related operations (i.e., acceleration, deceleration, constant speed) and (ii) direction-related operations (i.e., turning left, turning right, moving straight). The speed-related operations show how a driver operates the clutch pedal, gas pedal, and brake pedal of a vehicle. The direction-related operations show how a driver operates the steering wheel of a vehicle. The driving operations can be detected from GPS traces.*

Definition 2. Driving State. *A driving state concerns the way that a vehicle moves at a specific time point or in a small time window. In other words, a driving state of a vehicle contains both the speed status (i.e., acceleration, deceleration, constant speed) and the direction status (i.e., turning left, turning right, moving straight) of a vehicle. For instance, a driving state example of a car can be <constant speed, moving straight>.*

Definition 3. Driving State Transition Graph. *The driving states of a vehicle usually changes over time. For instance, a sequence of driving states for a vehicle can be: [<acceleration, moving straight>, <constant speed, moving straight>, $\cdots$, <deceleration, turning right>]. We propose to develop a driving state transition graph to summarize and characterize such time-varying sequence. In a driving state transition graph, nodes denote driving states, and the weights of edges can be the probability of state changes or transition duration between two driving states.*

Definition 4. Problem Statement. *In this paper, we study the problem of automated driving behavior profiling with GPS traces. Formally, given a driver (a vehicle) and corresponding GPS trajectories, we aim to find a mapping function $f : D \rightarrow V$ that takes the GPS trajectories $D = [< t, \varphi_t, \lambda_t >]_{t=1}^{T}$ as inputs, and outputs a sequence of time-varying yet relational vectorized representations $\mathbf{V} = [\mathbf{v}_n]_{n=1}^{N}$, in order to quantify the dynamics of the driver's driving behavior, where φ_t and λ_t respectively denote the latitude and longitude at the time t. We formulate this problem as a task of spatio-temporal representation learning. Essentially, we first construct a sequence of driving state transition graphs from GPS trajectories, and then learn the latent representations of driving behavior from the graphs.*

2.2 Framework Overview

Figure 1 shows an overview of our proposed framework that includes the following essential tasks: (i) constructing multi-view driving state transition graphs; (ii) automated profiling of driving behavior via peer and temporal-aware representation learning; (iii) solving the optimization problem; (iv) fusing representations from multi-view graphs; (v) applications to quantitative transportation safety. Specifically, in the first task, we detect driving operations from GPS sequences, identify driving states, and construct driving state transition graphs from the perspectives of transition frequency and duration. In the second task, we incorporate the modeling of both the graph-graph peer dependency and past-current temporal dependency into the Auto-Encoder based optimization framework to develop a spatio-temporal representation learning model. The proposed method jointly adopts and adapts the ideas of gated recurrent unit and spatial autocorrelation regularization. In the third task, we develop two optimization strategies for predictive tasks: (i) jointly optimization, minimizing the representation learning loss and the regression loss simultaneously; (ii) step-by-step optimization, first minimizing the representation learning loss and then minimizing the regression loss. In the forth task, we explore two strategies to fuse the learned representations from multi-view transition graphs: (i) simple alignment and (ii) collective fusion. In the fifth task, we exploit the learned representations of driving behavior graphs to enable important applications, including (i) prediction and historical assessment of driving scores, (ii) detecting risky regions.

3 Construction of Multi-view Driving State Transition Graphs

We propose a step-by-step testable analytic framework to convert GPS trajectories into driving state transition graphs. Specifically, for each driver, we first detect driving operations, identify driving states, and obtain a driving state sequence. The driving state sequence is over the time span in the dataset. Later, we segment the driving state sequence into small subsequences with a fixed time window, each of which is converted into a driving state transition graph. In the driving state transition graph, the subsequences of the time window is used to quantify the transition weights from a driving state to another. Since we have one driving state transition graph for each driving state subsequence, we can obtain a sequence of driving state transition graphs. The extracted sequence of driving state transition graphs is used to characterize the time-varying driving behavior of a driver.

Detection of Driving Operations.
From GPS trajectories, we identify two categories of driving operations: (i) speed-related operations that include "acceleration", "deceleration" and "constant speed"; (ii) direction-related operations that include "turning right", "turning left" and "moving straight". Formally, given three consecutive GPS points $< \varphi_1, \lambda_1 >, < \varphi_2, \lambda_2 >$ and $< \varphi_3, \lambda_3 >$ where φ_1, φ_2 and φ_3 respectively denote the three corresponding latitudes, λ_1, λ_2 and λ_3 respectively denote the three corresponding longitudes. We next show how to computationally detect the two types of driving operations.

(1) *Detection of speed-related operations.* Let $\Delta\varphi_{1,2}$ be the difference of φ_1 and φ_2, $\Delta\varphi_{2,3}$ be the difference of φ_2 and φ_3, $\Delta\lambda_{1,2}$ be the difference of λ_1 and λ_2, $\Delta\lambda_{2,3}$ be the difference of λ_2 and λ_3, and R be the radius of the earth. Then, the distance $d_{1,2}$ between the two GPS points $< \varphi_1, \lambda_1 >$ and $< \varphi_2, \lambda_2 >$ is given by Equation 1:

$$\begin{aligned} d_{1,2} =& 2R \cdot \operatorname{atan2}(\\ & \sqrt{\sin^2(\Delta\varphi_{1,2}/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda_{1,2}/2)}, \\ & \sqrt{1 - \sin^2(\Delta\varphi_{1,2}/2) - \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda_{1,2}/2)}), \end{aligned} \tag{1}$$

Similarly, the distance $d_{2,3}$ between the two GPS points $< \varphi_2, \lambda_2 >$ and $< \varphi_3, \lambda_3 >$ can also be calculated.

Then, given the time stamps of the three GPS points, denoted by t_1, t_2 and t_3, the speed s_2 at t_2 is given by $s_2 = d_{1,2}/(t_2 - t_1)$, the speed s_3 at t_3 is given by $s_3 = d_{2,3}/(t_3 - t_2)$. For t_3, if $s_3 > s_2$, the operation is detected as acceleration; if $s_3 < s_2$, the operation is deceleration; otherwise, the operation is "constant speed". In practice, due to the noise caused by GPS devices, we introduce a loosing boundary ϵ_s into the calculation. For t_3, if $s_3 > s_2$ and $|s_3 - s_2| > \epsilon_s$, the operation is detected as acceleration; if $s_3 < s_2$ and $|s_3 - s_2| > \epsilon_s$, the operation is deceleration; otherwise, the operation is "constant speed".

(2) *Detection of direction-related operations.* To detect the direction-related operations, we calculate the bearing $\theta_{1,2}$ between the two GPS points $< \varphi_1, \lambda_1 >$ and $< \varphi_2, \lambda_2 >$ by Equation 2:

$$\begin{aligned} \theta_{1,2} = \operatorname{atan2}(& \sin\Delta\lambda_{1,2} \cdot \cos\varphi_2, \cos\phi_1 \cdot \sin\varphi_2 \\ & - \sin\varphi_1 \cdot \cos\varphi_2 \cdot \cos\Delta\lambda_{1,2}). \end{aligned} \tag{2}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2935203, IEEE Transactions on Knowledge and Data Engineering

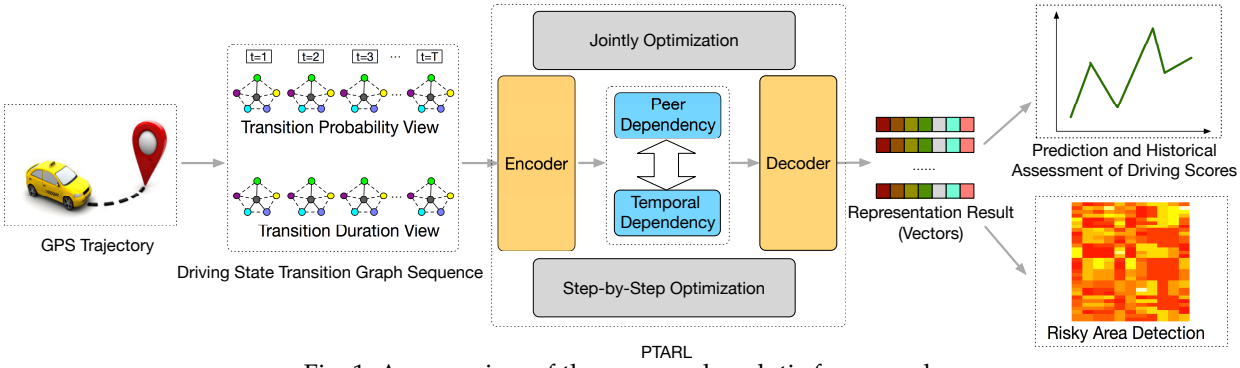


Fig. 1: An overview of the proposed analytic framework.

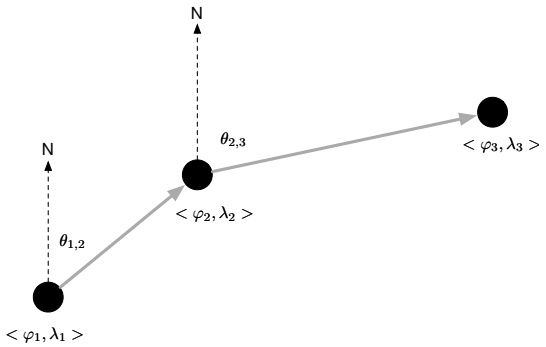


Fig. 2: An example for calculating directions.

Similarly, we can obtain the bearing $\theta_{2,3}$ between the two GPS points $< \varphi_2, \lambda_2 >$ and $< \varphi_3, \lambda_3 >$. Therefore, as shown in Figure 2, at t_3, if $\theta_{2,3} > \theta_{1,2}$, then the operation is "turning right"; if $\theta_{2,3} < \theta_{1,2}$, then the operation is "turning left"; otherwise, the operation is "moving straight". In practice, we also introduce a loosing boundary ϵ_d to estimate directions. At t_3, if if $\theta_{2,3} > \theta_{1,2}$ and $|\theta_3 - \theta_2| > \epsilon_d$, then the operation is "turning right"; if $\theta_{2,3} < \theta_{1,2}$ and $|\theta_3 - \theta_2| > \epsilon_d$, then the operation is "turning left"; otherwise, the operation is "moving straight".

Extraction of Driving State Sequences.

Based on the two speed-related operations and the three direction-related operations, we can define the following driving states in Table 2:

TABLE 2: Driving states.

(1)	acceleration while turning right
(2)	acceleration while turning left
(3)	acceleration while straightforward
(4)	deceleration while turning right
(5)	deceleration while turning left
(6)	deceleration while straightforward
(7)	constant speed while turning right
(8)	constant speed while turning left
(9)	constant speed while straightforward

With the above definitions, we can identify the driving state of a driver at each time stamp. In other words, each trajectory is associated with a driving state sequence, which is denoted by $\{(ID, t_n, S_n)\}_{n=1}^{N}$, where ID is the identity of the driver, N is the size of the driving state sequence, t_n is the n^{th} time stamp, and S_n is the driving state at t_n.

Construction of Multi-view Driving State Transition Graphs.

We next construct driving state transition graphs from driving state sequences. We first segment the driving state sequence into a set of partitions: $\{sq_i\}_{i=1}^{I}$, where each partition corresponds to a small time window ΔT (we will discuss the parameter setup of ΔT in the experimental settings). In these graphs, vertexes are regarded as driving states, and edges are regarded as transition relations. The transition relations can be formulated from two views: (i) transition probability, and (ii) transition duration.

(1) **A view of transition probability.** The transition probability of driving states shows how likely (frequency) a driver changes from one driving state to another, and thus can be used to characterize driving habits from a frequency perspective. For example, an aggressive driver might easily transit from "acceleration while straightforward " to "acceleration while turning left/right". Quantitatively, the transition probability among driving states can be estimated by the frequencies of state transitions. We normalize the transition frequencies as the transition probability. We conduct the normalization process as follows. Given a driver and time window, let $\text{Feq}_{l,m}$ denote the transition frequency from the driving state l to the driving state m. Then the transition probability $\text{Prob}_{l,m}$ from the driving state l to the driving state m can be represented as $\text{Prob}_{l,m} = \frac{\text{Feq}_{l,m}}{\sum_l \sum_m \text{Feq}_{l,m}}$.

(2) **A view of transition duration.** The transition duration of driving states is defined as the time between the beginning of the former state and the beginning of the latter state, which shows how long (duration) it takes for a driver to response from one driving state to another, and thus can be used to characterize driving habits from a time perspective. For example, if the transition duration between "acceleration while straightforward" and "deceleration while straightforward" is small, this reflects that the driver is impatient and does not care about passengers' feelings. Quantitatively, we can calculate the average time interval of the transitions between two driving states.

In summary, for each driver i at the time t, we define the driving state transition graph G_i^t where nodes are driving states, and edges are from two views. In the view of transition probability, each element of the transition graph (a matrix) G_i^t is the transition probability from one driving state to another; in the view of transition duration, each element of the transition graph (a matrix) G_i^t is the transition time from one driving state to another.

After that, we can obtain two graph sequences of driving state transition probability and driving state transition duration for each driver respectively.

4 Peer and Temporal-Aware Representation Learning

We present a spatio-temporal representation learning method to model the peer and temporal dependencies in representation learning.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2935203, IEEE Transactions on Knowledge and Data Engineering

4.1 Model Intuitions

There are peer and temporal dependencies among driving behavior. Therefore, in our approach, we model the representation of driving behavior based on the following intuitions.

Intuition 1: Structural Reservation After reducing driving behavior into graphs, we need a representation learning based method to transform graphs into vectors in a latent feature space for automated quantification and profiling. Consequently, the method should be able to project graphs into lower-dimensional vectors while reserving corresponding characteristics and structures.

Intuition 2: Peer Dependency. If two drivers exhibit similar driving habits, and the vehicle operation patterns of two corresponding trajectories are similar, then the driving state transition graphs of these two trajectories share a lot in terms of structures and characteristics. As a result, the learned representations of driving behavior should be close to each other. Consequently, the method should be able to model the graph-graph peer dependency in representation learning.

Intuition 3: Temporal Dependency. The driving operations of the current time slot have autocorrelation with previous driving states. For example, if a driver decelerates while straightforward at t, and if $\Delta(t, t+1)$ is small enough, then he is likely to accelerate at $t+1$. Consequently, the method should be able to model the current-past temporal dependency in representation learning.

4.2 Base Model

We utilize the deep Auto-Encoder model [3] as our base model. The motivation of using Auto-Encoder is that we aim to model the structural information of the driving state transition graph. Auto-Encoder shows the good performance of modeling structural information [1]. In addition, previous studies [18], [27] show that autoencoder is effective in modeling human mobility data, which fits the scenario in our work. Auto-Encoder is an unsupervised neural network model, which projects the instances in original feature representations into a lower-dimensional feature space via a series of non-linear mappings. The Auto-Encoder model involves two steps: encode and decode. The encode part projects the original feature vector to the objective feature space, while the decode step recovers the latent feature representation to a reconstruction space. In the auto-encoder model, we need to ensure that the original feature representation of instances should be as similar to the reconstructed feature representation as possible.

Formally, let $\mathbf{x_i}$ be the original feature representation of the i^{th} driver, and $\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^o$ be the latent feature representations of the diver at hidden layers $1, 2, \cdots, o$ in the encode step respectively, the encoding result in the objective lower-dimension feature space can be represented as $\mathbf{z_i} \in \mathbb{R}^d$ with dimension d. Formally, the relationship between these vector variables is denoted by:

$$\begin{cases} \mathbf{y}_i^1 & = \sigma(\mathbf{W}^1 \mathbf{x}_i + \mathbf{b}^1), \\ \mathbf{y}_i^k & = \sigma(\mathbf{W}^k \mathbf{y}_i^{k-1} + \mathbf{b}^k), \forall k \in \{2, 3, \cdots, o\}, \\ \mathbf{z_i} & = \sigma(\mathbf{W}^{o+1} \mathbf{y}_i^o + \mathbf{b}^{o+1}). \end{cases} \quad (3)$$

Meanwhile, in the decode step, the input will be the latent feature vector $\mathbf{z}_i$ (i.e., the output of the encode step), and the final output will be the reconstructed vector $\hat{\mathbf{x}}_i$. The latent feature vectors at each hidden layers can be represented as $\hat{\mathbf{y}}_i^o, \hat{\mathbf{y}}_i^{o-1}, \cdots, \hat{\mathbf{y}}_i^1$. The relationship between these vector variables is denoted by:

$$\begin{cases} \hat{\mathbf{y}}_i^o & = \sigma(\hat{\mathbf{W}}^{o+1} \mathbf{z}_i + \hat{\mathbf{b}}^{o+1}), \\ \hat{\mathbf{y}}_i^{k-1} & = \sigma(\hat{\mathbf{W}}^k \hat{\mathbf{y}}_i^k + \hat{\mathbf{b}}^k), \forall k \in \{2, 3, \cdots, o\}, \\ \hat{\mathbf{x}}_i & = \sigma(\hat{\mathbf{W}}^1 \hat{\mathbf{y}}_i^1 + \hat{\mathbf{b}}^1). \end{cases} \quad (4)$$

where $\mathbf{W}$s and $\mathbf{b}$s are the weight matrices and bias terms to be learned in the model.

The objective of the auto-encoder model is to minimize the loss between the original feature vector $\mathbf{x}$ and the reconstructed feature vector $\hat{\mathbf{x}}$. Formally, the loss function is

$$\mathcal{H}(\mathcal{U}) = \frac{1}{2} \sum_{u_i \in \mathcal{U}} \left\| (\mathbf{x}_i - \hat{\mathbf{x}}_i) \right\|_2^2 \quad (5)$$

where u_i denotes the i^{th} driver and $\mathcal{U}$ denotes the driver set.

4.3 Incorporating Temporal Dependency

The Auto-Encoder model is not able to capture the current-past temporal dependency. But, the Gated Recurrent Unit (GRU) is a variant of Long Short Term Memory networks (LSTMs), and can better connect previous information to the present task compared with the basic LSTMs [6]. To model the temporal dependency in representation learning, we incorporate GRU into the middle layer of auto-encoder, as shown in Figure 3.

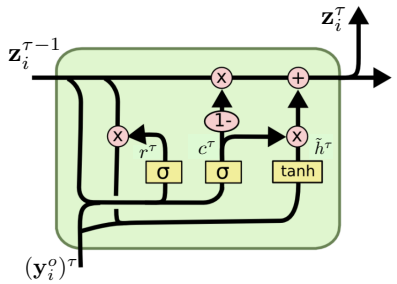


Fig. 3: Incorporation of GRU with Auto-Encoder.

The driving behavior transition graph sequence, denoted by $(G_i^0, G_i^1, \cdots, G_i^\tau, \cdots)$, represents the evolution of the time-varying driving behavior of the i-th driver u_i, at time slots $0, 1, \cdots, \tau, \cdots$, respectively. For each graph $G_i^\tau = [\{G_i^{\tau,1}\}^T, \{G_i^{\tau,2}\}^T, \cdots, \{G_i^{\tau,k}\}^T, \{G_i^{\tau,k}\}^T, \cdots, \{G_i^{\tau,9}\}^T]^T$, where $\{G_i^{\tau,k}\}^T$ is the k-th column vector to denote the k-th driving state, we flatten G_i^τ into one single vector $\mathbf{x}_i^\tau \in \mathbb{R}^{81 \times 1}$ by stacking the columns of G_i^τ on top of one another, following the rule of matrix vectorization. We utilize $\mathbf{x}_i^\tau$ as the original vector representation of the the driving behavior transition graph series, denoted by $(\mathbf{x}_i^0, \mathbf{x}_i^1, \cdots, \mathbf{x}_i^\tau, \cdots)$, where $\mathbf{x}_i^\tau$ is the original vector representation of G_i^τ.

In the evolution, the status of the driving behavior transition graph of the i^{th} driver (denoted by G_i^τ) at the time slot τ depends on the status of the graph at previous time slot τ (denoted by $G_i^{\tau-1}$). Formally, we can represent its status at these two time slots as $\mathbf{z}_i^{\tau-1}$ and $\mathbf{z}_i^\tau$ respectively. $\mathbf{z}_i^\tau$ evolves from $\mathbf{z}_i^{\tau-1}$, and the dependence relationship between them can be modeled with GRU. The temporal dependency between $\mathbf{z}_i^\tau$ and $\mathbf{z}_i^{\tau-1}$ can be denoted by

$$\mathbf{z}_i^\tau = (1 - c^\tau) \mathbf{z}_i^{\tau-1} + c^\tau \tilde{\mathbf{z}}_i^\tau, \quad (6)$$

where

$$\begin{cases} c^\tau = \sigma(\mathbf{W}_c[\mathbf{z}^{\tau-1}, (\mathbf{y}_i^o)^\tau]) \\ r^\tau = \sigma(\mathbf{W}_r[\mathbf{z}^{\tau-1}, (\mathbf{y}_i^o)^\tau]) \\ \tilde{\mathbf{z}}_i^\tau = tanh(\mathbf{W}[r^\tau \mathbf{z}^{\tau-1}, (\mathbf{y}_i^o)^\tau]). \end{cases} \tag{7}$$

Therefore, the temporal-aware Auto-Encoder is denoted by:

$$\begin{cases} & \#Sequential \quad Encode \quad Step \\ (\mathbf{y}_i^1)^\tau & = \sigma(\mathbf{W}^1 \mathbf{x}_i^\tau + \mathbf{b}^1), \\ (\mathbf{y}_i^k)^\tau & = \sigma(\mathbf{W}^k (\mathbf{y}_i^{k-1})^\tau + \mathbf{b}^k), \forall k \in \{2, 3, \cdots, o\}, \\ \mathbf{z}_i^\tau & = (1 - c^\tau)\mathbf{z}_i^{\tau-1} + c^\tau \tilde{\mathbf{z}}_i^\tau. \end{cases} \tag{8}$$

$$\begin{cases} & \#Sequential \quad Decode \quad Step \\ (\hat{\mathbf{y}}_i^o)^\tau & = \sigma(\hat{\mathbf{W}}^{o+1} \mathbf{z}_i^\tau + \hat{\mathbf{b}}^{o+1}), \\ (\hat{\mathbf{y}}_i^{k-1})^\tau & = \sigma(\hat{\mathbf{W}}^k (\hat{\mathbf{y}}_i^k)^\tau + \hat{\mathbf{b}}^k), \forall k \in \{2, 3, \cdots, o\}, \\ \hat{\mathbf{x}}_i^\tau & = \sigma(\hat{\mathbf{W}}^1 (\hat{\mathbf{y}}_i^1)^\tau + \hat{\mathbf{b}}^1). \end{cases} \tag{9}$$

where all outputs of each layer are labeled superscript by corresponding time slot. Then the loss function is:

$$\mathcal{H}(\mathcal{U}) = \frac{1}{2} \sum_{\tau \in \mathcal{T}} \sum_{u_i \in \mathcal{U}} \left\| (\mathbf{x}_i^\tau - \hat{\mathbf{x}}_i^\tau) \right\|_2^2 \tag{10}$$

4.4 Incorporating Peer Dependency

In the graph-graph peer dependency, trajectories that share similar driving behavior should have close representations in the learned representation feature space. Subject to such constraint, we introduce the loss function $\mathcal{H}_c(G^\tau)$ to model the peer dependency.

$$\mathcal{H}_c(G^\tau) = \sum_{u_i \in \mathcal{U}} \sum_{u_j \in \mathcal{U}, u_i \neq u_j} s_{i,j}^\tau \cdot \left\| \mathbf{z}_i^\tau - \mathbf{z}_j^\tau \right\|_2^2 \tag{11}$$

where $s_{i,j}^\tau$ is the function to evaluate the the similarity of driving behavior between the driver u_i and u_j at the time slot τ. We can define the function $s_{i,j}^\tau$ in many ways. For simplicity, in this paper, we define $s_{i,j}^\tau$ as the cosine similarity between the original representation vectors $\mathbf{x}_i^\tau$ and $\mathbf{x}_j^\tau$:

$$s_{i,j}^\tau = cos(\mathbf{x}_i^\tau, \mathbf{x}_j^\tau) \tag{12}$$

5 Optimization

We formulate the problem of driving performance assessment as a regression task to predict driving scores. Specifically, given the learned representations of driving behaviors, we aim to predict the driving scores of drivers. For simplicity, in our work, we choose linear regression to perform the regression task. Along this line, there are two parts of the loss: (i) representation learning loss and (ii) regression loss.

(i) Representation learning loss. The representation learning loss can be formulated as

$$\mathcal{H}_{rl} = \min \frac{1}{2} \sum_{\tau \in \mathcal{T}} \{ \sum_{u_i \in \mathcal{U}(n)} \left\| (\mathbf{x}_i^\tau - \hat{\mathbf{x}}_i^\tau) \right\|_2^2 + \alpha \cdot \mathcal{H}_c(G^\tau) \} \tag{13}$$

where α is the hyperparameter to control the regularizer $\mathcal{H}_c(G^\tau)$.

(ii) Regression loss. Let y_i denote the driving score of the driver i, and $\hat{y}_i$ denote the predicted driving score for the driver i. Then,

$$\hat{y}_i = z_i^\top \Lambda, \tag{14}$$

where Λ is the weight term of the linear regression. Then, the regression loss can be represented as

$$\mathcal{H}_{reg} = \frac{1}{2} \sum_i ||y_i - \hat{y}_i||^2 \tag{15}$$

There are two options to solve the optimization problem of representation learning: (i) jointly minimizing the loss of representation learning and regression, (ii) minimizing the loss step-by-step such that firstly for the loss of representation learning and then for the loss of the regression.

5.1 Jointly Optimization

The joint optimization objective function is:

$$\mathcal{H} = A \cdot \mathcal{H}_{rl} + B \cdot \mathcal{H}_{reg}, \tag{16}$$

where A and B are hyperparameters to control the weight of the representation learning loss and regression loss respectively. We utilize Stochastic Gradient Descent (SGD) to infer parameters.

5.2 Step-by-step Optimization

(1) Minimizing representation learning loss. To minimize the objective function, we utilize SGD to infer parameters. For parameters of decoder layers of PTARL, the updating rule is:

$$\hat{\mathbf{W}}_{new}^k = \hat{\mathbf{W}}_{old}^k - \lambda \cdot (- \sum_{u_i \in \mathcal{U}(n)} (\mathbf{x}_i^\tau - \hat{\mathbf{x}}_i^\tau) \cdot \hat{\mathbf{x}}_i^\tau (1 - \hat{\mathbf{x}}_i^\tau) \cdot \prod_{s=1}^{k-1} (\hat{\mathbf{y}}_i^s)^\tau (1 - (\hat{\mathbf{y}}_i^s)^\tau) \hat{\mathbf{W}}_{old}^s \cdot (\hat{\mathbf{y}}_i^k)^\tau) \tag{17}$$

$$\hat{\mathbf{b}}_{new}^k = \hat{\mathbf{b}}_{old}^k - \lambda \cdot (- \sum_{u_i \in \mathcal{U}(n)} (\mathbf{x}_i^\tau - \hat{\mathbf{x}}_i^\tau) \cdot (\hat{\mathbf{x}}_i^\tau)(1 - (\hat{\mathbf{x}}_i^\tau)) \cdot \prod_{s=1}^{k-1} (\hat{\mathbf{y}}_i^s)^\tau (1 - (\hat{\mathbf{y}}_i^s)^\tau) \hat{\mathbf{W}}_{new}^s) \tag{18}$$

For detailed parameter inferences, please refer to the appendix [1].

(2) Minimizing the regression loss. After we obtain the optimized learned representations $\mathbf{z}$, there is the closed form solution for linear regression:

$$\Lambda = ((\mathbf{z}^\tau)^T \mathbf{z}^\tau)^{-1} (\mathbf{z}^\tau)^T \mathbf{y}. \tag{19}$$

6 Multi-View Fusion

In Section 3, we construct multi-view driving state transition graphs, including (1) the view of transition probability, and (2) the view of transition duration. For each view of the graph, we can obtain its corresponding representation through our proposed framework. In another word, we have two learned vectorized representations in terms of two views. The challenge is to how to generate a fused representation incorporating two views. In this section, we introduce two strategies, (1) simple alignment and (2) collective fusion.

6.1 Simple alignment

Let $(\mathbf{z}_i^\tau)_{\text{prob}}$ denote the learned representation of transition-probability-view graph, $(\mathbf{z}_i^\tau)_{\text{dur}}$ denote the learned representation of transition-duration-view graph, then the fused representation can be represented as

$$\mathbf{z}_i^\tau = [((\mathbf{z}_i^\tau)_{\text{prob}})^\top, ((\mathbf{z}_i^\tau)_{\text{dur}})^\top]^\top \tag{20}$$

[1] https://goo.gl/cnECP8

6.2 Collective Fusion

We adopt the idea of collective representation learning in the work [27]. We add an feature ensembling procedure before the encoder, and attach an feature dispatching procedure after the decoder, as shown in Figure 4, other parts of the framework remain the same as introduced in Section 4.

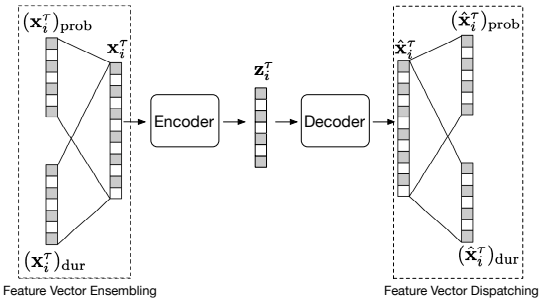


Fig. 4: Collective fusion of of multi-view graph.

Formally, let $(\mathbf{x}_i^\tau)_{\text{prob}}$ denote the original feature vector (flattened graph) of the transition-probability-view graph, $(\mathbf{x}_i^\tau)_{\text{dur}}$ denote the original feature vector (flattened graph) of the transition-duration-view graph, $(\hat{\mathbf{x}}_i^\tau)_{\text{prob}}$ denote the reconstructed feature vector of the transition-probability-view graph, $(\hat{\mathbf{x}}_i^\tau)_{\text{prob}}$ denote the reconstructed feature vector of the transition-probability-view graph. Then, the feature ensembling procedure can be represented as

$$\mathbf{x}_i^\tau = \sigma(\mathbf{W}_1(\mathbf{x}_i^\tau)_{\text{prob}} + \mathbf{W}_2(\mathbf{x}_i^\tau)_{\text{dur}} + b) \tag{21}$$

the feature dispatching procedure can be represented as

$$\begin{cases} (\hat{\mathbf{x}}_i^\tau)_{\text{prob}} & = \sigma(\hat{\mathbf{W}}_1(\hat{\mathbf{x}}_i^\tau) + \hat{b}_1) \\ (\hat{\mathbf{x}}_i^\tau)_{\text{dur}} & = \sigma(\hat{\mathbf{W}}_2(\hat{\mathbf{x}}_i^\tau) + \hat{b}_2). \end{cases} \tag{22}$$

In the experiment, we study the performance of these two fusion strategies.

7 Applications

We demonstrate the two applications in transportation safety: (i) prediction and historical assessment of driving scores and (ii) risky area detection. To that end, we invite the domain experts from the Department of Transportation (DoT) to provide a driving score for each driver, by examining their driving operations across the entire time span.

7.1 Prediction and Historical Assessment of Driving Scores

Our proposed method can learn a series of vectorized representations for a driver at each time slot and a regression model to predict driving of the last time slot. Therefore, we can apply previous learned vectorized representation to the regression model, to assess the historical driving scores at a specific previous time slot in a backward direction.

7.2 Risky Area Detection

It is important to understand how driving scores are distributed spatially and temporally. Therefore, we study the spatio-temporal dynamics of driving scores across all the areas, so as to detect risky areas. Specifically, if the vehicles in a given area are operated by low-score drivers, this area is likely to be risky. To detect risky areas, we first apply the trained driving scores predictive model to predict driving scores for a specific driver, at a specific time slot, and at a specific location. Moreover, we compute the average driving scores $\bar{\gamma}_l^\tau$ of all the drivers for a specific location l and a specific time slot τ. In addition, since the occurrence of traffic accidents follows a Poisson distribution [7], we define the threshold γ^τ for detecting risky areas at the time slot τ as the lower bound of the 95% confidence interval: $\gamma^\tau = \mu^\tau - 1.96\sigma^\tau$, where μ^τ and σ^τ are the mean and the standard deviation of the average driving scores at time τ respectively. If $\bar{\gamma}_l^\tau < \gamma^\tau$, then we detect the area l as a risky one; otherwise non-risky. We visualize the detection results using heat maps in the experiment.

TABLE 3: Statistics of the experimental data.

Properties	Statistics
Number of drivers	10,357
Time range	Feb.2 - Feb.8
Sampling Interval	117 Seconds
Sampling Distance	623 Meters
Total Trajectories	15 Million
Total Distance	9 million KM
Records are within a time window	83 in Average
City	Beijing

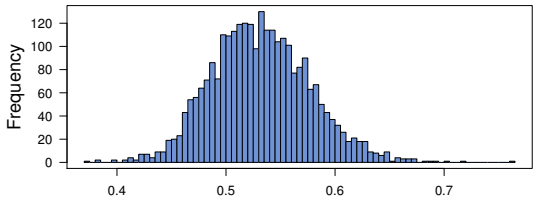


Fig. 5: Driving score distribution.

8 Experimental Results

This section details our empirical evaluation of the proposed method on real-world data.

8.1 Data Description

Table 3 shows the statistics of our real-world data sets *T-Drive* trajectory dataset [29], [30]. This dataset contains the GPS trajectories of 10,357 taxis during the period of Feb. 2 to Feb. 8, 2008 within Beijing. The total number of points in this dataset is about 15 million and the total distance of the trajectories reaches to 9 million kilometers. The average sampling interval is about 177 seconds with a distance of about 623 meters. Each GPS point contains the information of corresponding driver ID, latitude, longitude, and time stamp. To prepare benchmark driving scores, we invite the domain experts from DoT to help us evaluate the visualizations of trajectories, and assign a driving performance score ranging from 0 to 1 to each driver. The higher the score is, the safer the driver is. Figure 5 shows the distribution of the driving scores: only a small number of the drivers have very high or very low scores, while most scores are moderate and range from 0.45 to 0.6.

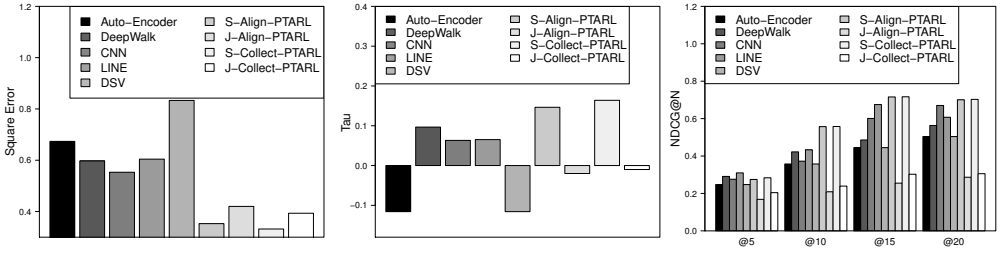
8.2 Evaluation Metrics

Let us assume that each driver i is associated with a benchmark score y_i and a predicted score f_i. To show the effectiveness of the proposed model, we use the following metrics for evaluation.

Square Error. We utilize *Square Error* (SE) to measure regression errors. $SE = \sum_i (y_i - f_i)^2$, where N is the number of drivers. The lower the SE is, the better the learned representation is.

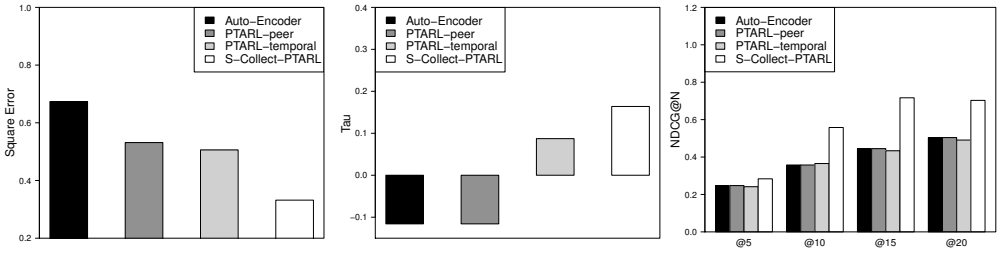
Kendalls Tau Coefficient. We utilize *Kendall's Tau Coefficient (Tau)* to measure the overall ranking accuracy. For a driver

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2935203, IEEE Transactions on Knowledge and Data Engineering



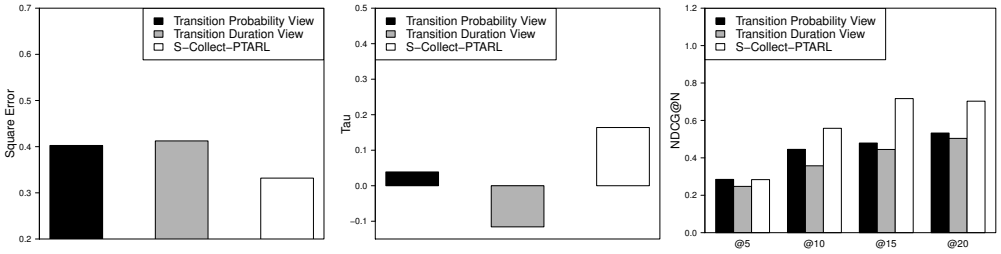
(a) SE (b) Tau (c) NDCG@N

Fig. 6: Overall comparison.



(a) SE (b) Tau (c) NDCG@N

Fig. 7: Performance with different dependencies.



(a) SE (b) Tau (c) NDCG@N

Fig. 8: Performance in different views.

pair $< i, j >$, $< i, j >$ is said to be concordant, if both $y_i > y_j$ and $f_i > f_j$ or if both $y_i < y_j$ and $f_i < f_j$. Also, $< i, j >$ is said to be discordant, if both $y_i < y_j$ and $f_i > f_j$ or if both $y_i < y_j$ and $f_i > f_j$. Tau is given by $\text{Tau} = \frac{\#_{conc} - \#_{disc}}{\#_{conc} + \#_{disc}}$.

Normalized Discounted Cumulative Gain. We utilize *Normalized Discounted Cumulative Gain* (NDCG@N) to measure the ranking accuracy at the top-N cases. The discounted cumulative gain (DCG@N) is given by $NDCG[N] = \sum_{i'=1}^{N} \frac{y_{i'}}{log_2(1+i')} / \sum_{i=1}^{N} \frac{y_i}{log_2(1+i)}$ where i denotes the original ranking order of the benchmark and $i^{'}$ denotes the ranking order of the prediction. The larger NDCG@N is, the higher top-N ranking accuracy is.

8.3 Baseline Algorithms

We compare the performances of our method against the following baseline algorithms.

(1) Auto-Encoder. The Auto-Encoder model [3] minimizes the loss between the original feature representations and reconstructed ones. In the experiments, we set the number of hidden layers $= 4$, the size of middle layer $= 20$.

(2) DeepWalk. The DeepWalk model [19] extends the word2vec model [14] to the scenario of network embedding. DeepWalk uses local information obtained from truncated random walks to learn latent representations. In the experiments, we set the number of walks $= 80$, the size of representation $= 20$, the walk length $= 40$, and the window size $= 10$.

(3) LINE. The LINE model optimizes the objective function that preserves both the local and global network structures with an edge-sampling algorithm [21]. In the experiments, we set the size of representation $= 20$, the number of negative samples $= 5$, and the starting value of the learning rate $= 0.025$.

(4) CNN. The CNN model refers to Convolutional Neural Network, which projects original feature space into a new space via a variation of multilayer perceptrons [11]. In the experiments, we set the number of convolutional layer $= 2$.

(5) Driving State Vector (DSV). In addition, we also compare our model with the traditional transportation approach. We adopt the driving states defined in [4], [36] to profile driving behavior, including (1)acceleration while turning, (2)acceleration while straightforward, (3) deceleration while turning, (4) deceleration while straightforward, (5) constant speed while turning, and (6) constant speed while straightforward. We formulate a driving state vector (DSV) for each driver, where each entry in the vector is the percentage of the corresponding driving state. We feed DSVs and corresponding driving scores into SVR to train the regression model.

Besides, we also evaluate four variants of our method in terms of the different optimization options and view fusion strategies. For simplicity, we denote the "jointly optimization" with "simple alignment view fusion" model as **J-Align-PTARL**, the "jointly optimization" with "collective view fusion" model as **J-Collect-PTARL**, the "step-by-step optimization" with "simple alignment view fusion" model as **S-Align-PTARL**, and the "step-by-step optimization" with "collective view fusion" model as **S-Collect-PTARL**.

To adapt the proposed model to the real data, we first set the time window $\delta_t = 30$ mins, and segment the GPS records based on the time window. Then, we construct multi-view driving state transition graphs based on the methods introduced in Section 3. Finally, for each user, we will have a sequential of multi-view driving state transitions graphs as the input of our proposed framework. For the structure of PTARL, we set the number of hidden layers of encoder $= 1$, the number of hidden layers of decoder $= 2$, the output size of vectors $= 20$. the penalty parameter $\alpha = 0.01$ for regularizer $\mathcal{H}_c(G^\tau)$, the learning rate $= 0.0001$. For **J-⋆-PTARL**, we set $A = 0.3, B = 0.7$. The source code of PTARL is available at link [2].

8.4 Overall Performances

We compare our method with the baseline methods in terms of SE, Tau and NDCG@N.

From Figure 6 we can obtain two interesting observations that (1) the performance of jointly optimization strategy is better than step-by-step optimization, and (2) the performance of collective-view-fusion is better than simple alignment.

For the comparison of optimization strategies, the potential explanation is that the jointly-optimization strategy considers both the representation learning loss and the regression loss, leading to better performance. Nevertheless there is a trade-off between the representation learning and regression for jointly-optimization, which makes the advantage of jointly optimization is not obvious for regression accuracy.

For the comparison of view-fusion strategies, the potential explanation is that there are correlations between the transition probability view and the transition duration view. The transition duration may be higher if driving state transits at a small probability, vice versa. The collective fusion strategy is able to fuse the correlations of these two view in a systematical way, while the simple alignment strategy ignores the correlations.

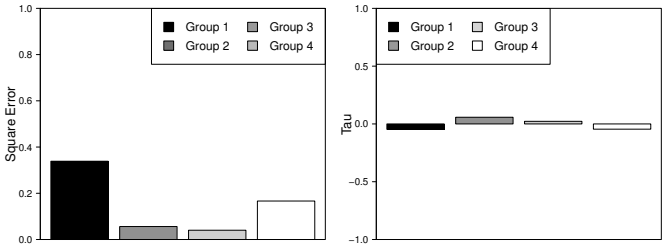
Auto-Encoder, DeepWalk, CNN, and LINE are not able to model sequential inputs. Therefore, in the experiment, we aggregate all the driving state transitions across all the time slots into one transition graph for these baselines. Since the aggregation will lose the peer and temporal dependencies of driving behavior, the performances of these baselines are severely disrupted. However, due to the previous analysis in this paper, the current driving behavior is influenced by previous ones, the aggregation loses temporal dependency of driving behavior.

Since **S-Collect-PTARL** achieves the best performance, we utilize **S-Collect-PTARL** to conduct the following evaluation.

8.5 Robustness Check

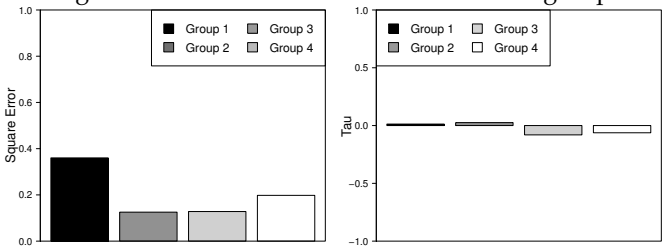
We apply the learned representations of driving behavior and the linear regression model to different subgroups of the drivers, to examine the robustness of our method in these subgroups. We use two grouping methods: (i) driving score based grouping, (ii) driving state based grouping. In the grouping method (i), we segment drivers into four subgroups by the driving score y_i, i.e. $y_i < 0.45$, $0.45 \leq y_i < 0.55$, $0.55 \leq y_i < 0.65$, and $y_i \geq 0.65$. In the grouping method (ii), we generate driving state vectors (introduced in Section 6.3) for each driver; we then apply K-Means [9] to cluster drivers into four groups based on their driving vectors.

[2]https://github.com/Merlin55/PTARL



(a) SE (b) Tau

Fig. 9: Robustness check in the score-based group.



(a) SE (b) Tau

Fig. 10: Robustness check in the driving-state-based group.

Figure 9 shows that for driving score based grouping, our model can achieve a relative stable performance, especially in terms of Tau. For driving state based grouping, the results validate the assumption that if two drivers show similar driving behavior, their predicted driving scores are similar as well.

8.6 Study of Peer and Temporal Dependencies

We study the effects of peer and temporal dependencies on the model, by comparing our **S-Collect-PTARL** with Auto-Encoder and two other variants of **S-Collect-PTARL**, i.e. (i) **PTARL-peer** that only considers the peer dependency, and (ii) **PTARL-temporal** that only considers the temporal dependency.

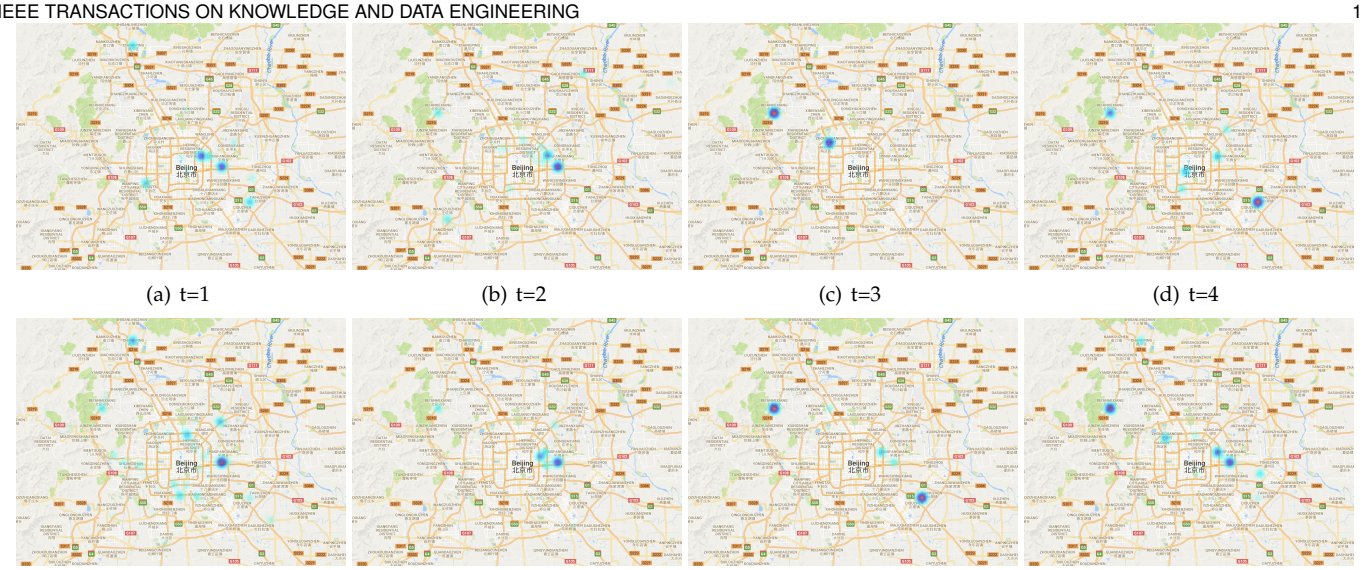
From Figure 7, we can observe that **S-Collect-PTARL** outperforms Auto-Encoder, PTARL-peer and PTARL-temporal significantly. Because of ignoring both peer and temporal dependencies, Auto-Encoder performs worst in the comparison. Also, the results show that PTARL-temporal performs better than PTARL-peer, which means that the temporal dependency is more significant in profiling driving behavior than the peer dependency.

8.7 Study of Performance in Different Views

We introduce two views in driving state transition graphs: transition probability view and transition duration view. Therefore, we investigate the performance of our model under these two views.

From Figure 8, it is interesting to observe that the performances of the transition probability view or transition duration view are worse than the combination of these two views. And, the difference of errors between two views are subtle. The possible explanation is that only one view, no matter probability view or time view, can not capture the complete information of driving behavior. By combining

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2935203, IEEE Transactions on Knowledge and Data Engineering



(a) t=1 (b) t=2 (c) t=3 (d) t=4

(e) t=5 (f) t=6 (g) t=7 (h) t=8

Fig. 11: Risky area detection.

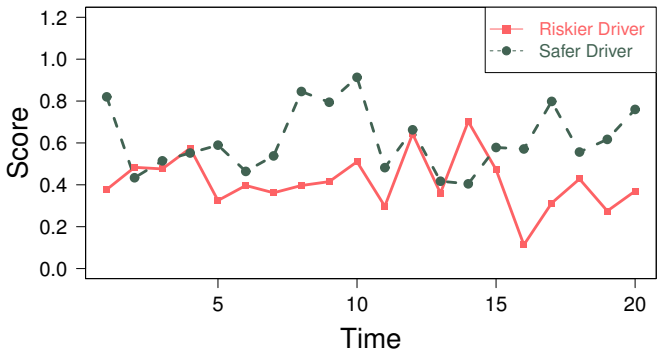


Fig. 12: An example of historical assessment of driving scores over time for a safer driver and a riskier driver.

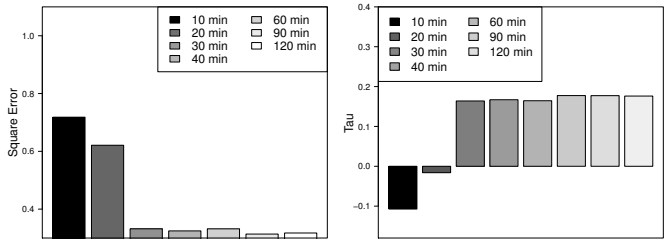
these two views, our model can systematically make up the deficiency of each single view.

However, for Tau, the performance of model in each view is significantly worse than **S-Collect-PTARL**. It is obvious that single view based model messes up the original order of driving scores, even though it can achieve well in square errors. **S-Collect-PTARL** makes up the defect under single view and ensure the predicted ranking order that makes much more sense for the driving score prediction.

8.8 Study of Time Window δ_t

In this section, we study the effects of time window δ_t on the performance of the proposed model. Specifically, we select the following time windows: 10 min, 20 min, 30 min, 40 min, 60 min, 90 min, and 120 min. We show the results in Figure 13.

As shown in Figure 13, with the very small time window, i.e. $\delta_t < 30min$, the performance is compromised badly. With the time window getting larger, the performance of the model is getting better. However, when the time window $\delta_t > 30min$, the model performance does not change much. The potential explanation is that when the time window is very small, the data will be too sparse to construct driving state transition graph. When the time window is larger than a specific threshold, the information is sufficient that the improvement of the model performance is not obvious.



(a) SE (b) Tau

Fig. 13: Study of time window δ_t.

8.9 Historical Assessment of Driving Scores

Driving scores are assigned by domain experts in terms of the whole dataset across all time slots. This means that every driver only has one driving score. Practically, a driver should have a driving score at each time. Fortunately, with our proposed PTARL, we can predict and assess the driving score for every driving at all time slots. We select the driver with the highest driving score and the driver with the lowest driving score. Specifically, we name the driver with the highest driving score as "Safer Driver", and the other one with lowest driving score as "Riskier Driver". We utilize their learned representation sequences to assess their historical driving sores. We report the experimental results in Figure 12.

There is an interesting observation that a "Safer Driver" is not always safe and a "Riskier Driver" is not always risky. The driving scores are varying over time. Sometimes, the driving score of the "Riskier Driver" is higher than the "Safer Driver". There is a pattern that scores of the "Safer Driver" are relatively higher at most time, while the scores of the "Riskier Driver" are relatively lower at most time. This observation is consistent with our common sense that driving behavior is affected by random factors, like weather, road condition and mind status, while the driving habit has a relatively stable pattern.

8.10 Risky Area Detection

Figure 11 shows the visualization results of risky area detection. We can observe the dynamic evolution of the distribution of risky areas over time. Behind the evolution, there are two "varying" lines: "varying" driving behavior

and "varying" locations for drivers. Figure 12 shows that driving behavior varying over time. Meanwhile locations of drivers are also changing, because drivers are always moving. Therefore, the mixture of two "varying" leads to the evolution of risky areas. For this reason, it is challenging to detect risky areas. But with our proposed method, we can detect risky areas over time, which can enhance the car accident warning and the administration of transportation.

9 Related Work

The related work can be categorized into the following categories: (i) graph representation learning, (ii) urban computing, and (iii) human mobility modeling.

Graph representation learning. Our work is relevant to graph representation learning. Graph representation learning, also known as graph/network embedding, aims to learn a low-dimensional vector to represent vertexes or graphs. *Wang et al.* proposed a deep model with a semi-supervised architecture, which simultaneously optimizes the first-order and second-order proximity [22]. *Ou et al.* preserved asymmetric transitivity by approximating high-order proximity which are based on asymmetric transitivity [16]. *Wang et al.* developed a Multi-task Representation Learning (MTRL) model to predict users demographic attributes [25]. *Zhang et al.* proposed an alternate FK-based aggregation method for document representation based on neural word embeddings [34]. *Wang et al.* used a regression learner to learn the optimized layout of heterogeneous elements on the search result page (SERP) [28]. *Zhang et al.* proposed TrioVecEvent, a method that leverages multimodal embeddings to achieve accurate online local event detection [33]. The work in [2] used an unsupervised learning method to obtain a hierarchy of features one level at a time and to learn a new transformation at each level to be composed with the previously learned transformations.

Human mobility modeling. Our work is related to the research of human mobility modeling. There are existing studies on human mobility modeling by exploiting mobility patterns to enable various applications [35]. *Wang et al.* encoded the dynamic mobility flow into vector representations of regions through a embedding method [23]. [10] studied the problem of destination prediction in Bike-Sharing Systems (BSSs), proposing a multi-view machine (MVM) method, by incorporating the context information from Point of Interest (POI) data and human mobility data into destination prediction. *Yuan et al.* analogized human mobility patterns as words, and exploited both topic modeling and spectrum analysis to analyze the urban functions of regions [31]. [26] and [24] developed a joint model that integrates Mixture of Hawkes Process (MHP) with a hierarchical topic model to capture the arrival sequences with mixed trip purposes. [5] developed a geographic method named ClusRanking to exploite the geographic dependencies of the value of an estate with online user reviews and offline moving behaviors. *Yuan et al.* proposed a Bayesian non-parametric model, named Periodic Region Detection (PRED), to discover periodic mobility patterns by jointly modeling the geographical and temporal information [32]. *Lin et al.* proposed a new passive verification method that requires minimal imposition of users through modeling users subtle mobility patterns [12].

Driving behavior analysis. Finally, our work has a connection with driving behavior analysis. Prior driving behavior analysis research can be summarized as non-contextual and contextual approaches, according to whether driving information is context relevant. For non-contextual methods, they solely applied vehicle kinematic information like speed and acceleration/deceleration, to model driving behavior. For example, *Ellison et al.* applied Temporal and Spatial Identifiers to provide a common measurement of driving behavior, using vehicle motion information [4]. *Paefgen et al.* performed driver risk profiling by constructing features from real-world GPS data that included accident and accident-free driver [17]. For contextual approaches, they added contextual information like weather and road quality to models, except vehicle motion data. For example, *Zhu et al.* proposed a Bayesian Network model which combined GPS driving observations, individual driving behavior and individual driving risks with contextual features such as road conditions and dynamic traffic flow information [36]. *Jun et al.* evaluated driving exposure and performance differences between who were involved in crashes or not, with the detailed exposure data of individual drivers (travel by time of day and by roadway type) alongside driving performance data (speed, throttle, braking, and acceleration) [8].

10 Conclusion Remarks

Driving behavior analysis has been important for assessing driver performances, improving traffic safety, and developing the intelligent and resilient transportation systems. In this paper, we investigated driving behavior analysis from the perspective of representation learning. We formulated the problem of driving behavior profiling and scoring as a task of spatial and temporal embedding and labeling with driving state transition graphs. We studied large-scale driving behavior data, and identified the peer and temporal dependencies. To improve the performance of automated behavior profiling, we developed an analytic framework that jointly modeled the peer and temporal dependencies. Specifically, we first construct multi-view driving state transition graphs from GPS traces to characterize driving behavior. Besides, we incorporated the idea of gated recurrent unit to model both the graph-graph peer dependency and integrate graph-graph peer penalties to capture the current-past temporal dependency in two optimization strategies, i.e. (i) jointly optimization and (ii) step-by-step optimization. In addition, we applied our proposed method to enable the applications of driving score prediction and risky area detection. The empirical experiments on real-world data demonstrated the effectiveness of spatio-temporal representation learning for profiling driving behavior.

11 Acknowledgment

This research was partially supported by the National Science Foundation (NSF) via the grant number: 1755946. This research was partially supported by the National Science Foundation (NSF) via the grant number: IIS-1814510.

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[2] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[4] Adrian B Ellison, Michiel CJ Bliemer, and Stephen P Greaves. Evaluating changes in driver behaviour: a risk profiling approach. *Accident Analysis & Prevention*, 75:298–309, 2015.

[5] Yanjie Fu, Yong Ge, Yu Zheng, Zijun Yao, Yanchi Liu, Hui Xiong, and Jing Yuan. Sparse real estate ranking with online user reviews and offline moving behaviors. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 120–129. IEEE, 2014.

[6] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 2017.

[7] Ismail Bulent Gundogdu. Applying linear analysis methods to gis-supported procedures for preventing traffic accidents: Case study of konya. *Safety Science*, 48(6):763–769, 2010.

[8] Jungwook Jun, Jennifer Ogle, and Randall Guensler. Relationships between crash involvement and temporal-spatial driving behavior activity patterns: use of data for vehicles with global positioning systems. *Transportation Research Record: Journal of the Transportation Research Board*, (2019):246–255, 2007.

[9] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[10] Jiawei Zhang Yanjie Fu Kunpeng Liu, Pengyang Wang and Sajal K. Das. Modeling the interaction coupling of multi-view spatiotemporal contexts for destination prediction. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2018.

[11] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[12] Miao Lin, Hong Cao, Vincent W Zheng, Kevin Chen-Chuan Chang, and Shonali Krishnaswamy. Mobility profiling for user verification with anonymized location data. In *IJCAI*, pages 960–966, 2015.

[13] HaiLong Liu, Tadahiro Taniguchi, Yusuke Tanaka, Kazuhito Takenaka, and Takashi Bando. Visualization of driving behavior based on hidden feature extraction by using deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2477–2489, 2017.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[15] Hideaki Misawa, Kazuhito Takenaka, Tomoya Sugihara, Hailong Liu, Tadahiro Taniguchi, and Takashi Bando. Prediction of driving behavior based on sequence to sequence model with parametric bias. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.

[16] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114. ACM, 2016.

[17] Johannes Paefgen, Florian Michahelles, and Thorsten Staake. Gps trajectory feature extraction for driver risk profiling. In *Proceedings of the 2011 international workshop on Trajectory data mining and analysis*, pages 53–56. ACM, 2011.

[18] Guannan Liu Yanjie Fu Charu Aggarwal Pengyang Wang, Jiawei Zhang. Ensemble-spotting: Ranking urban vibrancy via poi embedding with multi-view spatial graphs. In *Proceedings of 2018 SIAM International Conference on Data Mining (SDM'18)*. SIAM, 2018.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[20] Sarah M Simmons, Anne Hicks, and Jeff K Caird. Safety-critical event risk associated with cell phone tasks as measured in naturalistic driving studies: A systematic review and meta-analysis. *Accident Analysis & Prevention*, 87:161–169, 2016.

[21] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*. ACM, 2015.

[22] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM, 2016.

[23] Hongjian Wang and Zhenhui Li. Region representation learning via mobility flow. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 237–246. ACM, 2017.

[24] Pengfei Wang, Yanjie Fu, Guannan Liu, Wenqing Hu, and Charu Aggarwal. Human mobility synchronization and trip purpose detection with mixture of hawkes processes. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 495–503. ACM, 2017.

[25] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Multi-task representation learning for demographic prediction. In *European Conference on Information Retrieval*, pages 88–99. Springer, 2016.

[26] Pengfei Wang, Guannan Liu, Yanjie Fu, Yuanchun Zhou, and Jianhui Li. Spotting trip purposes from taxi trajectories: A general probabilistic model. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(3):29, 2018.

[27] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Xiaolin Li, and Dan Lin. Learning urban community structures: A collective embedding perspective with periodic spatial-temporal mobility graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(6):63, 2018.

[28] Yue Wang, Dawei Yin, Luo Jie, Pengyuan Wang, Makoto Yamada, Yi Chang, and Qiaozhu Mei. Beyond ranking: Optimizing whole-page presentation. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 103–112. ACM, 2016.

[29] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2011.

[30] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, pages 99–108. ACM, 2010.

[31] Nicholas Jing Yuan, Yu Zheng, Xing Xie, Yingzi Wang, Kai Zheng, and Hui Xiong. Discovering urban functional zones using latent activity trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):712–725, 2015.

[32] Quan Yuan, Wei Zhang, Chao Zhang, Xinhe Geng, Gao Cong, and Jiawei Han. Pred: Periodic region detection for mobility modeling of social media users. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 263–272. ACM, 2017.

[33] Chao Zhang, Liyuan Liu, Dongming Lei, Quan Yuan, Honglei Zhuang, Tim Hanratty, and Jiawei Han. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 595–604. ACM, 2017.

[34] Ruqing Zhang, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Aggregating neural word embeddings for document representation. In *European Conference on Information Retrieval*, pages 303–315. Springer, 2018.

[35] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.

[36] Xiaoyu Zhu, Yifei Yuan, Xianbiao Hu, Yi-Chang Chiu, and Yu-Luen Ma. A bayesian network model for contextual versus non-contextual driving behavior assessment. *Transportation Research Part C: Emerging Technologies*, 81:172–187, 2017.

Pengyang Wang received the BE degree from the Xi'dian University, China, 2014, the MS degree from the Beijing University of Posts and Telecommunications, China, 2017. He is currently working toward the PhD degree in the Computer Science Department at Missouri University of Science and Technology. His research interests include data mining, business analytics, geoeconomics, and representation learning.

Xiaolin Li is currently an Associate Professor in the School of Management at Nanjing University, China. Her main research interests include business intelligence, data mining, and decision-making. She has published a number of papers in refereed journals and conference proceedings, such as Information Sciences (INS), ECML, and PAKDD. She also has served as a reviewer and PC member for numerous journals and conferences, such as IEEE TEC, KAIS, and KDD15.

Yu Zheng is a Vice President and Chief Data Scientist in JD Finance Group and a Chair Professor at Shanghai Jiao Tong University. His research interests include big data analytics, spatio-temporal data mining, and ubiquitous computing. He leads the research on urban computing in Microsoft Research, passionate about using big data to tackle urban challenges. He frequently publishes referred papers as a leading author at prestigious conferences and journals, such as KDD, VLDB, UbiComp, and IEEE TKDE. He received five best paper awards from ICDE'13 and ACM SIGSPATIAL10, etc. Zheng currently serves as an Editor-in-Chief of ACM Transactions on Intelligent Systems and Technology and as a member of Editorial Advisory Board of IEEE Spectrum. He is also an Editorial Board Member of GeoInformatica and IEEE Transactions on Big Data. He has served as chair on over 10 well-known international conferences, e.g. the program co-chair of ICDE 2014 (Industrial Track) and UIC 2014. He has been invited to give over 10 keynote speeches at international conferences and forums (e.g. IE 2014 and APEC 2014 Smart City Forum). He is an IEEE senior member and ACM senior member. In 2013, he was named one of the Top Innovators under 35 by MIT Technology Review (TR35) and featured by Time Magazine for his research on urban computing. In 2014, he was named one of the top 40 Business Elites under 40 in China by Fortune Magazine, because of the business impact of urban computing he has been advocating. Zheng is also an Adjunct Professor at Hong Kong Polytechnic University, a visiting Chair Professor at XiDian University and an Affiliate Professor at Southwest Jiaotong University.

Charu C. Aggarwal received the BTech degree in computer science from the Indian Institute of Technology (1993) and the PhD degree in operations research from the Massachusetts Institute of Technology (1996). He has been a research staff member at the IBM T.J. Watson Research Center since June 1996. He has applied for or been granted 40 US patents and has published in numerous international conferences and journals. He has been designated master inventor at IBM Research. His current research interests include algorithms, data mining, and information retrieval. He is interested in the use of data mining techniques for Web and e-commerce applications.

Yanjie Fu received his Ph.D. degree from Rutgers University in 2016, the B.E. degree from Univ. of Sci. and Tech. of China in 2008, and the M.E. degree from Chinese Academy of Sciences in 2011. He is currently an Assistant Professor at the Missouri University of Science and Technology. His general interests are data mining and big data analytics. He has published prolifically in refereed journals and conference proceedings, such as IEEE TKDE, ACM TKDD, IEEE TMC, ACM SIGKDD.