Telematics and Contextual Data Analysis and Driving Risk Prediction

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

SeyedSobhan MoosaviNejadDaryakenari, M.S.

Graduate Program in Computer Science and Engineering

The Ohio State University

2020

Dissertation Committee:

Rajiv Ramnath, Advisor

Srinivasan Parthasarathy

Wei-Lun Chao

Abstract

Analysis of *telematics* data collected from drivers in real-time, along with *contextual* data (such as traffic, weather, and road-network characteristics), provides valuable insights regarding an individual's driving behavior, common driving habits, and characteristics of a road network. The primary focus in this dissertation is on predicting the risk in driving (which we term *driving risk prediction*), with the risk being a combination of the risk of the driver, the risk in a route, and the risk caused by driving conditions. We propose several data-analytic techniques for driving risk prediction and related caused and apply them to different sources of telematics and contextual data to extract useful insights. These insights may be used for applications such as usage-based insurance, driver coaching, and urban planning, potentially with relevance to the creation of smart cities. To be specific, we seek to answer the following research question: *How can telematics data and their context be modeled to make a fair and sound prediction about driving risk?* In answering this question, we take two concerns into account: 1) scoring of driving risk is not a context-agnostic process, and 2) driving risk is not independent of the personality of drivers and their driving skills. Thus, a reasonable solution for driving risk prediction must address these concerns. To design such a framework, we propose a solution that consists of three parts: a) characterizing driving context, b) characterizing driving style, and c) context-aware driving risk prediction. The first two parts of our proposal derive useful insights that we leverage to design the third part.

Driving context can be described as a combination of location (e.g., Interstate-90) and time (e.g., weekdays between 3pm to 7pm). Characterizing driving context is about exploring the properties of different contexts. We propose two solutions for this task. The first solution, which we term *segmentation and causality analysis*, derives the characteristics of contexts from the aggregate behavior of drivers. Given a set of telematics data, we start by segmenting trajectories to identify meaningful driving patterns (e.g., a hard brake or a sharp turn). Then we analyze each pattern with respect to contextual data to identify cause-and-effect patterns of significance (e.g., *traffic signal* $\rightarrow$ *hard-braking event*). We propose a novel approach for trajectory segmentation, and a data-driven solution to explore cause-and-effect patterns.

The second solution for characterizing driving context is a *geo-spatiotemporal pattern discovery* framework based on contextual data such as traffic and weather

data. Existing frameworks to perform pattern discovery in spatiotemporal data rely on a simplistic definition of a spatiotemporal neighborhood, which demands spatial closeness and temporal overlap. However, geo-spatiotemporal data requires stricter metrics to define the neighborhood, which has an impact on methodology and framework design. To fulfill this requirement, we propose a new framework that explores two types of patterns in geo-spatiotemporal data, namely *propagation patterns* and *influential patterns*. Propagation patterns show common cascading patterns of traffic and weather entities (e.g., *rain* $\rightarrow$ *accident* $\rightarrow$ *congestion*). Influential patterns show the impact of long-term entities on their spatial neighborhood (e.g., *major construction* $\rightarrow$ *more congestion events*). Exploring such patterns derives insights about transportation infrastructures and common driving habits in different contexts, which we show through extensive experiments using large-scale, real-world data.

Characterizing driving style is about illustrating drivers' personalities and skills, by capturing variations in driving behavior that discriminate different drivers from each other. We propose a deep convolutional recurrent neural network (D-CRNN) model to derive useful driving style information from telematics data. The objective is to predict driver identity for a given trajectory by utilizing driving style information. In order to prevent *spatial memorization* – as a result of employing spatially-similar trajectories taken from a driver that could mislead the task and invalidate the results – we propose several algorithms to sample trajectories per each driver and build a fair evaluation framework. Through multiple testing scenarios based on real-world datasets, we show the effectiveness of our proposal in comparison to the state-of-the-art models.

Context-aware driving risk prediction can be achieved by *macro-* and *micro-level* solutions. Macro-level driving risk prediction is about predicting the possibility of *traffic accidents* (as natural indicators of driving risk) inside a region during a time interval. To accomplish this, we propose a novel deep-neural-network-based solution that uses contextual data (such as traffic, weather, characteristics of road-network, and temporal data) to efficiently perform the prediction task. Our solution relies on easy-to-obtain data; thus it is scalable to large-scale, real-world applications. In addition, the proposed framework is applicable to real-time accident prediction given its design components. Our extensive experiments based on real-world data illustrate the effectiveness of the framework for macro-level driving risk prediction.

Micro-level driving risk prediction is about utilizing telematics and contextual data to perform individual-level risk prediction. Existing solutions usually suffer from under-utilization of telematics and contextual data, inherent biases in modeling, and dependence on input of extensive telematics and contextual data. Thus, existing solutions are impractical for real-world, large-scale applications. To address these shortcomings, we propose a new framework that relies on telematics data collected in terms of GPS trails and contextual data in terms of characteristics of road-network. The main process of this framework can be summarized in three important steps: 1) contextualizing telematics data to better

represent driving behavior in different contexts; 2) building a risk cohort classifier using contextualized telematics data and weak risk labels; and 3) constructing a driving risk prediction process to be employed for real-world purposes. Our proposal provides an easy way to extend the use of heterogeneous sources of contextual data and employ additional techniques to represent telematics data. In addition, it offers the possibility of adjusting the number of risk cohorts based on the characteristics of different regions. Our analysis and results based on real-world data prove the effectiveness of our framework for the important task of micro-level driving risk prediction.

To my wife and my family

for their tremendous support and for motivating me to pursue my goals

Acknowledgments

Sometimes in your life, you must accept some challenges that not only make life more meaningful, but also help to better know and improve your strengths. Getting a Ph.D. is one of those challenges that makes you grow up and helps you to think differently. Unlike any other education level, during the Ph.D., you can think freely and be more innovative and creative, and that brings excitement. For the first time, you find this opportunity to be a voice of science and not just a listener. These are just a few reasons among many that I enjoyed the path that I chose five years ago and I feel it as an important accomplishment in my life.

I am thankful to many people who helped me on this journey. First and foremost, I want to thank my advisor, Rajiv Ramnath, for his invaluable support, kindness, and his trust in me and my decisions. Rajiv helped me to better understand industry-driven research, and how to define our goals and expectations. I found myself very lucky to work with him and enjoyed his support throughout my Ph.D. Next, I would like to thank my co-advisors, Professor Srinivasan Parthasarathy and Professor Arnab Nandi, who helped me to improve my critical thinking and analytical skills and who were always supportive, from the time when I was proposing new ideas to the time when I was writing papers and presenting my work. Last, I want to thank Professor Jayashree Ramanathan, who offered her support and trust and was the main reason that I was able to join the Ohio State University.

Because a Ph.D. program always starts with an interesting problem to solve, I express my gratitude to the industrial sponsor of my research, the Nationwide Mutual Insurance Company, who provided me a unique opportunity to conduct industry-driven research based on real-world problems and valuable data. I also want to thank my incredible managers and mentors at Nationwide Insurance, Bruce R. Craig, Colleen Saunders-Chukwu, James (Jim) Tyo, Todd Anello, and Tara Paider.

Being an international student, you always feel apart from your family and loved ones, and this feeling makes it difficult to live or study. However, being married to a great person

substantially helped me to fill the emotional gap and stay focused and motivated. I thank my wife, Setareh, for her tremendous support and unconditional love. I would also like to thank my family in Iran, who were always supportive and encouraged me to pursue my goals and dreams. I definitely could not have come this far without their genuine support.

Finally, I thank my great colleagues, lab mates, and friends at Ohio State, who helped me to expedite the work and expand discoveries. Among them, I would like especially to thank Mohammad Hossein Samavatian, Behrooz Omidvar-Tehrani, Jiankai Sun, Pravar Mahajan, and Kayhan Moharreri.

Vita

2019 ..... Ph.D.
Computer Science and Engineering,
The Ohio State University, USA

2012 ..... M.S.
Computer Software Engineering,
University of Tehran, Iran

2009 ..... B.S.
Computer Science,
Shahid Beheshti University, Iran

Publications

Research Publications

Sobhan Moosavi, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, Rajiv Ramnath "Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights". *In the 27th International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2019)*, (Chicago, IL), 2019

Sobhan Moosavi, Mohammad Hossein Samavatian, Arnab Nandi, Srinivasan Parthasarathy, Rajiv Ramnath "Short and Long-term Pattern Discovery Over Large-Scale Geo-Spatiotemporal Data". *In the 25th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 2019)*, (Anchorage, AK), 2019

Jiankai Sun, Sobhan Moosavi, Rajiv Ramnath, and Srinivasan Parthasarathy "QDEE: Question Difficulty and Expertise Estimation in Community Question Answering Sites". *In the 12th International Conference on Web and Social Media (AAAI ICWSM 2018)*, (Stanford, CA), 2018

Sobhan Moosavi, Behrooz Omidvar-Tehrani, Rajiv Ramnath "Trajectory Annotation by Discovering Driving Patterns". *In the 3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics (UrbanGIS'17)*, (Los Angeles, CA), 2017

Sobhan Moosavi, Behrooz Omidvar-Tehrani, R. Bruce Craig, Arnab Nandi, Rajiv Ramnath "Characterizing driving context from driver behavior". *In the 25th International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2019)*, (Los Angeles, CA), 2017

Sobhan Moosavi, Arnab Nandi, Rajiv Ramnath "Discovery of driving patterns by trajectory segmentation". *In the 3rd ACM SIGSPATIAL PhD Symposium*, (San Francisco, CA), 2016

Fields of Study

Major Field: Computer Science and Engineering

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Telematics data refer to information collected from drivers in real time using a variety of sensors installed on vehicles to profile driving behavior. Contextual data, in this dissertation, refers to any data that provide *sufficient context* to better analyze driving behavior. Examples of contextual data are *traffic data*, in the form of raw speed observations or incidents reports, collected using sensors installed throughout the road network or reported by traffic authorities; *weather data*, as reported from weather stations using a variety of sensors; *road-network characteristics*, in terms of properties of the road network (e.g., road type and road shape), points of interest, and other available attributes; and *temporal information*. Analysis of telematics alongside contextual data provides valuable insights regarding an individual's behavior, common driving habits, and characteristics of the road network with regard to dynamic traffic flow.

Analysis of driving behavior has seen an enduring interest over the past few decades, with the important goals being driving risk prediction [3, 6, 8, 10, 11, 12, 57, 69, 74, 121, 125, 129, 133, 134, 135, 137, 139, 141], driving accident prediction [15, 21, 30, 82, 92, 114, 119, 120, 132, 136], driver coaching to improve driving skills [31, 69], and understanding traffic phenomena to be used for urban planning and traffic control and management [16, 21, 67, 103, 112, 143].

The primary focus in this dissertation is on *driving behavior analysis and risk prediction* to be used for applications such as *usage-based insurance* and urban planning. Usage-based insurance is about adjusting the insurance policy rates based on the behavior of drivers, by utilizing telematics and contextual data [144, 154]. Generally, usage-based insurance programs are beneficial for both drivers and insurance companies. Drivers can benefit from

improving their skills and achieving lower insurance rates. Insurance companies can benefit by adjusting the policy rates based on *riskiness* of drivers and by prediction and prevention of insurance claims [144, 154]. Therefore, the important research question here is: *How do we employ and process telematics and contextual data to predict driving risk?* To answer this question, we need to consider the following concerns:

- Driving risk scoring is not a **context-agnostic** process, which means the driver's behavior cannot be studied in isolation, regardless of properties of context and behavior of other drivers in the same context. As an example, a risky driver in city C_1 may be a safe driver in city C_2, regarding the behavior of other drivers, traffic, weather, and road-network characteristics, when comparing these cities.

- Driving risk depends on the **personality of drivers** and their **driving skills**, where such properties are context-agnostic. Thus, a reasonable prediction of driving risk needs to employ such information in order to make satisfactory predictions.

Given these concerns, we can rephrase the original question as: *How do we use and model telematics and contextual data to explore properties of driving contexts as well as drivers' personalities and skills, to perform driving risk prediction?*

Existing solutions for driver behavior analysis and driving risk prediction suffer primarily from failure to properly utilize telematics data [129, 133, 137, 141] and contextual data [57, 69, 74, 121, 125, 134, 135, 139], inherent biases in modeling that invalidate prediction results [134], and reliance on extensive sources of telematics and/or contextual data that makes the solution impractical for large-scale and real-world applications [57, 69, 139].

To address the shortcomings of the existing studies and to answer the primary research question, we propose several techniques with respect to different aspects of the question. First, we use various sources of telematics and contextual data to explore characteristics of driving contexts, using two different approaches: *segmentation and causality analysis* and *geo-spatiotemporal pattern discovery*. Then, we propose a *deep-neural-network-based* solution using telematics data to capture variations in driving behavior to learn about drivers' personalities and skills. Next, we propose two solutions for driving risk prediction. The first solution is a *macro-level* driving risk prediction framework that predicts the possibility of traffic accidents inside a region on a real time basis, with respect to contextual information.

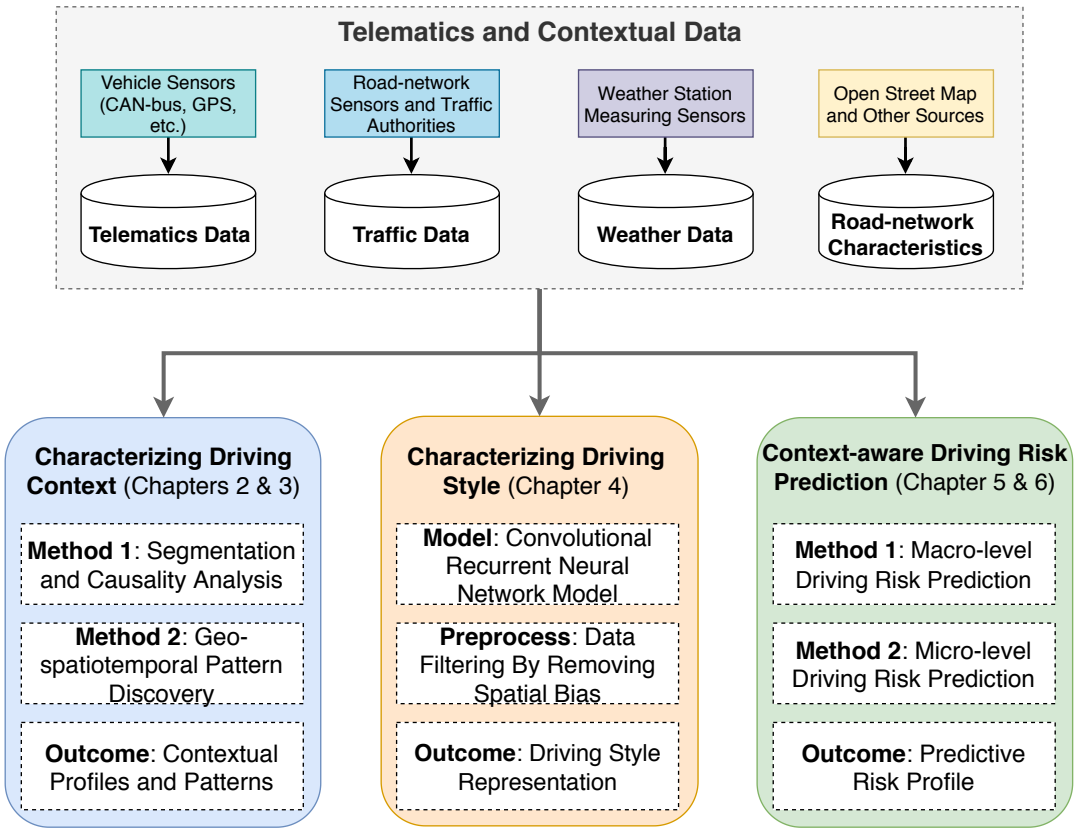


Figure 1.1: Using various sources of telematics and contextual data, in this dissertation, we seek to properly answer the following research question: *How do we use and model telematics and contextual data to predict driving risk?* To answer this question, we propose a solution that consists of three parts: 1) characterizing driving context to explore properties of different regions based on telematics and contextual data; 2) characterizing driving style to capture variations in driving behavior that demonstrate drivers' personalities and skills; and 3) context-aware driving risk prediction using macro- and micro-level approaches. The first two parts derive useful insights that we leverage to develop the third part.

The second solution is a *micro-level* approach that provides individual-level driving risk prediction based on telematics and contextual data. The big picture of this dissertation is presented in Figure 1.1. Each of the three parts applies a different process to extract insights and provide the prerequisites for the primary goal of this dissertation, that is, "driving risk prediction."

1.2 Thesis Statement

In this dissertation, we develop several approaches to employ and process telematics and contextual data for the important task of "driving risk prediction". We seek to answer the following questions: *How do we employ telematics and contextual data to characterize the properties of different regions? How do we derive driving style information based on telematics data? How do we employ contextual and telematics data to build an effective driving risk prediction model?* We proposed several techniques to answer these questions, and Figure 1.1 shows the big picture of this dissertation.

1.3 Research Contributions

The main research contributions of this dissertation are as follows:

- **Characterizing Driving Context by Trajectory Segmentation and Causality Analysis**: We propose a framework to characterize driving context using telematics and contextual data. In this framework, we employ a novel *trajectory segmentation* approach that serves to extract meaningful patterns (e.g., speed-up, turn, and hard brake) from trajectories. The important steps of our segmentation approach are: 1) building a probabilistic Markov model from telematics data to represent *aggregate behavior of drivers*; 2) transforming each trajectory to a time-series representation using the Markov model to represent the *unlikelihood* of the behavior of a driver; and 3) applying a dynamic-programming-based time-series segmentation algorithm to identify *coherent patterns* or as we term "segments." Then, we study the relationship between each segment (or meaningful part of a trajectory) and extrinsic sources of contextual data (e.g., traffic, road characteristics, and weather events) to explore any meaningful relationship. In other words, we perform causality analysis to explore probable *causes* for a given pattern in a trajectory (e.g., *traffic signal* $\rightarrow$ *observation of hard brake*). We note that a significant number of driving patterns are determined by extrinsic causes. Deriving such insights for different contexts is a practical approach to characterize them.

- **Characterizing Driving Context by Geo-Spatiotemporal Pattern Discovery**: We propose a geo-spatiotemporal pattern discovery framework to extract *cause-and-effect* patterns in geo-spatiotemporal data such as traffic and weather data. Using

this framework, we seek to explore two types of patterns: *propagation* and *influential*. Propagation patterns reveal the common cascading forms of geo-spatiotemporal entities in a region (e.g., *rain* $\rightarrow$ *accident* $\rightarrow$ *congestion*). Influential patterns demonstrate the impact of long-term geo-spatiotemporal entities on their spatial neighborhood within a time frame (e.g., *major construction* $\rightarrow$ *more congestions*). We propose a *short-term pattern discovery* process to extract propagation patterns, and a *long-term pattern discovery* process to identify influential patterns. Such patterns provide macro-level insights about prevalent driving habits and characteristics of transportation infrastructure for different regions or contexts.

- **Characterizing Driving Style**: Driving style includes unique characteristics of driving behavior for an individual when compared to other drivers in the same context. We propose a novel deep-neural-network-based solution to capture such characteristics from telematics data. Our proposal is a *deep convolutional recurrent neural network model* (termed D-CRNN for short), that uses numerical matrix representation of a trajectory as input and predicts the identity of its driver. By capturing useful driving style information based on extensive experiments using real-world datasets, our proposal achieves satisfactory results to predict the identities of drivers. We mitigate the impact of spatial similarity between trajectories taken from the same driver to avoid any potential bias in modeling and evaluation (which we discovered to be an important challenge for this task).

- **Macro-level Driving Risk Prediction**: Traffic accidents are significant indicators of driving risk. We propose a new solution to predict driving accidents using contextual data such as traffic, weather, points-of-interest, and temporal information. Our solution is a novel deep-neural-network model that uses heterogeneous sources of contextual data and predicts the possibility of traffic accidents for a region during a fine-grained time interval. This is a macro-level driving risk prediction framework because the model offers region-based prediction rather than individual-level prediction. We show the usefulness of our proposal via real-world experiments based on large sets of heterogeneous contextual data.

- **Micro-level Driving Risk Prediction**: Individual-level or micro-level driving risk prediction based on telematics data is a rather new topic that has gained attention in the

past few years. We propose a new solution for driving risk prediction by utilizing telematics and contextual data. Our solution comprises the following steps: 1) contextualizing telematics data by effectively representing trajectories to encode essential driving and contextual data, 2) building a risk cohort classifier using contextualized telematics data and weak risk labels, and 3) deriving a risk prediction process based on contextualized telematics data to be employed for real-world applications. Our framework can be extended by employing original methods of contextualizing telematics data through the use of a) new sources of contextual data or b) new forms of representation of telematics data. Further, the framework allows the definition of as many risk cohorts as needed for a given region, which makes it adaptable to different regions with different contextual properties. Our experiments and results based on real-world data prove its applicability for the important task of driving risk prediction.

1.4 Outline

The dissertation is organized as follows. In Chapter 2, we present our solution for characterizing driving contexts by segmentation and causality analysis using telematics and contextual data. Chapter 3 describes another framework for characterizing driving contexts using geo-spatiotemporal pattern discovery based on contextual data. Chapter 4 illustrates our solution for characterizing driving style using telematics data. The macro-level driving risk prediction using contextual data is presented in Chapter 5 and Chapter 6 provides the details on micro-level driving risk prediction based on telematics and contextual data. Finally, Chapter 7 concludes this dissertation by summarizing the important conclusions and directions for future research.

Chapter 2: Characterizing Context by Segmentation and Causality Analysis

In this chapter, we propose a data-driven framework to characterize driving context, where context can be illustrated by a combination of *location* and *time*. As input, this framework uses large-scale telematics data and applies a novel *trajectory segmentation* approach to identify meaningful driving patterns (e.g., making a turn, braking, and acceleration). Then, using a *causality analysis* approach and contextual data (e.g., data on traffic and properties of routes), the framework seeks to relate each meaningful driving pattern to at least one contextual cause (e.g., stop sign $\rightarrow$ hard brake), whenever it is possible. The results of this framework form a profile for each context, which later can be used to evaluate the behavior of un-seen drivers for applications such as usage-based insurance, driving risk prediction, and driver coaching.

2.1 Motivation

The amount and availability of telematics data has increased drastically thanks to the ubiquity of sensors in various applications and high-capacity data centers that can store and make available these data. Examples of telematics data are the New York taxi cab [146], Porto cab [63], and GeoLife [47]. The availability of these large telematics data sources has led to the development of analytic applications to gain insights from these data. The discovery of characteristics of *driving context* is a new area that we introduce in this chapter. Driving context can be described as a combination of *location* (e.g., Interstate-90) and *time* (e.g., weekdays between 3pm to 7pm). We define a *characteristic* for a context as the *correlation* between a driving pattern (e.g., hard braking) and an environmental effect (e.g., traffic congestion). Thus, characteristics of a context can relate meaningful driving patterns to environmental effects. By having information about characteristics of different

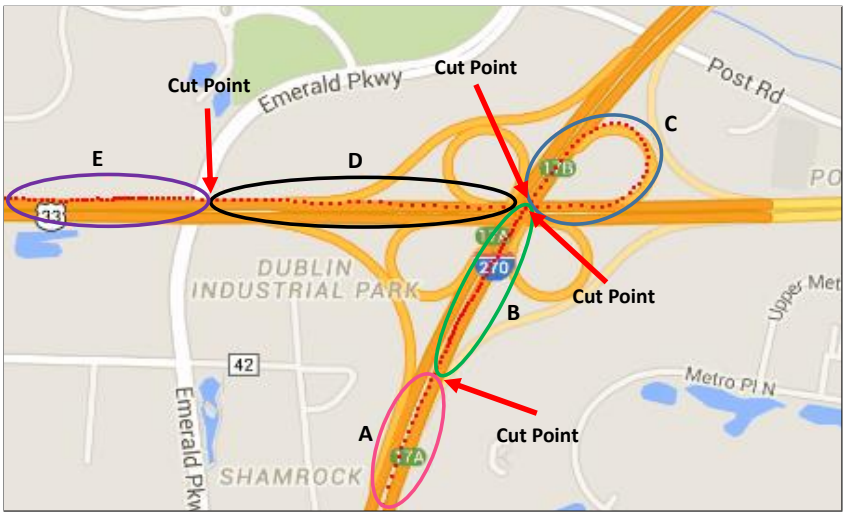


Figure 2.1: A sample trajectory with several driving patterns specified by ovals. Red arrows show the points of transition between patterns. Each pattern illustrate a homogeneous part of the trajectory and existence of each pattern is correlated to some causes (e.g., traffic congestion).

driving contexts, one can validate hypotheses about the behavior of an individual driver within a context and provide feedback to drivers in order to help them to improve their skills. Analysis of driving behavior is related to *usage-based insurance* (see Example 2.1.1) and feedback o drivers is known as *driver coaching* [31].

In this chapter, we address the problem of *discovering driving context* by exploring characteristics for a given *context*, based on the behavior of drivers and using complementary sources of spatio-temporal data to analyze behavior. We define the behavior of a driver in terms of meaningful *driving patterns*. Moreover, we seek to explore *causes* that underlie a specific pattern within a context by conducting analysis across several spatio-temporal data sources (e.g., traffic data, data on the features of roads, etc.). This process will shape our *framework* and allow us to identify the characteristics for a context as we demonstrate later in this chapter. Figure 2.1 shows an example of a trajectory, where red dots show the location of the vehicle over time. The trajectory begins at the bottom center and continues to the left after a clockwise turn. Different parts of the trajectory exhibit different driving patterns, marked by ovals. For instance, oval B shows *slow down*, oval C shows a *loop (ramp)*, and oval D is a *speeding-up*.

A driving pattern shows the consistent behavior of a driver within a sub-trajectory. The cause for a transition between patterns, hence introducing a new pattern, can be *extrinsic* (e.g., an accident, a traffic signal, and traffic congestion) or *intrinsic* (e.g., driver distraction and personality of the driver). For example, in Figure 2.1, a transition occurs in the middle of the highway, where a *slow down* is happening (oval E). The cause for this change may be "traffic congestion." The focus of this study is on extrinsic causes.

The problem of discovery of driving context is complex because we have to deal with the following challenges. First, we work on a large-scale dataset obtained by observing only externally visible phenomena (e.g., vehicle's speed) with no additional intrusive monitoring. This is in contrast to studies [13, 14, 35] in which data were collected using a fully monitored environment (for example, with cameras placed inside the car monitoring driver's every move and expression) and a small set of drivers and routes. Second, even if we could comprehensively monitor drivers, routes, and vehicles, relating a pattern to *extrinsic* causes can be difficult because intrinsic causes can also explain specific pattern transitions [7]. Thus, finding a valid set of patterns and also the exact set of extrinsic causes that underlie each pattern, which are the bases of deriving characteristics for a context, are two challenging tasks, worthy of our study. Example 2.1.1 illustrates a potential use of our results.

Example 2.1.1 *Regarding usage-based insurance, an insurance company provides a personalized insurance policy for a customer, based on his/her driving history [154, 144]. For this purpose, the insurance company needs to compare the behavior of a driver to a reference population of drivers to find out how "risky" or "safe" the driver is. Assume Mark is a new customer and his driving history reveals that his driving behavior, within a context C, shows 20% more* hard braking *and* hard-acceleration *patterns compared to a reference population of drivers in the same context. Hence, Mark appears to show abnormal behavior, and his driving may be characterized as risky.*

In this chapter, we introduce DriveContext, a framework to efficiently discover the characteristics of driving contexts. This framework consists of two major components, dSegment and dDescribe. The first component of our framework, dSegment, applies a behavior-based trajectory segmentation algorithm to find meaningful driving patterns within a trajectory. The second component of our framework, dDescribe, then reveals the extrinsic causes for each driving pattern. To evaluate the first component and compare it

with state-of-the-art segmentation approaches, we use a dataset of annotated car trajectories (DACT), which includes a set of trajectories with their segments specified by experts [113]. We then apply DriveContext on a real-world dataset of car trajectories to illustrate how to derive interesting characteristics for different contexts.

The main contributions of this chapter are summarized as follows:

- We propose a novel trajectory segmentation approach, dSegment, to find meaningful driving patterns based on the behavior of drivers.
- We propose a novel use of spatio-temporal data sources, dDescribe, to explore causes for a driving pattern.
- We leverage the causality analysis results, conducted for a set of driving patterns within a context, to explore the characteristics of that context.

2.2 Preliminaries and Problem Statement

Assume we are given a telematics database $\mathcal{D}$ of the form $\langle \mathcal{V}, \mathcal{T} \rangle$ where $\mathcal{V}$ and $\mathcal{T}$ are the set of vehicles and trajectories, respectively. Each trajectory $T \in \mathcal{T}$ is a sequence of $|T|$ data points $\langle \rho_1, \rho_2, \ldots, \rho_{|T|} \rangle$. Each data point ρ is a tuple of the form $\{t, lat, lng, s, a, h\}$ which captures a vehicle's status at time t as its latitude and longitude are $\langle lat, lng \rangle$, with speed s (km/h), acceleration a (m/s^2), and heading h (degrees). Time is considered to be measured in seconds. Also, the heading is the direction of the moving vehicle, described by a degree-value between 0 and 359, where 0 means the north.

We study the "discovery of driving context" in terms of two sub-problems: *Segmentation* and *Causality Analysis*. A segmentation of a trajectory T into n segments, denoted as seg_T, is a set of cutting indexes $seg_T = \langle I_1, I_2 \ldots, I_n \rangle$ that mark the end points of the segments within a trajectory. Thus, we can define a set of cutting data points for the segmented trajectory T as $\langle p_{I_1}, p_{I_2} \ldots, p_{I_n} \rangle$. Note that $p_{I_n} = \rho_n$ (since segments are specified by their last data point, the last cutting point is the last data point of T). All data points between indexes I_{i-1} and I_i, excluding point $\rho_{I_{i-1}}$ and including point ρ_{I_i}, belong to the i^{th} segment. Note that segments are non-overlapping. Each segment represents a *driving pattern* and each cutting point $p_{I_i}, I_i \in seg_T$, represents a *transition between patterns*. Figure 2.1 shows several segments (by ovals) and cutting points (by arrows). Considering the segmentation task as

an optimization problem, we define the optimization goals as (i). maximizing homogeneity within segments, and (ii). minimizing the number of extracted segments.

The existence of a segment is potentially relevant to extrinsic or intrinsic causes. In this work, the focus is on extrinsic causes that we refer to as *events*. We keep track of events in an event database $\mathcal{E}$ of the form $e = \langle t, lat, lng, type \rangle$, where each event $e \in \mathcal{E}$ occurs in time t, in a geographical area whose center is $\langle lat, lng \rangle$ of type *type*. An event can be of any of types including *Physical Fact* (e.g., traffic light in a road), *Physical-Temporal Event* (e.g., traffic congestion), or *Temporal Event* (e.g., tornado). Given the set of cutting points $\langle p_{I_1}, p_{I_2} \ldots, p_{I_n} \rangle$, identified as result of segmenting trajectory T, and the database $\mathcal{E}$ of events, the second sub-problem (i.e., causality analysis) is one of finding if, and to what extent, each cutting point p_{I_i}, $1 \leq i \leq n$, is related to (or caused by) an event $e \in \mathcal{E}$.

2.3 Related Work

To the best of our knowledge, no previous research has proposed a similar framework for the discovery of the driving context. However, our work does relate to the research of a number of others, specifically research in *trajectory segmentation* (as used in dSegment) and *making sense of trajectories* (as discussed in dDescribe). Also, our work is related to *driving pattern discovery*. A review of related work is presented next.

2.3.1 Trajectory Segmentation

The task of segmentation has been addressed in the literature in several studies such as [24, 38, 56, 65]. In [38], a greedy segmentation algorithm exploits a set of monotonic spatio-temporal criteria (e.g., defining relative thresholds for some feature values) on features like speed, heading, etc. Alewijnse et al. extended this previous work to both monotonic and non-monotonic criteria [65]. However, criteria-based methods need human input for tuning parameters. Moreover, they are *context-agnostic* in the sense that they only consider the input trajectory and not the whole dataset. Therefore, the optimization process is a local one, where we propose a global optimization for segmentation. Similar to dSegment, where we propose a *context-aware* approach by building a Markov Model, Alewijnse et al. [64] present a solution that builds a Brownian Bridge model and uses a dynamic programming approach to capture the best set of segments of animal movements. While

dSegment bears some similarities with [64], it exploits a normal distribution model instead, which we find that more suitable for car transportation data. Transforming trajectory prior to segmentation is also previously discussed by [52], however, their transformation is a local approach, based on comparing line segments of input trajectory. Instead, we perform a global, likelihood-based transformation to provide a segmentation where the extracted segments represent meaningful driving patterns. Essentially, dSegment is a global optimization-based segmentation approach that builds up a model on the entire dataset. Note also that there is no need for human intervention in dSegment as in [38, 65].

2.3.2 Making Sense of Trajectories

Similar to dDescribe, there are some other approaches that try to make sense of driving data and to explore insights encapsulated in trajectories. Among these approaches, we can point to discovery of transportation mode [46], map matching [36, 48], points of interest (POI) discovery [62, 101], and providing descriptive summary for trajectories [87]. Besides, Wu et al. [106] recently proposed to predict traffic based on some external data sources including POI data, collision data, weather data, and geotagged tweet data. This is similar to our analysis in terms of dDescribe, where we try to find a correlation between driving patterns and traffic congestions and physical properties of routes. The latter one is, in some sense, similar to POI. However, we pursue a different goal which is the identification of characteristics of a context.

2.3.3 Driving Pattern Discovery

Discovery of driving patterns (e.g., make a turn, change/keep lane, etc.) has been prominently studied in the literature [35, 25, 19]. In [13], a fully monitored test environment is elaborated where a small set of drivers are provided with instructions in order to measure several feature values. Then, a Hidden Markov Model (HMM) is applied to predict specific driving patterns. Driver eye movement is analyzed in [14] as an additional feature to predict driving patterns. However, all these works exploit a fully monitored context, which is costly and nearly infeasible on large-scale or to be used for Usage-Based Insurance. On the other hand, some computational based approaches proposed in the literature, like [86], which proposes a time-series matching solution to discover recurring driving patterns. While the application of this approach on large-scale data is straight-forward, there is no guarantee

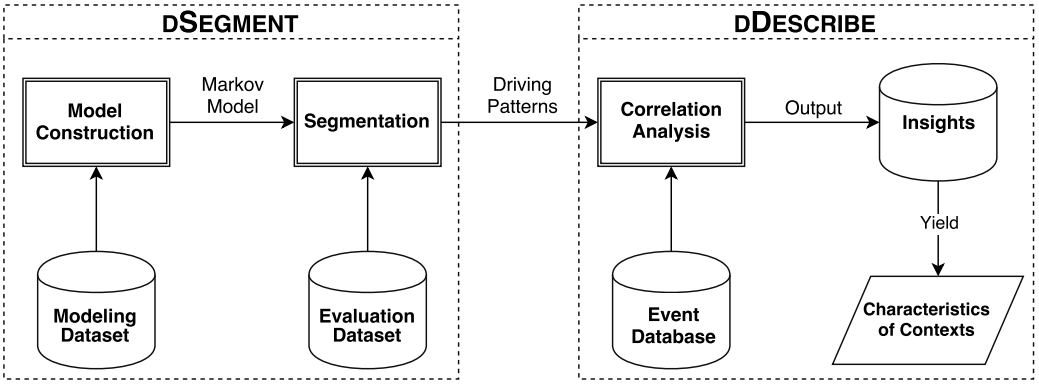


Figure 2.2: The overall process of DRIVECONTEXT framework which consists of two components, the DSEGMENT which includes the *model construction* and the *segmentation approach*, and the DDESCRIBE which includes the *correlation analysis* based on extracted patterns that yields insights in terms of *characteristics of contexts*.

to find *meaningful* patterns. On the contrary, DRIVECONTEXT is developed based on externally observable features that are rather easy to collect. Hence, our framework is applicable on large-scale. Also, in comparison to related work, more chances of finding meaningful patterns are available.

2.4 The DriveContext Framework

In this section, we present the novel framework DRIVECONTEXT to discover characteristics of different driving contexts. Figure 2.2 depicts the overall process of DRIVECONTEXT where it consists of two major components, which are DSEGMENT and DDESCRIBE. Details on these components are provided in the following sub-sections.

2.4.1 dSegment Component

DSEGMENT is a novel approach to wisely partition a trajectory based on the behavior of the driver, such that each resulting segment corresponds to a meaningful driving pattern (e.g., turn, speed-up, etc.). Based on Figure 2.2, DSEGMENT consists of two parts, *Model Construction* and *Segmentation*. The first part, "Model Construction", includes *Dataset Preprocessing* and *Markov Model Creation* . The second part, "Segmentation", comprises

Trajectory Transformation and *Trajectory Segmentation*. We describe each of aforementioned sub-parts as follows.

Dataset Preprocessing

Regarding the description of the data model in Section 2.2, the dataset is a collection of trajectories, where each trajectory is a sequence of data points. The preprocessing of dataset consists of the following steps: 1) Removing data points with missing or noisy (out of range) GPS records, 2) Rounding the values of *Acceleration* to be divisible by 0.25, and 3) Using *Change of Heading* instead of absolute heading value. Steps 2 and 3 help to simplify the Markov Model by reducing the possible number of states, where there will be no significant effect on the generalization of the model. Moreover, the step 3, which is an empirical decision, helps to reflect the change of heading more clearly. For example, the difference between 0 and 359 is 1, while we cannot directly model such difference using absolute heading values.

Markov Model Creation

The goal is to model the behavior of drivers in terms of a finite state machine that provides the probability of transition from one *driving state* to another one. In this way, we build a memory-less Markov model $M = \{\Phi, \Delta, \Pi\}$, where Φ is the set of states, Δ is the set of transition between states (along with the frequency of each transition), and Π is the set of probabilities of transition between states. We use the following principles to create M:

- State: We define a state $\phi \in \Phi$ as $\phi = \langle s, a, h \rangle$, where s, a, and h are speed, acceleration, and change of heading, respectively.

- Transition: Given a trajectory $T = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$, for each pair of consecutive data points ρ_i and ρ_{i+1} of γ, $1 \leq i < n$, we create two states $\phi_i = \langle s_i, a_i, h_i \rangle$ and $\phi_{i+1} = \langle s_{i+1}, a_{i+1}, h_{i+1} \rangle$ for ρ_i and ρ_{i+1}, respectively. We denote a transition from state ϕ_i to ϕ_{i+1} as $\phi_i \rightarrow \phi_{i+1}$. If Δ doesn't contain transition $\phi_i \rightarrow \phi_{i+1}$, then we add $\langle \phi_i \rightarrow \phi_{i+1}, 1 \rangle$ to Δ. Otherwise, we increase the frequency of transition $\phi_i \rightarrow \phi_{i+1}$ by 1.

- Probability of Transition: For a state ϕ, let us assume there is a $\delta \subseteq \Delta$, where $\delta = \{\langle \phi \rightarrow \phi_1, n_1 \rangle, \ldots, \langle \phi \rightarrow \phi_k, n_k \rangle\}$ and n_i is the number of observed transitions from ϕ to

ϕ_i in the training (modeling) dataset. Then, we update Π by inserting the probability of each transition $\phi \rightarrow \phi_i$, $1 \leq i \leq k$, using Equation 2.1:

$$prob_{\phi \rightarrow \phi_i} = \frac{n_i}{\sum_{j=1}^{k} n_j} \tag{2.1}$$

By using above principles, we may end up with a sparse Markov model. For example, we may not observe a transition from $\phi_1 = \langle 25, -2, 0 \rangle$ to $\phi_2 = \langle 24, -2, 0 \rangle$, although such a transition is quite likely to happen. In order to deal with this shortcoming of a basic Markov model and also to avoid the overfitting problem that the model construction is purely based on the training (modeling) trajectories, we need a further processing step known as *Regularization*. To do this, we adapt an existing, intuitive regularization approach known as *Wedding Cake* technique [27]. Assume we have a state $\phi = \langle s, a, h \rangle$ which has transition to a set of states $\bar{\Phi} = \{\langle s_1,a_1,h_1 \rangle, \langle s_2,a_2,h_2 \rangle, \ldots, \langle s_n,a_n,h_n \rangle\}$. Also, consider values S_{th}, A_{th}, and H_{th} as thresholds on speed, acceleration, and heading, respectively, to define regularization intervals. We use Algorithm 1 to regularize the Markov Model.

Algorithm 1: Wedding Cake Regularization [27]

Input: ϕ, $\bar{\Phi}$, S_{th}, A_{th}, H_{th}

for $sp = (\phi.s - S_{th})$ ***to*** $(\phi.s + S_{th})$ **do**
  for $ac = (\phi.a - A_{th})$ ***to*** $(\phi.a + A_{th})$ **do**
    for $hd = (\phi.h - H_{th})$ ***to*** $(\phi.h + H_{th})$ **do**
      ▷ Expanding state ϕ
      for $\phi' \in \bar{\Phi}$ **do**
        $\phi'' \leftarrow \langle (\phi.s + sp), (\phi.a + ac), (\phi.h + hd) \rangle$
        $prob_{\phi'' \rightarrow \phi'} \mathrel{+}= \frac{prob_{\phi \rightarrow \phi'}}{Euclidean(\phi, \phi'')}$
      end
      ▷ Expanding states in $\bar{\Phi}$
      for $\phi' \in \bar{\Phi}$ **do**
        $\phi'' \leftarrow \langle (\phi'.s + sp), (\phi'.a + ac), (\phi'.h + hd) \rangle$
        $prob_{\phi \rightarrow \phi''} \mathrel{+}= \frac{prob_{\phi \rightarrow \phi'}}{Euclidean(\phi', \phi'')}$
      end
    end
  end
end

Output: Regularized Markov Model

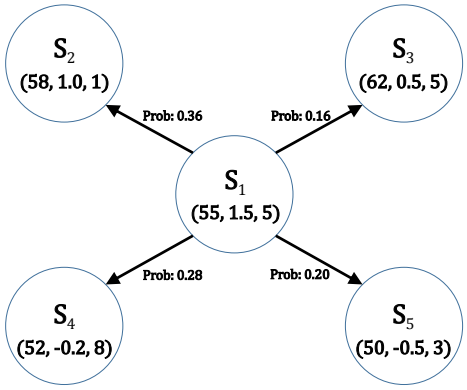


Figure 2.3: A portion of a sample Markov Model. Each state consists of a triple of Speed (km/h), Acceleration (m^2/s), and Change of Heading. Probability of transition is shown on each edge.

In Algorithm 1, a *Euclidean* function calculates the Euclidean distance between two states. Prior to using this distance measure, we normalize the value of all features to lie between 0 and 1 (i.e., min-max normalization). The idea of regularization is intuitive, where we first try to expand the set of states of the basic Markov model (lines 6 and 11), and then update the probability values of the transitions for the updated states (lines 7 and 12). Expansion of states simply means creating new states, if they do not exist already, by updating the feature values of the initial state (e.g., ϕ in line 6), using different thresholds. Moreover, the update of probabilities is about assigning the probability of transition to newly created states (or update the probability of existing ones), as a fraction of transition probability between initial states (e.g., $prob_{\phi \to \phi'}$ in line 7). Also, note that we set the aforementioned thresholds during the experiments.

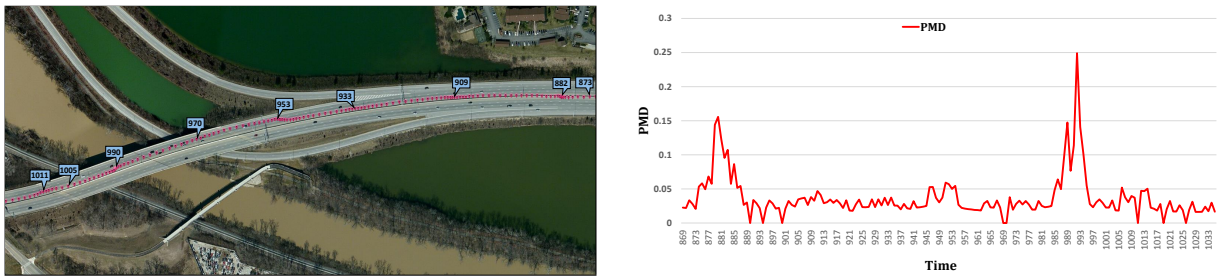
Our goal with the regularization is to reduce the gap between the state transition probabilities of the training (modeling) trajectories, and those of the test (evaluation) trajectories. The resulting regularized Markov Model is like a finite state machine that models the behavior of drivers in terms of probability of transitions between different driving states. For example, by having such a model, we may infer the speed change from 20 km/h to 50 km/h is not likely to happen. On the contrary, the change in speed by less than 5 km/h is quite likely. Figure 2.3 shows a sample Markov Model which contains transitions from S_1 to four other states in the model.

Trajectory Transformation

Recall that the aim of dSegment is to provide a segmentation of trajectories based on "behavior of drivers". To accomplish this goal and prior to segmentation, we apply a *transformation* on input trajectory to a *signal* in a new space which we call that Probabilistic Movement Dissimilarity (PMD) space. Given a trajectory $T = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ and a regularized Markov Model $M = \{\Phi, \Delta, \Pi\}$, we propose Algorithm 2 to map T to a signal S_T in PMD space. Given consecutive data points $\rho_i, \rho_{i+1} \in T$, Algorithm 2 first maps them to states ϕ and ϕ', respectively. Then, the algorithm calculates how *unlikely* the transition $\phi \rightarrow \phi'$ is, given the model M. In this algorithm, *ReturnState* returns a state corresponding to input data point ρ_i, and *ReturnProb* returns transition probability from ϕ to ϕ'. *TransitionFrom* returns a set of states R for an input state ϕ, such that $\{\phi \rightarrow r\} \in \Delta$, for $r \in R$. Also note that if ϕ and ϕ' represent the same state with zero acceleration, then the transition is quite likely. In other words, the unlikelihood of this specific kind of self transition is zero.

Algorithm 2: Trajectory Transformation

Input: γ, M
$S_\gamma \leftarrow \langle\rangle$
for $i = 1$ ***to*** n-1 **do**
  $\phi \leftarrow ReturnState(M, \rho_i)$
  $\phi' \leftarrow ReturnState(M, \rho_{i+1})$
  $v = 0$
  if $\phi \neq \phi'$ ***or*** $\phi.a \neq 0$ **then**
    $prob_{\phi \rightarrow \phi'} = ReturnProb(M, \phi, \phi')$
    $R \leftarrow TransitionFrom(M, \phi)$
    ▷ $R = \{r |\ (\phi \rightarrow r) \in \Delta\}$
    for $r \in R$ **do**
      $prob_{\phi \rightarrow r} = ReturnProb(M, \phi, r)$
      v += $Euclidean(\phi', r) \times prob_{\phi \rightarrow r}$
    end
    $v = \frac{v}{|R|}$
  end
  $S_\gamma \leftarrow Append(S_\gamma, v)$ ▷ Append v at the end of S_γ
end
Output: S_γ ▷ S_γ is transformed version (signal) of γ



(a) Sample Trajectory

(b) Signal in PMD space

Figure 2.4: The sample trajectory (a) and corresponding signal (b). Red dots on map show trip points, where the vehicle goes from right to left. Numbers in rectangular call-outs in (a) show time stamps which can be matched with time axis in (b). The PMD value shows the unlikelihood of driver's behavior.

Based on Algorithm 2, we map a trajectory into a signal in PMD space. The signal of a trajectory demonstrates the unlikelihood of driving behavior for each moment of the trajectory. An unlikelihood score is calculated based on the transition probabilities in the Markov Model M, which are a demonstration of the behavior of drivers in a population. Lines 7 to 14 in Algorithm 2 measure how far the observed transition $\phi \rightarrow \phi'$ is from the expectation, regarding the M. Figure 2.4a depicts a sample trajectory, where its corresponding signal in PMD space is represented in Figure 2.4b. The numbers in rectangular call-outs in Figure 2.4a show time stamps which can be mapped to *Time* axis in Figure 2.4b. The larger the PMD values, the more unlikely the behavior of the driver is. For instance, a large PMD value is observable for timestamp 990 in Figure 2.4b, where the actual trajectory in Figure 2.4a shows an unexpected reduction in speed and probably a lane change. The main takeaway from this sub-section is that we use a signal in PMD space as a representation of the behavior of a driver for a given trajectory.

Trajectory Segmentation

As we intend to identify the homogeneous parts of a trajectory as segments which are also representatives for driving patterns, we leverage an existing approach for segmentation of *electrical* signals, which is proposed by Han et al. [17], to find optimal segments of the signal of a trajectory. This approach is a dynamic programming algorithm that uses the Maximum Likelihood principle for segmenting one-dimensional signals. Given an input

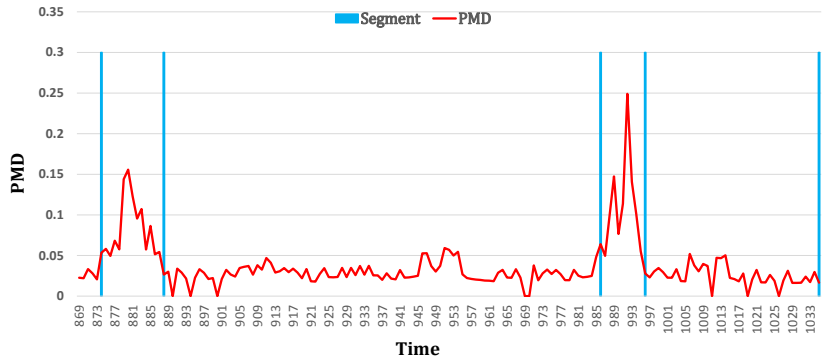


Figure 2.5: Segmentation of a sample trajectory by dSEGMENT, where the best number of segments is found as 5 by MDL.

signal $S = \langle x_1, x_2, \ldots, x_N \rangle$, the Maximum Likelihood (ML) of S can be defined by Equation 2.2.

$$ML(\theta; x_1, x_2, \ldots, x_N) = f(x_1, x_2, \ldots, x_N|\theta) = \prod_{i=1}^{N} f(x_i|\theta) \tag{2.2}$$

In Equation 2.2, θ is the set of parameters for a probability density function (PDF) f, which can be estimated based on data points of signal S. Similar to [17], we leverage the *Gaussian distribution* to find the parameters of the PDF f. Thus, $\theta = \langle \mu, \sigma \rangle$, where μ and σ are the sample mean and the standard deviation, respectively.

Recall that the goal of segmenting a trajectory T, so its corresponding signal $S_T = \langle x_1, x_2, \ldots, x_N \rangle$, is to find a set of cutting indexes $seg_T = \langle I_1, I_2 \ldots, I_n \rangle$ that maximize the likelihood within segments and minimize $n \leq N$, the best number of existing segments (see section 2.2). The recurrence relation to segment signal S_T is defined by Equation 2.3.

$$SSC(S_T, i, \nu) = \underset{i+5 \leq j \leq N}{\arg\max}(ML(\theta; x_i, \ldots, x_j) + SSC(S_T, j+1, \nu-1)) \tag{2.3}$$

In Equation 2.3, $SSC(S_T, i, \nu)$ gives the best Segmentation SCore (SSC) for a sub-sequence of signal S_T which starts at index i, with the goal being to find ν segments. Also, $ML(\theta; x_i, \ldots, x_j)$ gives the maximum likelihood score for sub-sequence $\langle x_i, x_{i+1}, \ldots, x_j \rangle$ of S_T. Note that we assume the minimum length of a segment to be five[1], this is why j starts from $(i+5)$. The initial call for Equation 2.3 is $SSC(S_T, 1, n)$. The interested reader may refer to [17] for more details.

[1]This values is found empirically and it will be described later in Section 2.5.

The last question in this sub-section is: *how do we find the best number of existing segments within a signal?* We use the Minimum Description Length (MDL) [2] for this purpose, which has been applied in [17] as well. MDL minimizes the Equation 2.4 for $n = 1, 2, \ldots, K$, where n is the number of segments and K is the upper bound on the number of segments (chosen by the user):

$$MDL(n) = -ln\prod_{i=1}^{n} f(x_{I_{i-1}+1}, x_{I_{i-1}+2}, \ldots, x_{I_i}, |\theta_i) + \frac{r_n}{2} lnN \tag{2.4}$$

In Equation 2.4, θ_i is the parameter set of the corresponding PDF, r_n is the number of estimated parameters (where n is the number of segments), and N is the length of the signal. We also set $I_0 = 0$. Figure 2.5 shows a part of a segmented signal which is related to the sample trajectory in Figure 2.4a. The blue lines in Figure 2.5 show the end points of segments (i.e., the cutting points). The best number of segments which has been found by MDL is 5. An interesting observation in Figure 2.5 is homogeneity of the behavior of driver *within* segments (patterns) and dissimilarity of patterns of behavior *between* neighboring segments, which is compatible with our optimization goals. As an example of a behavior-based driving pattern that is captured by dSegment, we point to the segment which starts at timestamp 986 in Figure 2.5. Matching it with the actual trip in Figure 2.4a, we observe that this segment is related to a driving pattern where the driver reduces speed and changes the lane. It is likely that these actions are due to traffic congestion. We describe how to discover underlying causes behind driving patterns in the next section. Also, more analysis on correctness of dSegment to extract valid segments in compare to the state-of-the-art is provided in Section 2.5.2.

2.4.2 dDescribe Component

Regarding Figure 2.2, the dDescribe is the second important component of DriveContext which analyzes extracted driving patterns (segments) to explore the underlying causes that make each pattern happen. This step helps to identify the characteristics for a given context as we describe later in Section 2.5.4. As we discussed in Section 2.2, the existence of a driving pattern is potentially related to extrinsic or intrinsic causes. In this chapter, our focus is to find the extrinsic causes, so-called *events*.

Recall that for a given trajectory T, dSegment returns a set of cutting points $\langle p_{I_1}, p_{I_2} \ldots, p_{I_n} \rangle$. Having a database of events $\mathcal{E}$ (Section 2.2) and a cutting point p_{I_i},

$1 \leq i \leq n$, the goal is to find whether p_{I_i} is *related* to an event $e \in \mathcal{E}$ or not. If we found that p_{I_i} is related to e, then this means the segment which starts at cutting index $(I_i + 1)$ is potentially caused by event e. Note that we accommodate the fact that a segment (pattern) can be caused by more than one event in some cases. We define the relevancy relationship between a cutting point p and an event e based on the type of the event. In this study, we consider three types of events: *physical fact*, *temporal-physical event*, and *temporal event*[2]. Following is how we measure relevancy for each type of the event:

- **Physical Fact**: An example is the presence of a traffic signal. In such a case, the relevancy can be measured as the distance between the locations of cutting point p and event e. We then say p and e are correlated if their locations are within a specified distance threshold.
- **Temporal-Physical Event**: An example is the existence of a traffic congestion in a specific place during a time interval. In this case, we say p and e are correlated if the two following conditions are satisfied: i. the time of the trajectory T, where $p \in T$, overlaps with the time interval of the event e, and ii. the distance between locations of p and e are lower than a threshold.
- **Temporal Event**: An example is having a severe storm within a specific time interval. In this case, we say p is correlated with e if the time of trajectory T, where $p \in T$, overlaps with the time interval of event e. The implicit assumption for this case is that we assume the event e is happened in the same region (city, state, etc.) as trajectory T is happened.

In this study, we leverage "physical facts" and "temporal-physical events" to build the event database $\mathcal{E}$ to be used as input of dDescribe component. More detail about creating the event database is provided in the next section.

2.5 Evaluation

In this section, we first describe the datasets which we used in this study. Then, we evaluate dSegment with respect to a ground-truth dataset. Next, we apply dSegment on a real-word dataset of car trajectories and conduct causality analysis using dDescribe. Finally, we provide examples of the applicability and usefulness of the DriveContext framework to demonstrate the implication of practice.

[2]There may be additional types of event, but we only consider the ones listed.

2.5.1 Dataset

We used four different sets of spatio-temporal data sources to build and evaluate components of DriveContext. These four datasets consists of (1) *Dataset of Annotated Car Trajectories (DACT)*, (2) *Nationwide Trajectories*, (3) *Physical Facts*, and (4) *Temporal-Physical Events*. We describe each dataset in the following subsections.

Dataset of Annotated Car Trajectories

The dataset of annotated car trajectories (or DACT [113]) provides a ground truth set to evaluate trajectory segmentation approaches. In this work, we leverage this set to compare dSegment with the state-of-the-art solutions. The DACT includes 50 trajectories which cover about 13.3 hours of driving data. The Easy-Aggregation set contains $1,372$ segments for 50 trajectories. Likewise, the Strict-Aggregation set contains $2,465$ segments for the same set of trajectories.

Nationwide Trajectories

In order to build and to show the applicability of DriveContext framework, we use a rich, real-world dataset of trajectories we term the "Nationwide Trajectories", provided by the Nationwide Mutual Insurance company based in Columbus, Ohio. To our knowledge, Nationwide Trajectories is one of the few large scale datasets with driving data for personal vehicles (as opposed to taxi cabs or other kinds of commercial vehicles). Further, this data is precise, having been collected by highly accurate devices connected to the On-Board Diagnostic (OBD-II) port of the vehicles, as well as rich, consisting of a variety of useful data items such as speed, acceleration, GPS coordinates, heading, etc. Finally, this dataset is highly granular, with data being collected at a consistent sampling rate as 1 second for all trajectories. The Nationwide Trajectories data was collected between July 2011 and January 2014 from 103 drivers, and contains 83,406 trajectories and covering about 20,689 hours of driving data.

We divided the Nationwide Trajectories into two sets: *modeling* and *evaluation*. We use the former to build the dSegment model, and the latter to evaluate the dDescribe, and also to show the application of DriveContext framework as an end-to-end solution. In order to build the evaluation set, we first sampled all the trajectories for 5 popular routes in the city (i.e., Columbus Ohio). Then, for each driver in the sampled data, we randomly

chose 40% of their trajectories to be used in the evaluation set. Thus, the modeling set contains 81,895 trajectories (20,073 hours of driving), produced by 103 drivers, and the evaluation set contains 1,421 trajectories (616 hours of driving), produced by 48 drivers. Also, it is worth mentioning that for each of five common routes in the evaluation set, we have the same *start* and *end* points (on map) for trajectories in the evaluation set. More details about the evaluation set is summarized in Table 2.2.

Physical Facts

Physical facts were drawn from two different sources of data, as follows:

i. Open Street Map (OSM)[3]: We used OSM as a publicly available source of annotations for different places all around the world. We only used a subset of the available annotations, specifically those related to physical facts, such as *exit/merge*, *ramp*, *bridge*, etc.

ii. Hand-Curated Annotations (HCA): Since OSM cannot be considered as a comprehensive source of annotations, we manually annotated routes in the evaluation set by using *Google Street View* and created a set of hand-curated annotations. Examples of annotations in this set include *sharp-turn* , *smooth-turn*, *exit/merge*, *intersection*, etc.

The set of physical facts contains 1,825 annotations from OSM and 95 HCA.

Temporal-Physical Events

One of the best examples of a temporal-physical event is *traffic congestion* which may be found in traffic congestion reports. However, since there is no publicly available historical sources of traffic congestion report which could be matched to our dataset, we used two different APIs, specifically, *Bing Traffic API* [4] and *Map Quest Traffic API* [5], to collect real-time traffic reports. A summary of the current dataset of congestion reports which covers the data for a period of one year, from February 2016 to February 2017, for the routes in the evaluation set is provided in Table 2.1. We present more details about how we use this traffic congestion data for causality analysis in Section 2.5.3.

[3] `www.openstreetmap.org`

[4] `https://msdn.microsoft.com/en-us/library/hh441725.aspx`

[5] `https://developer.mapquest.com/products/traffic`

Table 2.1: Summary of traffic congestion report dataset, collected using Bing and MapQuest APIs from Feb 2016 to Feb 2017

Route	#Bing congestion	#MapQuest congestion
Interstate-70	477	112
Interstate-71	401	1,369
Interstate-270	290	800
Interstate-670	365	1,189
315 Freeway	155	1,025

Table 2.2: Summary of evaluation set and segmentation outcome.

Route	Route Length	Number of Trajectories	Avg. Trajectory Length (secs)	Avg. Number of Segments
Interstate-70	6.7 km	535	333	4
Interstate-71	13.8 km	438	546	5
Interstate-270	12.3 km	195	444	3.3
Interstate-670	7.0 km	131	359	4.2
315 Freeway	14.6 km	120	703	8.9

2.5.2 dSegment Evaluation

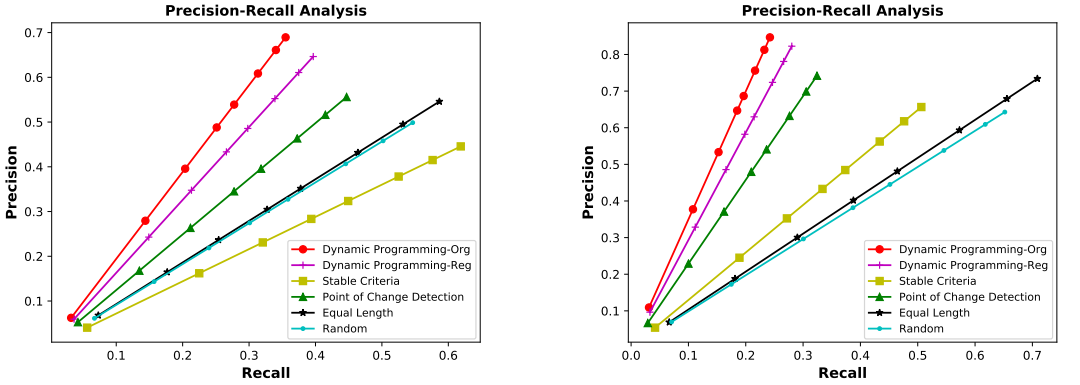
As shown in Figure 2.2, we first use the modeling set of Nationwide Trajectories to create the Markov model, setting S_{th}, A_{th}, and H_{th} in Algorithm 1 to 3, 0.5, and 6, respectively. The regularized Markov model consists of 47,495 states and about 5.8 million transitions between states. In order to evaluate our segmentation approach, we use the DACT annotation sets [113]. For comparison purposes, we use following four baselines:

- **Stable Criteria**: This approach is an alternative to dSegment, where a set of spatio-temporal heuristics are used for segmentation. An example heuristic is the *maximum* amount of the change of an input feature (e.g., speed) that can be allowed within a segment [38, 65].

- **Point of Change Detection**: As we transform a trajectory to a time series (PMD signal), instead of using the dynamic programming approach, one can use a *point of change detection* solution. Here we use a state-of-the-art approach proposed by Liu et al. [60] to first obtain the change score for each point of time series. Then, as authors

described in their paper, we empirically find a threshold on the change score to find peak points, which then define our segment borders (or cutting points).

- **Equal Length**: In this approach, we first assume all trajectories have the same number of segments, say η, and then we divide a trajectory to η equal size segments. Later in this section, we describe how to find η.
- **Random**: This approach is similar to the "Equal Length" approach, except we find segment borders at random, where the goal is to end up with η segments.



(a) Comparison based on Easy-Aggregation set (b) Comparison based on Strict-Aggregation set

Figure 2.6: Comparing different segmentation approaches based on DACT annotations

In order to find the upper bound on the number of existing segments, i.e., K (see section 2.4.1), we set $K = \frac{N}{5}$, where N is the length of the trajectory. The minimum length of a segment is assumed to be 5, which is also compatible with our findings in [113]. Since we have two sets of annotations for trajectories in DACT, we evaluate and compare our approach based on both sets. Also, we use *Precision* and *Recall* as evaluation metrics. Given a trajectory T with annotations $Ant_T = \langle a_1, a_2, \ldots, a_n \rangle$, if algorithm Alg finds cutting points $CP_{Alg} = \langle p_1, p_2, \ldots, p_m \rangle$ for T, then we use Equations 2.5 and 2.6 to define precision and recall, respectively.

$$Precision = \frac{Ant_T \cap CP_{Alg}}{m} \tag{2.5}$$

$$Recall = \frac{Ant_T \cap CP_{Alg}}{n} \tag{2.6}$$

We obtain the intersection between Ant_T and CP_{Alg} as follows: to find a match for $p_i \in CP_{Alg}$, we calculate its *Haversine* distance to all *available* annotations in Ant_T. If we find a pair (p_i, a_j), $a_j \in Ant_T$ for $1 \leq j \leq n$, such that their Haversine distance is lower than a predefined threshold, then we say there is a match for p_i. Once that we found such a_j, we no longer use that to match other cutting points in CP_{Alg}. We use values in set $\{0, 25, 50, 75, 100, 150, 200, 250\}$ as distance thresholds, where the measure is $meters$. Taking the aforementioned into account, Figures 2.6a and 2.6b show the comparison between different segmentation approaches based on two different sets of annotations in DACT, when varying the distance threshold. Note that we report the results of dSEGMENT by *Dynamic Programming-Org* and *Dynamic Programming-Reg*. The former refers to the case where we use the original Markov Model for segmentation, without regularization, while the latter refers to the regularized Markov Model. For Equal Length and Random approaches, we use $\eta = 30$ based on Easy-Aggregation and $\eta = 50$ based on Strict-Aggregation annotation sets. These numbers are set based on the average number of segments in a trajectory, as reported in [113]. Figures 2.6a and 2.6b show that dSEGMENT outperforms the other baselines by reasonable margins, based on both ground truth datasets. Moreover, using the original Markov Model leads to higher precision for prediction of segment borders, however, the regularized graph can improve the recall by loosing some precision points. This shows the regularization helps to generalize the model. Note that the choice of maximum distance threshold (i.e., 250 m) is based on the observation that precision and recall are not significantly changed by using larger thresholds. Also, given the average length of routes in evaluation set which is about 10 km (see Table 2.2), and the average number of segments for a trajectory based on DACT datasets which is about 40 [113][6], we have $\frac{10km}{40} = 250m$. Thus, using larger thresholds may invalidate the evaluation results.

Figure 2.7 depicts the frequency of extracted segments found by the different approaches. As one can see, dSEGMENT and the point-of-change-detection baseline tend to extract fewer segments, while stable-criteria extracts many more. This can justify the difference between the recall values of different approaches in Figures 2.6a and 2.6b. Note that a solution which

[6]We have the average number of trajectories for *easy* and *strict* sets as 30 and 50, respectively.

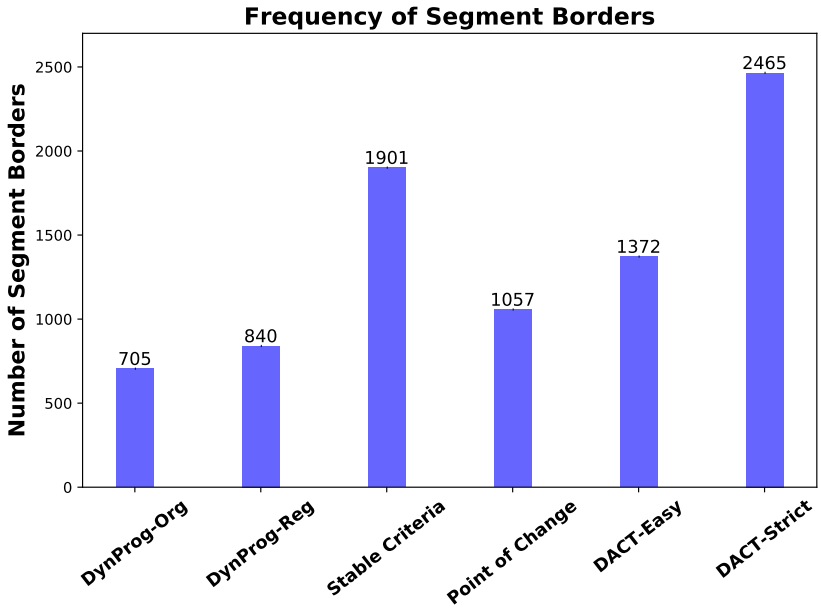


Figure 2.7: Frequency of extracted Segment Borders

maximizes the precision is preferred because we need *valid* segments to conduct a precise causality analysis to confidently derive the characteristics for a context. In Figure 2.7, DACT-Easy, and DACT-Strict show the number of annotations (segments) in ground truth sets. We omitted the values for Equal-Length and Random baselines as the number of extracted segments by these two approaches depends on the choice of η.

An important observation from Figures 2.6a and 2.6b is the linear relationship between precision and recall, which can be justified by the formulation of these two metrics, in that they have the same numerator but different denominators (Equations 2.5 and 2.6). The denominator for precision is the number of extracted segments (m), and for the recall is the number of specified segment borders in the ground-truth set (n). As shown in Figure 2.7, one can compare these numbers for different cases to obtain the slope of precision-recall lines. For instance, for the case of *DynamicProgramming-Org* and *DACT-Easy* as ground-truth set, we have $n/m = 1.95$, which is very close to the slope of corresponding line in Figure 2.6a, which is obtained as 1.94 by *Linear Regression* analysis.

2.5.3 dDescribe Evaluation

Context

As it is described earlier in the chapter, we define the context as combination of *location* (e.g., Interstate-270, from intersection Interstate-70 to intersection US-33) and *time* (e.g., weekdays between 3pm to 7pm). We show the list of routes in Table 2.2. For time, we use two granularity levels: *i. Type of Day* and *ii. Time of the Day*. The first level contains *Weekday (WD)* and *Weekend (WE)*, and the second level contains five time intervals including: *P1*: from 6am to 9:59am, *P2*: from 10am to 2:59pm, *P3*: from 3pm to 6:59pm, *P4*: from 7pm to 9:59pm, and *P5*: from 10pm to 5:59am. To find the appropriate time intervals, we used a one-year traffic congestion reports by Map Quest for the city of Columbus (Section 2.5.1). As shown in Figure 2.8, one can see how traffic condition (i.e., frequency of congestion events) is changing during the different hours of a day. We derived the aforementioned intervals regarding changes in traffic conditions.

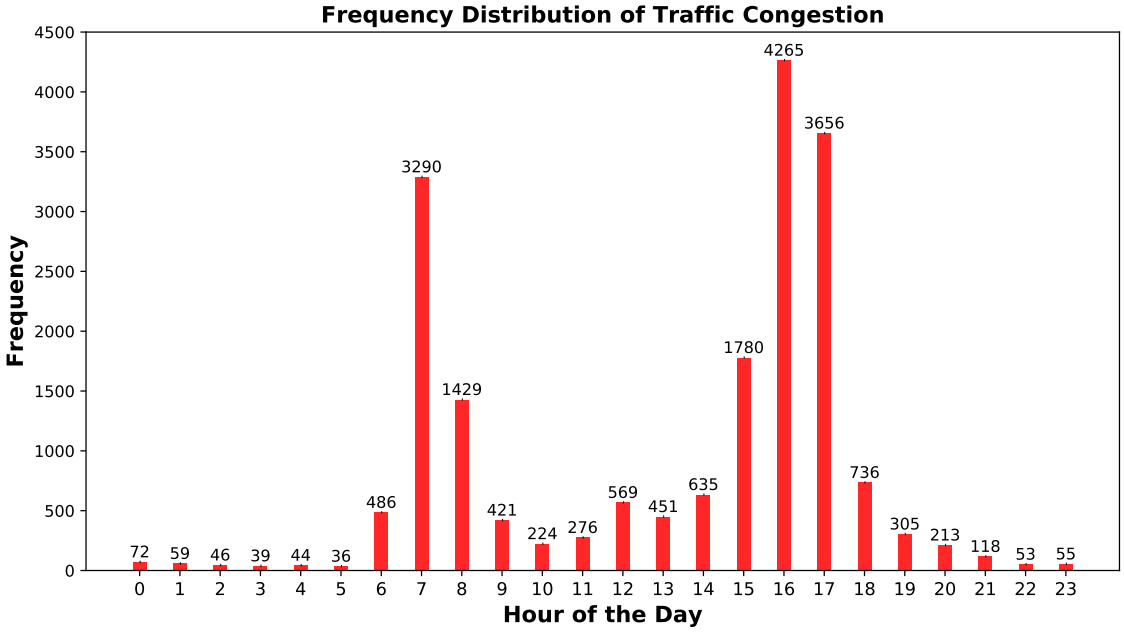


Figure 2.8: Frequency distribution of congestion events for the City of Columbus Ohio for one year (from Feb 2016 to Feb 2017), based on the MapQuest Traffic reports.

Causality Analysis

After showing the applicability of dSegment (see Section 2.5.2), we apply it on our evaluation set. Table 2.2 provides some statistics about the evaluation set and also reports the average number of extracted segments for trajectories in each route in the evaluation set. In summary, dSegment extracted 6,674 segments from trajectories in the evaluation set. Next, we present the result of applying dDescribe on the extracted segments to discover properties for each context. As it is described previously, we use physical facts, from OSM and HCA, and temporal-physical events, from traffic congestion reports, to create the event database $\mathcal{E}$. We conduct the causality analysis by introducing a new measure *Correlation* which we define it as follows: suppose that for a set of trajectories $\mathcal{T} = \{T_1, T_2, \ldots, T_M\}$, which happened in context C, we found a sequence of cutting points $CP_i = \langle p_{i_1}, p_{i_2}, \ldots \rangle$ for each $T_i \in \mathcal{T}$, as result of segmentation process (Section 2.4.1). Then, given an event database $\mathcal{E}$, we use Equation 2.7 to obtain the correlation for context C. In this equation, the *CheckRelevancy* returns 1 or 0. Later in this section we provide more details about this function with respect to the type of event.

$$Correlation(C, \mathcal{E}) = \frac{\sum_{i=1}^{i=M} \sum_{p \in CP_i} CheckRelevancy(p, \mathcal{E})}{\sum_{i=1}^{i=M} |CP_i|} \tag{2.7}$$

For causality analysis, we define three different tasks based on the source of event data, as we present them next.

Event Data as Physical Facts First, we use physical facts to build the event database $\mathcal{E}$ and then conduct the causality analysis. For this case, we define the *CheckRelevancy* function in Equation 2.7 as calculating the Haversine distance between a cutting point p and a physical event $e \in \mathcal{E}$, and then checking if their distance is lower than a pre-specified threshold th. We empirically set $th = 200m$, and it also takes the length of the route of each context into account. In this way, Figure 2.9a shows the correlation between cutting points (i.e., the extracted segments) of different contexts with physical facts. Note that correlation analysis is only done for those contexts for which we have enough data. Our observation is that on average, about 76.5% of the driving patterns (segments) are correlated with the

physical properties of routes. That said, different contexts may show different patterns of correlation, depending on the properties of each context.

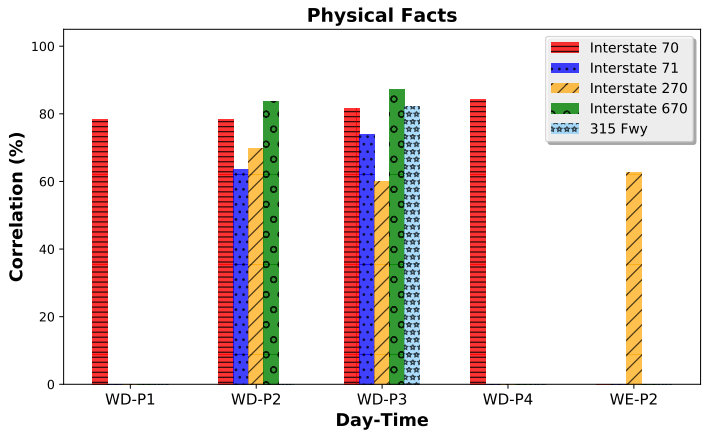
Event Data as Temporal-Physical Events The second analysis is to use the temporal-physical events to build the database $\mathcal{E}$ and conduct causality analysis. The challenging part here is to define the *CheckRelevancy* function in Equation 2.7. To do this, we propose a solution as follows which consists of two steps[7].

- Step 1: Heuristic to find potential congestion evidences. Given a trajectory T, we try to find sub-trajectories of minimum length 5, where the speed of all points in such sub-trajectories is less than 55 kmh. We consider such sub-trajectories as showing potential evidence of congestion. The minimum length 5 is given from [113], and the speed 55 kmh is the average congestion speed in the traffic congestion dataset (Section 2.5.1).

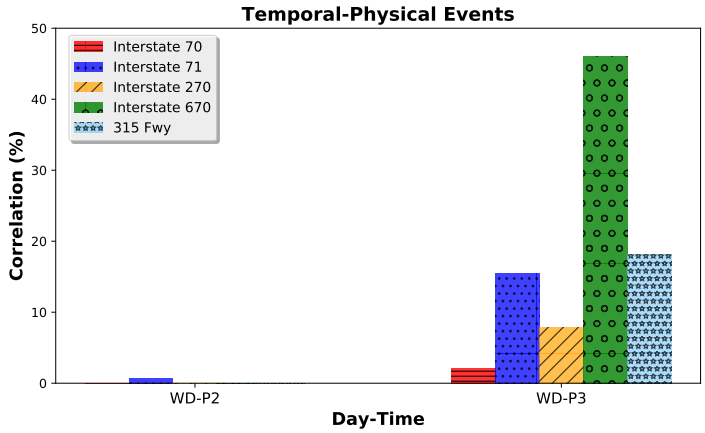
- Step 2: Finalize the decision about potential evidences. After finding potential evidence $\mathcal{C}$ of congestion, we scan through our traffic congestion dataset (Section 2.5.1) to see if there are at least 12 evidence which happened in the *neighborhood* of location of $\mathcal{C}$, within the same day of the week and hour of the day (e.g., Tuesday 4 pm). The minimum number 12 is chosen to reflect the observation of one report per month for a potential congestion spot. Also, in order to define the neighborhood, we use a 200 $meters$ threshold and calculate distance using the Haversine metric.

Following above guidelines, we identified 465 traffic congestion sub-trajectories within 1,421 trajectories of evaluation set. Now, the rest is straightforward: we need to find the correlation between identified congestion evidences with cutting points. For a cutting point ρ of trajectory T, if ρ is in the neighborhood (i.e., $th = 200m$) of at least one of traffic congestion evidences of T (if there is any), then *CheckRelevancy* returns 1, otherwise, it returns 0. Taking aforementioned, Figure 2.9b illustrates the correlation analysis results between driving patterns and temporal-physical events. On average, about 10.5% of driving patterns were correlated with traffic congestion.

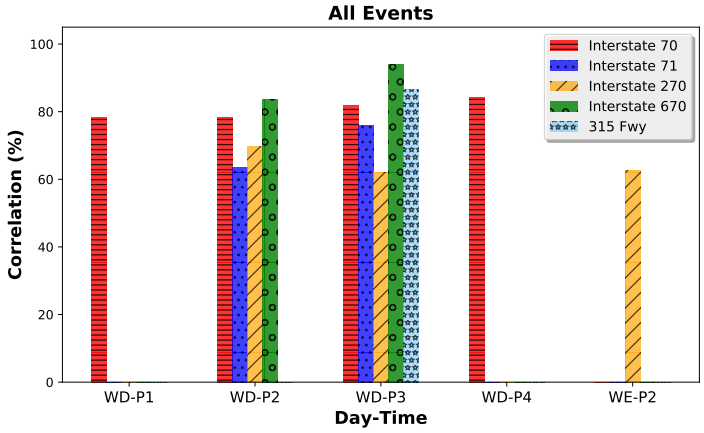
[7]Note that we only need these steps if there is not traffic data available to be matched with the timeline of the trajectory dataset. Otherwise, we can directly use the traffic congestion report data and perform correlation analysis.



(a) Physical



(b) Temporal-Physical



(c) All

Figure 2.9: Correlation of extracted driving patterns (segments) with (a) Physical Facts, (b) Temporal-Physical Events, and (c) All Facts and Events. WD and WE are stand for weekday and weekend. P1, P2, P3, and P4 are time intervals. We only show correlation for those contexts that we have enough data for.

Event Data as Union of Fact and Events Finally, we consider both physical facts and temporal-physical events to build the event database $\mathcal{E}$. Given a cutting point p of trajectory T, in order to find if it is correlated with an event $e \in \mathcal{E}$ by using function *CheckRelevancy*, we use either of approaches described previously, with respect to the type of the event. Figure 2.9c demonstrates the correlation of driving patterns with the set of all existing events, where, in summary, 78.1% of segments are correlated with at least one of the event types. Moreover, by comparing Figures 2.9b and 2.9c, one can see that both analyses lead to almost the same patterns of correlation. This shows a significant number of segments are correlated with both sources of events.

2.5.4 Deriving Characteristics of Context

Thus far, we have shown the applicability of dSegment, and also how to conduct the causality analysis in terms of dDescribe. Now, we show how one can use these two components to derive the characteristics for a context. We also describe other applications of the framework.

First-Order Insights

The results of causality analysis provide the strongest signal from which to derive the characteristics for a context. As an example, Figure 2.9a shows that one can expect to see about 82% correlation between driving patterns and physical properties of routes during weekdays, between 3 pm to 7 pm for Interstate-70. Another example is the effect of traffic on driving patterns in the context of 315-Freeway, during weekdays between 3 pm to 7 pm (Figure 2.9b).

Second-Order Insights

Besides the direct usage of dDescribe results, one can further analyze the results to identify *second-order insights* about a context. As an example, based on Figure 2.9c, one can see the correlation ratio for Interstate-270 during weekdays between 3 pm to 7 pm (WD-P3) is significantly lower than in other contexts. Also, about 83% of the evaluation data set belongs to the category of WD-P3. Such a lower correlation for a specific route may be interpretable with some other events (facts), rather than what we have used in dDescribe. To understand what these events may be, we examined the available Annual Average Daily

Table 2.3: Annual Average Daily Traffic (AADT) volume estimation for **2010** and **2014** for Franklin county of Columbus Ohio [73].

Route	**I-70**	**I-71**	**I-270**	**I-670**	**315 Fwy**
Route Length (miles)	7.5	11.1	10.5	7.7	11.2
#Vehicles (millions)	1.1	1.7	1.0	1.2	1.2
Truck Load (2010)	11.5%	9.7%	**13.5**%	5.3%	4.3%
Truck Load (2014)	9.4%	8.9%	**11.4**%	4.9%	3.7%

Traffic (AADT) reports which are provided by the Department Of Transportation (DOT) for Columbus Ohio [73]. Based on these reports for 2010 and 2014 (the most recent ones), as demonstrated by Table 2.3, we observed that the proportion of trucks that use Interstate-270 for transportation is significantly larger than other routes in the evaluation set. Thus, the presence of trucks may be a potential reason for the difference in the correlation of segments in this context. Such a finding can be considered as a second-order insight that is not directly derivable from the causality analysis results.

Other Applications of the Framework

DriveContext may also be used as an analysis tool for usage-based insurance purposes. As an example, for a given context c, the DriveContext may find driving patterns are on average 55% correlated with physical properties of the route. An insurance company may use such contextual information to study the behavior of an individual driver in order to evaluate how risky or safe he/she is, regarding the characteristics of context c (see Example 2.1.1). The resulting insights from our framework may also be used for driver coaching, which recommends further training to those drivers whose driving behavior in a context is not compatible with the properties of that context. Finally, by learning the characteristics of different contexts, our framework may be able to discover valuable insights about similarities and differences in driving habits for different types of roads, different cities, regions, etc.

2.6 Summary and Conclusion

In this chapter, we introduce the DriveContext framework as a solution for deriving characteristics of a context by extracting meaningful driving patterns (dSegment) and

analyzing the extracted patterns (dDescribe). Our proposed segmentation approach, to our knowledge, is the first behavior-based solution that takes the behavior of a driver into consideration. Moreover, dDescribe is also a novel method for conducting analysis across a set of spatio-temporal data to find underlying causes for driving patterns. Our analysis shows how the dSegment is comparable to state-of-the-art methods for finding meaningful driving patterns. In addition, the results of dDescribe show the ability of our framework to interpret driving patterns, leading to new insights. The proposed framework can be used for applications such as usage-based insurance, driver coaching, and urban planning.

Chapter 3: Characterizing Context by Geo-Spatiotemporal Pattern Discovery

In Chapter 2, we leveraged telematics data to characterize driving context. In this chapter, we use contextual data such as data on *traffic* and *weather*, to explore characteristics of different contexts, through geo-spatiotemporal pattern discovery. Relying on a simplistic definition of the spatiotemporal neighborhood is a common characteristic of the existing general-purpose pattern discovery frameworks. This makes them impractical for geo-spatiotemporal data (such as traffic and weather data). To address this challenge, we introduce a new geo-spatiotemporal pattern discovery framework to explore two types of patterns, *propagation* and *influential* patterns. Propagation patterns reveal the common cascading forms of geospatial entities in a region. Influential patterns demonstrate the impact of long-term geospatial entities on their spatial neighborhood within a time frame. We apply this framework on a large dataset of traffic and weather data at a countrywide scale, as an example of geo-spatiotemporal data, collected for the contiguous United States over two years. Our important findings include the identification of 90 common propagation patterns of traffic and weather entities (e.g., $rain \rightarrow accident \rightarrow congestion$), and interesting influential patterns with respect to "location," "type," and "duration" of long-term entities (e.g., *major construction* $\rightarrow$ *more congestions*), both of which provide insights on driving habits and infrastructure characteristics within different regions of the US. The outcome of this framework can be used for applications such as driving risk prediction, urban planning, exploring flaws in transportation infrastructure design, and traffic control and prediction.

3.1 Motivation

Spatiotemporal pattern discovery has gained considerable attention over the past decade, with various frameworks proposed to find interesting patterns [33, 18, 22, 37, 41, 51, 44,

75, 107, 105]. The application domains of relevance include public safety, transportation, earth science, epidemiology, climatology, and environmental management [85]. These frameworks can be used to discover patterns of collocation and co-occurrence; interactions and correlations; and cascading, sequential, or cause-and-effect relationship patterns. However, they all rely on a simplistic definition of spatiotemporal neighborhood, essentially spatial closeness based on a Euclidean or Cartesian system and temporal overlap [18, 22, 33, 51], which often makes their use impractical for applications such as traffic, transportation, or weather analyses. For example, a traffic accident on one lane of a freeway has no impact on traffic flow on an opposite lane, yet general-purpose frameworks will locate both lanes in a single neighborhood. Another example arises when studying the impact of a snow event (on traffic flow), which continues well past when the snow event has ended. The time overlap constraint required by existing frameworks would hinder such a study.

To address these challenges, we propose a new framework for finding patterns in geo-spatiotemporal data. This framework consists of two parts, one to explore *propagation* patterns, and the other to reveal *influential* patterns. Identifying propagation patterns requires the exploration of partially ordered sets of geospatial entities that are spatially collocated and temporally co-occurring, with potential "cause-and-effect" relationships between the entities. An example of this type of pattern is a `rain` event, which causes an `accident`, with the accident then causing `congestion`. Identifying influential patterns, on the other hand, requires studying the impact of long-term geospatial entities (e.g., major construction) on their spatial neighborhoods. An example of this type of pattern is the increase in the number of `congestion` events in a region because of a long-term `snow` event.

To explore propagation patterns, also referred to as "cascading patterns"[51] or "spatiotemporal couplings" [85], we propose a tree-pattern-mining-based process, which we name *short-term pattern discovery*. This process employs a strict definition of spatial neighborhood to ensure spatial collocation, and a definition of temporal co-occurrence specific to geo-spatiotemporal data and application domain constraints. To explore influential patterns, also referred to as "tele-couplings" [85], we propose a new process, which we name *long-term pattern discovery*. This process examines the effect of long-term entities on their neighborhood to reveal any significant impact. As in, and drawing from [58, 67], this process may be used to study impacts with respect to different *types*, different *locations*, and *duration* of long-term geospatial entities.

To evaluate our framework, we used a large-scale, real-world geo-spatiotemporal dataset of traffic and weather data. This dataset covers the contiguous United States[8], includes data collected from August 2016 to August 2018, and contains about 13.1 million instances of traffic entities (e.g., accident, congestion, etc.), and about 2.2 million instances of weather entities (e.g., rain, snow, storm, etc.). Through the processes mentioned above, we found 90 common patterns of propagation of relatively short-term traffic or weather entities. In addition, we carefully studied the impact of relatively long-term traffic or weather entities on traffic, and identified a variety of insights with respect to "location," "type," and "duration" of the entities.

The main contributions of this chapter are as follows:

- Short-term pattern discovery: We propose a new process for discovering propagation patterns in geo-spatiotemporal data, which models spatiotemporal collocation and co-occurrence in terms of tree structures and adapts an existing tree-pattern-mining approach to reveal prevalent patterns. In comparison to general-purpose frameworks, this method better suits the application domain requirements of a stricter definition of spatiotemporal neighborhood.

- Long-term pattern discovery: We propose a new process for discovering influential patterns in geo-spatiotemporal data and examining the impact of long-term geospatial entities on their neighborhood to reveal significant influential patterns. Exploring such patterns with existing frameworks is not feasible, due to lack of effective spatiotemporal neighborhood metrics to explore longer-term (i.e., lagging) impacts.

- Data collection and processing: We present a set of processes for collecting real-time traffic and historic weather data, using which we built a publicly available dataset of 13.1 million traffic entities (e.g., accident, congestion, construction, etc.) and 2.2 million weather entities (e.g., rain, snow, storm, etc.).

- Findings and insights: By applying our new framework on the above dataset of traffic and weather data, we present a range of insights for different regions in the United States. These insights may be further used for applications such as urban planning,

[8]The contiguous United States excludes Alaska and Hawaii and considers District of Columbia (DC) as a separate state.

exploring flaws in transportation infrastructure design, traffic control and prediction, impact prediction, and potentially the creation of smart cities.

3.2 Preliminaries and Problem Statement

In this section, we first provide preliminaries and definitions and then present the problem statement[9].

3.2.1 Definitions

- Geospatial Entity: a geospatial entity e is represented by a tuple $\langle type, start, end, loc \rangle$, which shows an entity of type *type*, happened in time interval $\big[start,\ end\big]$, and its location is specified by *loc*. The definition of *loc* is related to the application domain. For traffic data, we have $loc = \langle latitude, longitude, Street\_Name,\ Street\_Side, Zipcode,\ City,\ State \rangle$, where *Street_Side* shows the relative side of a street which is either *Right* or *Left*. For weather data, we have $loc = \langle airport\_code \rangle$, which represents the "airport" which e is reported from its weather station. A geospatial entity is called *long*, if it takes place over a relatively long time interval (see Section 3.5.2).

- Weak-Dependency Relationship: two *co-occurring* and *co-located* geospatial entities are called weakly dependent. Co-occurrence for two entities e_1 and e_2 means $0 < \big|e_1.start - e_2.start\big| \leq$ *T-thresh*, where *T-thresh* is a time-threshold. Collocation for two traffic entities requires *location matching* as well as *spatial closeness*. The former means that all location fields except the GPS coordinates should be the same. By latter, we require that $dist(e_1, e_2) \leq$ *D-thresh*, where *dist* is the Haversine distance function [153] based on GPS coordinates, and *D-Thresh* is a distance threshold. With respect to matching a pair of weather and traffic entities, collocation means a match between the "airport station" at which the weather entity is reported and the "airport station" closest to the traffic entity's location. Note that we do not define such a relationship between two weather entities.

- Child-Parent relationship: for two weakly dependent geospatial entities e_1 and e_2, e_1 is a parent for e_2 if e_1 begins before e_2. We treat parent-child relationships as indicative of

[9]Several of definitions in this section are adapted from [29].

cause and effect type of relationship. A weather entity may only be the parent (or cause) of a traffic entity, and we do not define such relationships between two weather entities.

- Tree structure: given a set of vertices $\mathcal{V} = \big(v_1, v_2, \ldots, v_n\big)$, we define tree $T = (V, E)$, where $V \subset \mathcal{V}$ and $E = \{e_1, e_2, \ldots, e_m\}$ is a set of edges, and each edge $e \in E$ connects a pair of vertices $v_i, v_j \in V$ using an *un-directed* edge. A tree is an *acyclic* graph, and vertices with the same parent are *siblings*. Trees in this work have a *root* node, sibling nodes are *un-ordered*, and nodes are *labeled*. Figure 3.1-(a) shows several examples of such tree structure. In this work, each node of a tree is a geospatial entity, and each edge shows a child-parent relationship between two entities.

- Embedded Subtree: given a tree $T = (V, E)$, we define a subtree as $S = (V', E')$, where $V' \subset V$ and $E' \subset E$. A subtree S is said to be an *embedded subtree* of T if for each edge $e = (v_a, v_b) \in E'$, v_a is an ancestor (and not necessarily the parent) of v_b in T.

3.2.2 Short and Long-term Pattern Discovery

We now formalize the two related problems studied in this chapter.

Short-term Pattern Discovery

Here we seek to find common short-term *propagation patterns* that indicate *how* geospatial entities cause other entities to happen. We represent a set of weakly dependent geospatial entities as un-ordered, rooted, labeled trees, where the entities are nodes, weak dependency relations are the edges, and entity types (e.g., rain, accident, and congestion) are the labels of the nodes. Thus, given a forest $F = \{T_1, T_2, \ldots, T_k\}$ of such trees, the short-term pattern discovery problem is about finding all embedded subtrees in F which are occurred relatively frequently. Formally, for a subtree S and tree T we define $support(S, T)$ by Equation 3.1:

$$support(S, T) = \begin{cases} 1 & \text{if } S \text{ is a subtree of } T \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Then, we define $support(S, F)$ by Equation 3.2:

$$support(S, F) = \frac{\sum_{T \in F} support(S, T)}{|F|} \tag{3.2}$$

For a subtree S, if $support(S, F) \geq min\_sup$, where min_sup is a user-defined minimum support threshold, then we say S is a frequent embedded subtree in F. An example of a

forest with some of frequently occurring subtree patterns is shown in Figure 3.1. In this example, we have a forest which includes four trees. Using a minimum support threshold of 75%, we identify several frequently occurring embedded sub-tree patterns, four of which are shown in Figure 3.1-(b).

We use "short-term pattern discovery" to indicate that we search for patterns of immediate or short-term impacts, as opposed to long-term impacts, which is discussed next. Note that although the relationship between entities may be a weak dependency, the frequency of occurrence (i.e., the support) is an indicator of a strong relationship. In other words, when weak dependencies appear frequently, such observations cannot be insignificant.

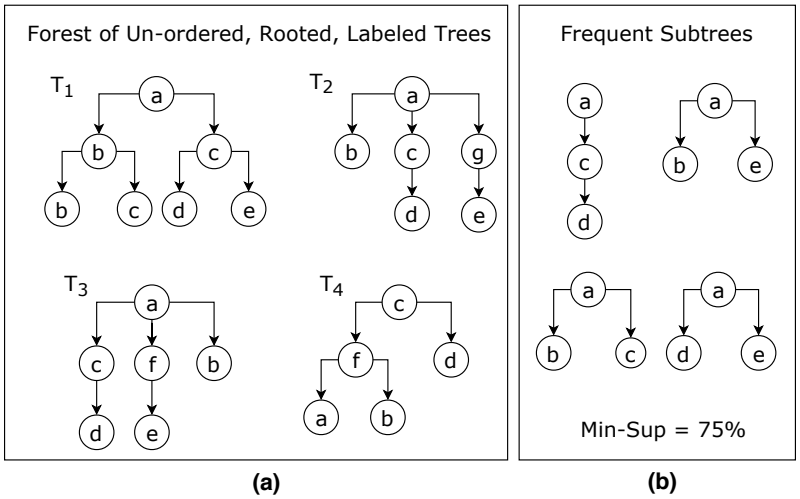


Figure 3.1: (a) A forest of four trees, (b) Four of embedded frequently occurred subtrees with a minimum support 75%.

Long-term Pattern Discovery

Long-term pattern discovery is about exploring the *magnitude of impact* of long-term geospatial entities on their neighborhood. As an example, consider a *major construction* event in region A, because of which, we might observe more congestion events in the same region (when compared to a time when there was not such a construction event). Given a long entity L, let $\mathcal{S}_R = [e_1, e_2, \dots]$ be the set of geospatial entities in the *vicinity* of that, where

R is the maximum distance threshold[10]. Let $L.start < e.start$ and $L.end > e.end$, $\forall e \in \mathcal{S}_R$. To study the impact of a long entity, we also define two other sets, $\mathcal{S}$–$before_R$ and $\mathcal{S}$–$after_R$. The former contains all geospatial entities which happened within distance R from L, during a time interval of the same length as L, but before L started. The latter contains all entities in the same neighborhood as L, but which happened after L ended. Given sets $\mathcal{S}_R$, $\mathcal{S}$–$before_R$, and $\mathcal{S}$–$after_R$, we define the problem of "long-term pattern discovery" as finding those long entities for which there is a significant difference between the size of set $\mathcal{S}_R$ and the two $\mathcal{S}$–$before_R$ and $\mathcal{S}$–$after_R$ sets. In other words, a statistically significant difference between the number of entities when a long entity like L is present, and the number of entities before or after L, shows the magnitude of the impact. We call such an occurrence a long-term or influential pattern.

Connection Between Problems

Once again, note that short-term pattern discovery is about finding *immediate* impacts. On the other hand, long-term pattern discovery is about exploring the "long-lastingness" of impacts (i.e., *lagging* impacts). In other words, these two are *complementary* problems, with each one focused on a separate aspect of dependency and pattern discovery, while using the same set of input data.

3.3 Related Work

Spatiotemporal pattern discovery has been thoroughly discussed in literature [18, 18, 33, 37, 41, 44, 51, 75]. Earlier work focused more on spatial prevalence and paid less attention to temporal aspects [18], while later work considered both aspects simultaneously [85]. The common process of spatiotemporal pattern discovery is to first define spatiotemporal co-occurrence and collocation criteria; then introduce an interest measure (e.g., participation index); and finally outline a *miner* algorithm to find interesting patterns [18]. Techniques in these papers being general-purpose solutions, rely on simplistic definitions of collocation (spatial) and co-occurrence (temporal), and unable to reveal complex spatiotemporal correlations (such as influential patterns). Further, they have been developed and only tested on small-scale (real-world or synthetic) data. To address these challenges with respect to

[10]For each $e_i \in \mathcal{S}_R$, its location is within distance R from L.

geo-spatiotemporal data, we propose a new framework that provides an appropriate and precise definition of collocation and co-occurrence criteria. Moreover, we outline the process of finding complex spatiotemporal patterns and prove its applicability through extensive experiments. Lastly, we apply our framework on a large-scale, countrywide geo-spatiotemporal dataset of traffic and weather data to explore interesting patterns.

Regarding the application domain, there are numerous studies for finding patterns in traffic and weather data, with the following goals: to study the impact of precipitation on likelihood or severity of accidents [16, 61, 55, 67, 116]; to explore the impact of weather on traffic intensity [39, 88]; to reveal the effect of climate change and weather condition on road safety [42, 58, 72]; to characterize road accidents locations [100]; or, to discover frequent spatiotemporal patterns in traffic data [44, 59, 97]. The scale of data in most of these studies is limited to one city or at most a few cities. Note that the interaction and correlations between the different types of traffic entities (e.g., accident, congestion, etc.) have not been studied before. Although the idea of long-term pattern discovery has been previously studied [16, 58, 67], we extend this idea by 1) examining a wider range of weather and traffic entity types besides precipitation; 2) exploring properties of different "locations"; and 3) analyzing the impact of "duration length" on traffic flow.

3.4 Dataset

In this section, we describe the process of preparing *traffic* and *weather* dataset, which serves as an example of geo-spatiotemporal data. In this dataset, traffic entities are collected in real-time using a REST API provided by *MapQuest* [151], and weather data is collected from *Weather Underground* [152]. In total, we extracted about 13.1 million traffic and 2.2 million weather entities, which are collected from August 2016 to August 2018.

3.4.1 Traffic Data

Data Collection Process

In this work, we use the MapQuest traffic API to collect real-world traffic data for a period of two years, from August 2016 to August 2018. To our knowledge, this API transmits the same traffic information which is distributed based on Traffic Message Channel (TMC) protocol [5]. Real-time traffic data is collected for the contiguous United States by 92

threads, each of which collected data for a bounding box (or geofence) of size $360km \times 360km$ on map. The frequency of API request was every 90 seconds from 6 am to 11 pm, and every 150 seconds from 11 pm to 6 am. To handle the timezone differences, each bounding box was mapped to one of the following timezones: Eastern, Central, Mountain, or Pacific. As the raw traffic entities came with GPS coordinates, we employed another process to perform *reverse geocoding*, and translate them to *address* which includes attributes such as street, street side, city, state, and zipcode.

Data Cleaning Process

Following cleaning steps are employed to prepare the data:

i. Resolving explicit and implicit duplicates: duplicates can be identified either by id (i.e., two entities have the same id) or by content (i.e., two entities of the same type which happened at the same time and location). The former is called explicit and the latter implicit. To resolve such cases, we keep one entity and remove the other one.

ii. Denoising the data: noise in this context is related to the "type" of traffic entity, where the TMC code provided as part of the information for an entity could be different from its default reported type by the traffic API. In order to deal with this problem, we first extracted 250 different TMC codes in our data, and manually checked and defined a valid *refined type* per each TMC code using [5] as reference. Finally, we replaced the new taxonomy with the default one.

Data Entity Description

Traffic entities are proxy for traffic flow in a region or area of interest. We define following taxonomy for traffic entities.

- Accident: a common type, which may involve one or more vehicles, and could result in fatality.
- Broken-Vehicle: refers to the situation when there is one (or more) disabled vehicle(s) in a road.
- Congestion: refers to the situation when the speed of traffic is lower than the expected speed.

Table 3.1: Details on Traffic Dataset, collected for the contiguous United States from Aug 2016 to Aug 2018.

Entity Type	**Raw Count**	**Relative Frequency**
Accident	1,169,507	8.9%
Broken-Vehicle	308,112	2.34%
Congestion	10,542,020	80.18%
Construction	209,933	1.60%
Event	32,817	0.25%
Lane-Blocked	246,832	1.88%
Flow-Incident	637,489	4.85%
Total	13,146,710	100%

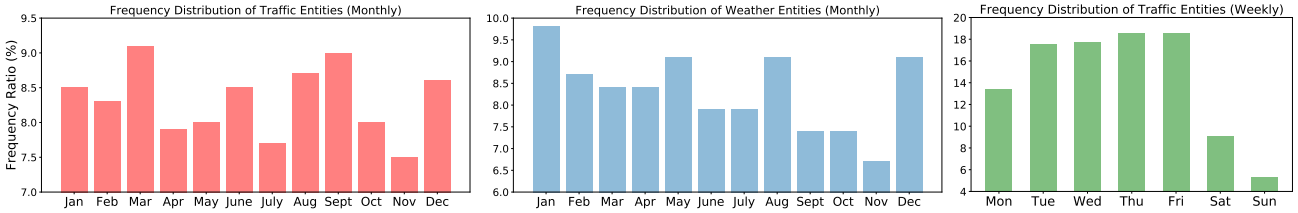
- Construction: refers to an on-going construction or re-paring project in a road.
- Event: refers to the situations such as *sport event*, *concert*, and *demonstration*.
- Lane-blocked: refers to the cases when we have blocked lane(s) due to traffic or weather condition.
- Flow-incident: refers to all other types of traffic entities. Examples are *broken traffic light* and *animal in the road*.

Table 3.1 provides more details on the traffic dataset. The most frequent entity type is "congestion" which includes about 80% of the data, and "accident" is the second most frequent entity type. Figure 3.2a also depicts the monthly frequency distribution, where the most entities are observed in March and September and the least in November. Additionally, *weekly* frequency distribution of traffic entities is shown by Figure 3.2c, where 'Friday" and "Sunday" are found to be the days with the most and the least number of recorded entities, respectively.

3.4.2 Weather Data

Data Collection Process

The raw weather data comes in the form of data records, where each record consists of several attributes such as temperature, humidity, wind speed, pressure, precipitation (in



(a) Monthly Traffic Distribution (b) Monthly Weather Distribution (c) Weekly Traffic Distribution

Figure 3.2: Relative frequency distribution of traffic and weather data, collected from Aug 2016 to Aug 2018, for the contiguous United States.

millimeters), and condition[11]. For each weather station, we have several data records per day, which are recorded upon any significant change in any of the measured attributes.

Threshold Definition Process

To define the taxonomy of weather entities, we require to extract some threshold values. To do so, we used the United State observations of temperature, wind speed, and precipitation amount for rain and snow for a period of seven years, from January 2010 to January 2016, and applied K-Means clustering algorithm [43] on each of these attributes. The obtained cluster centers are used as a threshold for these attributes. For temperature, we identified five cluster center values[12] $-23.7°$, $-8.6°$, $6.7°$, $21.3°$, and $35.8°$, which we refer them as *severe-cold*, *cold*, *cool*, *warm*, and *hot*, respectively. For wind speed, we found three cluster centers $13.2kmh$, $36.2kmh$, and $60kmh$, which we refer them as *calm*, *moderate*, and *storm* windy conditions, respectively. For rain, we identified three cluster centers 2.5, 7.1, and 11.6 millimeters, which we refer them as *light*, *moderate*, and *heavy* rainy conditions, respectively. Lastly, for snow we found three cluster centers 0.6, 1.7, and 2.5 millimeters, which we refer them as *light*, *moderate*, and *heavy* snowy conditions, respectively.

Entity Extraction Process

Given the above threshold values and the raw weather data records from August 2016 to August 2018, we processed each record to use it (if it represents an entity), merge it (if it is

[11]Possible values are *clear*, *snow*, *rain*, *fog*, *hail*, and *thunderstorm*.

[12]All degrees are in Celsius.

part of a previously found entity), or remove it (if it does not represent any entity), and define the following taxonomy:

- Severe-Cold: the case of having extremely low temperature, with $temperature \leq -23.7°$.
- Fog: the case where there is low visibility condition as result of *fog* or *haze*.
- Hail: the case of having solid precipitation including *ice pallets* and *hail*.
- Rain: the case of having rain, including any type of the rain, ranging from *light* to *heavy*.
- Snow: the case of having snow, including any type, ranging from *light* to *heavy*.
- Storm: the extremely windy condition, where the wind speed is at least $60kmh$.
- Precipitation: a generic label which we frequently observed in raw weather data, however, we have no further information to include them in any of the previously described entity types.

As described in Section 3.2, the raw weather data is collected from different weather stations located in different airports all around the country. In total, we used the data from 1,973 airport weather stations and extracted 2,178,949 weather entities for the period of two years. Table 3.2 provides more details on our weather data. Based on Table 3.2, the most frequent entity types are "rain", "fog", and "snow". Figure 3.2b also demonstrates the monthly frequency distribution of weather entities, where the most of the entities are collected in January and the least in November.

3.5 Pattern Discovery and Results

In this section, we describe the pattern discovery framework along with analyses and results. This framework consists of two major parts, one for the discovery of propagation patterns and the other for influential patterns. In the following subsections, we describe both parts in detail.

Table 3.2: Details on Weather Dataset, collected for United States from Aug 2016 to Aug 2018.

Entity Type	Raw Count	Relative Frequency
Severe-Cold	67,285	3.09%
Fog	454,704	20.87%
Hail	1,252	0.06%
Rain	1,384,588	63.54%
Snow	236,546	10.86%
Storm	14,863	0.68%
Precipitation	19,711	0.9%
Total	2,178,949	100%

3.5.1 Short-Term Pattern Discovery

We propose a multi-steps process to perform short-term pattern discovery in geo-spatiotemporal data and extract propagation patterns. Figure 3.3 illustrates the process, which includes: 1) finding *child-parent* relationship; 2) building *relation trees*, and 3) extracting *frequent tree patterns*.

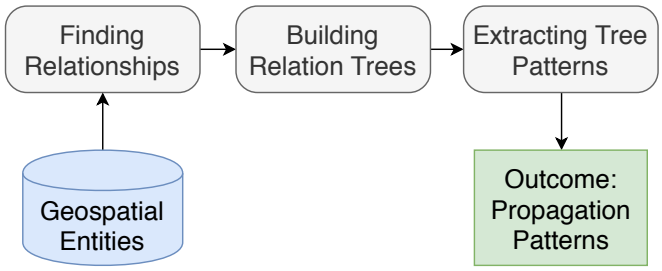


Figure 3.3: The process of Short-term pattern discovery.

Short-Term Child-Parent Relationship

The first step of this process is to extract all the weakly dependent pairs of entities and to define the child-parent relationship among them. In our example dataset of geo-spatiotemporal data, a weather entity w cannot have any parent, yet it can have an arbitrary number of children, which all should be traffic entities. Also, a traffic entity t can have

more than one parent, and any arbitrary number of children, which all should be traffic entities. We use thresholds *D-thresh* = 300 *meters* and *T-thresh* = 10 *minutes* to find all the short-term dependent entities, and then extract the child-parent relations[13]. These thresholds are found empirically, where *D-thresh* ensures the spatial closeness, and *T-thresh* is large enough to consider the delay in a "cause and effect" type of relationship, with respect to our application domain. Using this setting, we found 5,952,729 Child-Parent relationships among 13,146,710 traffic and 2,178,949 weather entities. In total, 39.33% of traffic entities have at least one weakly dependent entity, and 12.82% of weather entities have at least one weakly dependent traffic entity.

Relation Trees

The next step is to create relation trees using the extracted child-parent relations. Here, a tree is a rooted, labeled, un-ordered tree (see Section 3.2). In total, we created 1,723,637 trees out of 5,952,729 child-parent relations. The maximum number of nodes in a tree is 25. Note that as a traffic entity t can have more than one parent, we randomly pick one of them and ignore the rest. Regarding the size of the data and the number of trees, we expect that no existing pattern to be ignored using this strategy.

Frequent Tree Pattern Mining

The next step is to perform frequent tree pattern mining. As described in Section 3.2, the goal is to extract all those un-ordered subtrees which can be frequently found in our database of relation trees. Examples of such tree patterns with minimum support 75% are shown in Figure 3.1-b. Details of this process are described below.

- **Pattern Mining Algorithm**: in this work, we use the SLEUTH algorithm, a growth-based approach proposed by Zaki [23], to extract frequent, embedded, un-ordered sub-trees in our database of relation trees. The main reasons behind using this particular algorithm are: 1) it is proved to extract all the frequent embedded unordered tree patterns in a dataset [23]; and 2) the speed of the algorithm is sufficiently fast even for a large set of input trees, with the least memory usage. The standard implementation of this algorithm is also available online [155].

[13]For event type *snow*, we empirically set *T-thresh* = 40 minutes, because we expect to see a longer impact of snow on traffic flow.

- **City-level frequent pattern discovery**: we choose *city* as granularity level to perform pattern extraction in our illustration use case. A city includes several zip-codes, and each State includes several cities. After extracting frequent patterns for a city, we use them as frequent patterns of the corresponding State. The reason for not using State as granularity level is to avoid the problem of potential high diversity of relation trees in State-level which leads to not getting all the existing frequent patterns. As an example, consider a state which includes several cities with a different population and/or weather condition. Thus, the result of pattern discovery in City-level can be different from State-level, and because of the diversity, the frequent City-level patterns are infrequent in State-level, hence might be ignored.

- **Choice of Minimum Support**: setting the value of minimum-support is a challenging task, as we need to deal with two issues: 1) *size* of the data, and 2) *seasonality*. For some of the cities (such as Los Angeles, New York City, etc.) we have a very large number of relation trees. Therefore, using common values like 30%, 50%, etc. will not result in any appropriate outcome. Moreover, our data has seasonal property, which is about having recurrent patterns over time. Examples of such seasonal properties are the frequency of snow events in different months of the year, traffic entities frequency in weekdays versus weekends, etc. Given these two issues, we follow the approach proposed by Fournier-Viger [40] to adjust the minimum support based on the size of the data and also consider the seasonality. Based on this approach, we use Equation 3.3 to find the minimum support, where a, b, and c are positive constants that we empirically set them to be 0.004, 1.5, and 0.05, respectively. Here, x shows the number of relation trees in a set, and the minimum value for the relative minimum-support is 5%.

$$\text{min\_sup} = e^{-(ax+b)} + c \tag{3.3}$$

Results and Analyses

By applying the above process on our illustration dataset of traffic and weather entities, we extracted 708 frequent tree patterns for the contiguous United States. In total, there are 90 unique frequent patterns, where the minimum number of nodes in a tree pattern is 2 and the maximum is 7. Also, the top frequent pattern appeared in all 49 States, and 35 patterns appeared in just one State. Figure 3.4 demonstrates the top 20 frequent tree patterns. Each

Table 3.3: Details on *top cities*, *top states*, and *road-network focus* for top frequent short-term patterns.

Pattern	Top Cities	Top States	Road-network
1	Los Angeles (CA), Miami (FL), Houston (TX)	CA,FL,TX,NY,PA	Mixture
2	Los Angeles (CA), Miami (FL), Houston (TX)	CA,FL,TX,NY,PA	Mixture
3	Miami (FL), Chicago (IL), New York City (NY)	FL,TX,NY,PA, CA	Mixture
4	Los Angeles (CA), Houston (TX), Miami (FL)	CA,TX,FL,NY,PA	Mixture
5	Chicago (IL), Minneapolis (MN), Buffalo (NY)	PA,NY,MI,OH,IL	Interstates and Freeways
6	Chicago (IL), Minneapolis (MN), Detroit (MI)	PA,NY,MI,IL,OH	Interstates and Freeways
7	Los Angeles (CA), Houston (TX), New York City (NY)	CA,PA,NY,TX,FL	Mixture
8	Los Angeles (CA), Miami (FL), Houston (TX)	CA,TX,FL,NY,WA	Cities
9	Miami (FL), Chicago (IL), New York City (NY)	FL,NY,TX,GA,CA	Cities
10	Los Angeles (CA), Miami (FL), Houston (TX)	CA,FL,TX,NY,WA	Cities
11	Miami (FL), Los Angeles (CA), Houston (TX)	CA,FL,TX,NY,PA	Cities
12	Charlotte (NC), Raleigh (NC), Baton Rouge (LA)	SC,FL,CA,NC,TX	Cities
13	Chicago (IL), Detroit (MI), Columbus (OH)	PA,NY,MI,IL,OH	Interstates and Freeways
14	Los Angeles (CA), Houston (TX), Miami (FL)	CA,TX,FL,NY,GA	Cities
15	Los Angeles (CA), Houston (TX), Miami (FL)	CA,TX,NY,FL,WA	Cities
16	Los Angeles (CA), Houston (TX), Miami (FL)	CA,TX,FL,WA,NY	Cities
17	Denver (CO), Atlanta (GA), Chicago (IL)	CA,CO,TX,NY,GA	Cities
18	Miami (FL), Chicago (IL), New York City (NY)	FL,TX,NY,GA,WA	Cities
19	Los Angeles (CA), Houston (TX), New York City (NY)	CA,PA,NY,TX,WA	Mixture
20	Tampa (FL), Riverview (FL), Miami (FL)	FL,CA,TX,GA,MA	Mixture

pattern in this figure is described by the number of States which have instances of that, the average support value, and the bi-hourly frequency distribution of instances of the pattern based on the start time. Here, *Cng*, *Acd*, *Snw*, *Flow*, and *Cold* are short for congestion, accident, snow, flow-incident, and severe-cold, respectively. Each pattern shows how traffic or weather entities can be propagated in a region, on a short-term basis. For instance, pattern 1 shows congestion which is propagated and caused another congestion. Likewise, pattern 2 shows another propagation pattern of congestion entities, which is like a chain effect in adjacent locations. Moreover, Table 3.3 demonstrates some other details for each of the top short-term frequent patterns, in terms of *top cities*, *top-states*, and *road-network*. The "road-network" shows which *type* or *part* of the road-network is the most commonplace to observe instances of a pattern; where it can be inside a city (cities for short), interstates and freeways which connect different cities or states, or a mixture of both.

We summarize our observations based on the frequent patterns as follows:

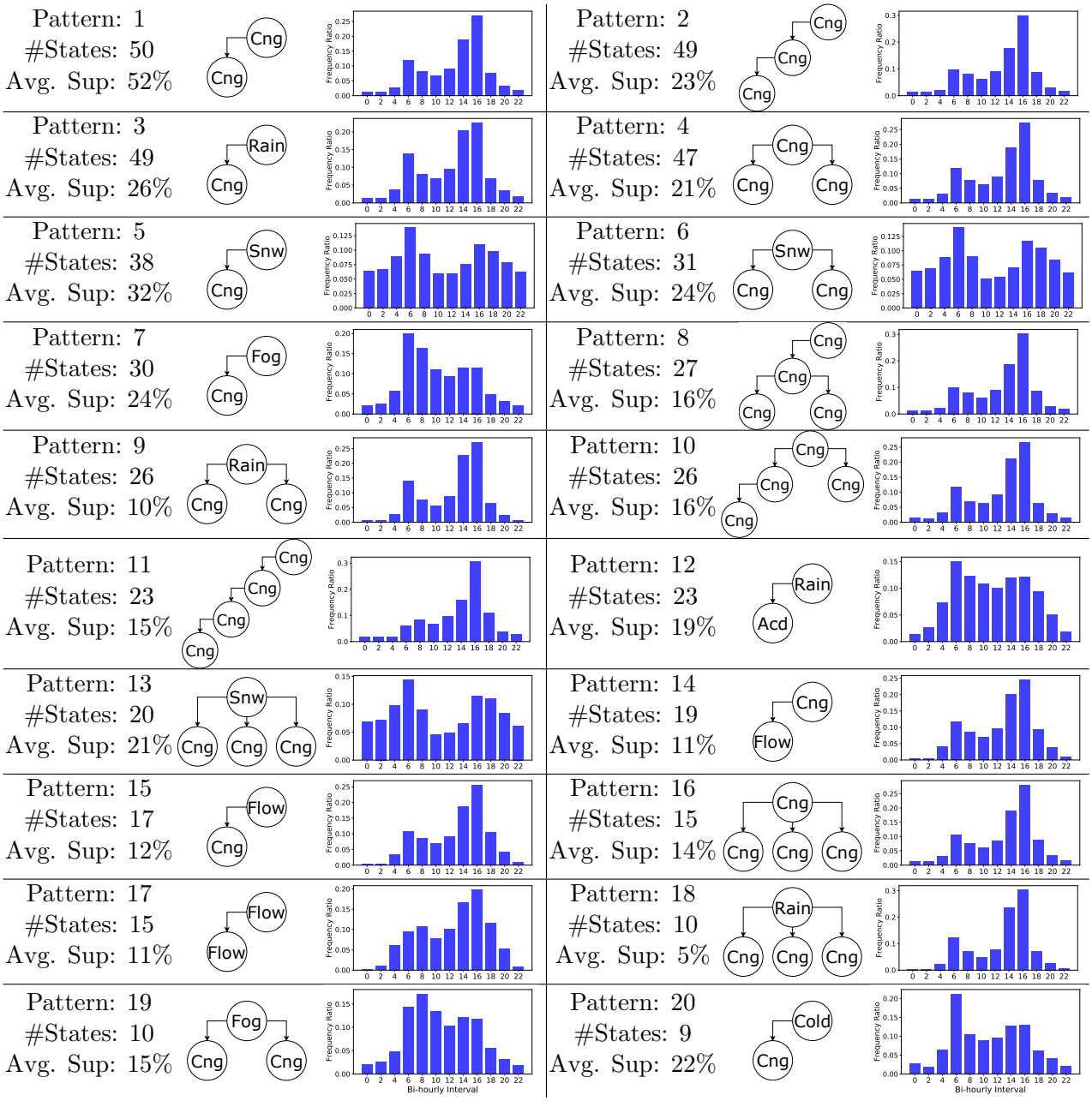


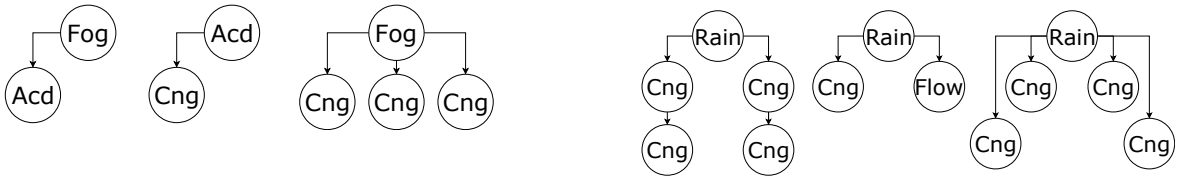
Figure 3.4: Top frequent tree patterns found based on the short-term dependency relation trees. "Cng", "Acd", "Flow", "Snw", and "Cold", are short for congestion, accident, flow-incident, snow, and severe-cold, respectively.

– *Weather initiated patterns*: in total, 50 of 90 unique frequent patterns are initiated by a weather event, where 17 of these patterns are initiated by rain, 14 by snow, 11 by

fog, and 8 by the other types of weather entities. These observations demonstrate the significant impact of weather on traffic, which has also been frequently discussed in much prior research [28, 32, 39, 110]. However, here we reveal the propagation patterns which show "how" these weather entities can cause traffic entities.

- *Bi-hourly distribution*: in Figure 3.4 we can see the bi-hourly distribution for instances of different propagation patterns. For the majority of congestion-related patterns, the peak of the frequency distribution is in the afternoon rush hour. For weather initiated patterns (except the rain-related cases), we have the peak of frequency on morning rush hour. The only rain-related pattern with the peak on morning rush hour is pattern 12, which leads to an accident. These are interesting observations where a weather event usually causes more traffic issues in the morning rather than the afternoon.
- *Average support*: usually, the *simpler* patterns have higher average support values. The less node which a tree has, the simpler the pattern is. Also, snow-initiated patterns usually have higher support values than rain-initiated ones. However, two very common rain-initiated patterns, i.e., patterns 3 and 12 have quite large supports, which is compatible with previous observations like [32, 110, 116].
- *Top cities*: based on Table 3.3, usually we have the same group of cities among top-cities of a congestion-related pattern. However, if such a pattern is initiated by a specific weather event, then we might have a different combination of top cities (e.g., patterns 3, 5, and 6). Although accidents could happen in any city with a high density of population, pattern 12 is an interesting observation that shows rain causes more accidents in cities with relatively less density of population such as Charlotte and Raleigh in NC. This observation could be related to the flaws in transportation infrastructure design or common driving habits in these cities.
- *Road network*: in Table 3.3, we can see the focused portion of road-network for instances of different patterns. Snow-initiated patterns (i.e., patterns 5, 6, and 13) usually happened on interstates and freeways, while most of the rain-initiated patterns happened within cities' road-network. Also, most of the complex congestion-related patterns happened within cities' road-network, and average support of patterns that mostly happened in a city is lower than those which mostly happened on interstates and freeways, or the whole set of road-network.

– *Comparing Top Cities*: in Figure 3.5, we compared two of the top cities with the most traffic issues, i.e., Los Angeles and Miami, in terms of the propagation patterns which we observed in one of them but not in the other. Based on these patterns, we can see more *complex* rain-related patterns in Miami, and more accident and fog-related patterns in Los Angeles. The nature of such observations may be related to weather conditions, transportation infrastructure, or driving habits in these cities.



(a) Los Angeles (CA) patterns

(b) Miami (FL) patterns

Figure 3.5: Comparing some of distinct top short-term patterns between two top cities, Los Angeles (CA) and Miami (FL).

To further analyze the short-term patterns, we created a one-hot vector of size 90 for each state that represents presence or absence of each short-term pattern. By applying K-means clustering [43] on these vectors we categorized different states based on their short-term propagation patterns. To find the best number of clusters, we adapted Description Length (DL) for K-Means [94], which is represented by Equation 3.4. In this equation, $p(.)$ is the probability density function based on the distance of each data point x from its cluster center c_x, P is the number of parameters of distribution function, K is the number of clusters, and X is the set of all data points. By assuming the distribution function for distance from cluster centers to be a Gaussian distribution, we set $P = 2$. By choosing K from set $[2, 3, \ldots, 10]$, we found the optimal number of clusters to be 4, which provides the minimum description length.

$$DL(K) = -\sum_{x \in X} log(p(||x - c_x||)) + \frac{1}{2} P \ log(|X|) + K \ log(|X|) \tag{3.4}$$

We also employed the *Silhouette* metric to find the optimal number of clusters, using which we obtained results shown in Figure 3.6. The results show the optimal number of

clusters to be 4. Note that we omitted the Silhouette results for $k = 2$ and $k = 10$, as these choices would result in too few or too many clusters that are not desirable.

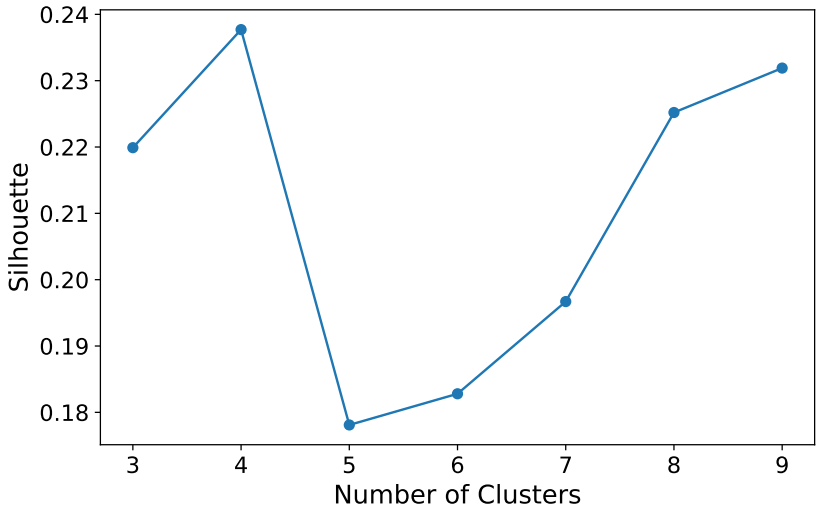


Figure 3.6: Analysis of the best number of clusters using Silhouette metric

Table 3.4: Clustering of 49 states into 4 clusters based on their short-term patterns, using K-Means.

Cluster	**States**
Cluster 1	AL, AR, CT, DC, DE, IA, IN, LA, MA, ME, MI, MN, MO, MS, NE, NH, OH, OK, RI, SD, TN, VT, WI
Cluster 2	AZ, CO, ID, KS, KY, MD, MT, NC, ND, NJ, NM, NV, OR, PA, SC, UT, VA, WV, WY
Cluster 3	FL, GA, IL, NY, TX, WA
Cluster 4	CA

Table 3.4 shows the result of clustering, in which we profile clusters as follows:

- **Cluster 1**: It mostly contains states with fewer traffic incidents (as related to weather). These are either states with lower population (e.g., NE, SD, etc.); or states where the impact of weather is mitigated by effective road crews (e.g., OH and MN).

- **Cluster 2**: It mostly contains states with considerably more traffic issues in comparison to the states in cluster 1. Distinguished patterns which only observed for this cluster are chain of accidents and complex snow-initiated patterns.

- **Cluster 3**: It contains states with at-least one major city with significant traffic issues. Distinguished patterns observed for this cluster are those that initiated by construction, rain, severe-cold, and storm.

- **Cluster 4**: It contains only one state whose traffic patterns bore no similarity to any other state. Majority of distinguished patterns of this cluster are complex congestion-related, fog-initiated, and flow-incident related ones.

It is worth noting that the states that were clustered together were not necessarily located in the same geographical region and might not have the same weather condition during the different seasons. However, their propagation patterns of traffic and impact of weather on traffic was found to be the same, which led to them being in the same cluster.

3.5.2 Long-Term Pattern Discovery

In this section, we primarily describe the process of long pattern discovery to study the magnitude of the impact of long-term entities. This process which is shown by Figure 3.7 is summarized into three steps; 1) extracting long-term entities, 2) retrieving vicinity entities, and 3) performing statistical significance testing. We explain each of these steps in the following subsections.

Extracting Long-term Entities

Regarding the definitions in Section 3.2, we call an entity to be long if it lasted for a relatively long time interval. Therefore, the first step is to define a threshold to extract long entities. To do so, we first obtain the distribution of duration of the entities over the whole dataset and then mark the 99 percentile of the duration as a threshold to extract long entities. Based on the distribution of duration of entities in our example dataset of

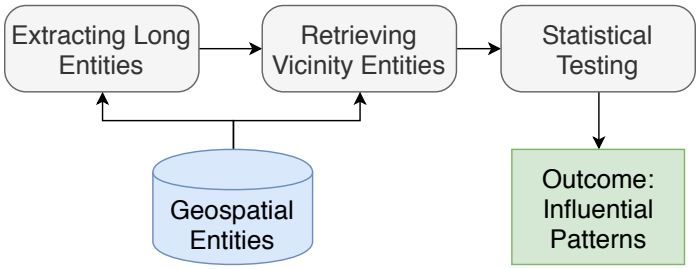


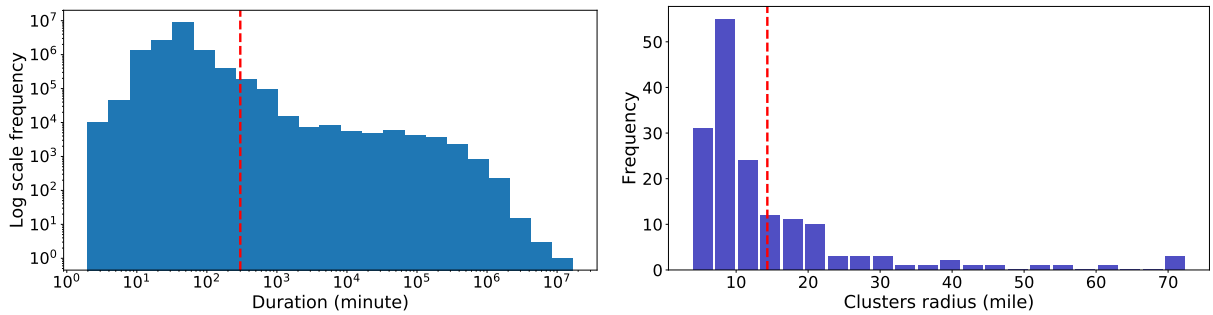
Figure 3.7: The process of Long-term pattern discovery to find *influential* patterns.

geo-spatiotemporal data, shown by Figure 3.8a, the 99 percentile of duration is found as ~ 300 (represented by the red dashed line). Also, we set the maximum duration to 21 days to deal with outliers which are relatively about 5% of the long entities in our data. The next step is to resolve *overlaps*, where overlap here refers to the case of having time and spatial overlap for a pair of long geospatial entities. To explore and merge such overlaps, we propose Algorithm 3. In this algorithm, for each long entity l, we first find the list of all long entities which have a time conflict with l, and also happened in the vicinity of l (lines 2–7). Time overlap will be identified by function "co-occurrence" in line 4 of the algorithm, using the start and end time of the entities. For spatial overlap, we use function "collocation" in line 4. Generally, two geospatial entities e_1 and e_2 are collocated if their geographical distance is less than a threshold ρ. However, for the case of a traffic entity t and a weather entity w, they have spatial overlap if they happened in the same region which is covered by the *airport station* which w is reported from[14]. After finding all the conflicted long entities for a given long entity l, lines 8–11 merge these entities by 1) changing the start and end time for l; 2) updating the location of l to the center of all the conflicted entities; and 3) changing the type of l to the concatenation of all unique type of entities in list. Finally, the list of conflicted entities will be removed from the set of all long entities (line 12).

Retrieving Vicinity Entities

After extracting the set of all long entities and handling the overlapped cases, we retrieve vicinity geospatial entities for each long entity. For a long entity L, we need to find subsets

[14]See definition of weak-dependency between geospatial entities in Section 3.2.



(a) Entities' duration distribution.

(b) Clusters' radius distribution.

Figure 3.8: Distribution of (a) Traffic and weather entity duration, and (b) Radius of clusters generated by DBSCAN on a sample of 2 million traffic entities.

$\mathcal{S}_R$, $\mathcal{S}$–$before_R$, and $\mathcal{S}$–$after_R$, given the maximum vicinity distance R. Following is the detail of creating such sets for a long entity L:

- $\mathcal{S}_R$: for this set, we look for all those geospatial entities which happened in vicinity distance R from L, their start time is strictly after the start time of L, and they all finished by the end time of L.
- $\mathcal{S}$–$before_R$: this set is similar to the previous one unless we pick a different time interval to find vicinity entities. Such interval is called "before", and Figure 3.9 shows how to choose that with respect to the start end time of long entity L. Based on this process, we move the start and end time of L to $W + D$ days before, where W stands for one week, and D shows the duration of L in days. In such an interval, we extract all the geospatial entities which happened in vicinity distance R from L.
- $\mathcal{S}$–$after_R$: similar to previous one, but we move the start and end days of L to $W + D$ days after.

The main reason for using this strategy to define the "before" and "after" intervals is to keep the same condition, from *day of the week* and *time of the day* point of view. Also, similar ideas have been previously applied in literature as *control day* [58] and *matched interval* [67]. Regarding our illustration use case, we only count the number of traffic entities to create vicinity lists $\mathcal{S}_R$, $\mathcal{S}$–$before_R$, and $\mathcal{S}$–$after_R$ for each long traffic/weather entity. Now, the question is how to determine parameter R? To define this parameter, we separate the cases of long traffic and long weather entities.

Algorithm 3: Merge Geospatial Overlaps

Input: Long entity set $\mathcal{L}$, and distance threshold ρ

for l ***in*** $\mathcal{L}$ **do**
  $List = []$
  for l' ***in*** $\mathcal{L}$ **do**
    if $co\text{-}occurrence(l, l')$ ***and*** $collocation(l, l', \rho)$ **then**
      List.add(l')
    end
  end
  $l.StartTime = \min_{\forall e \in List} StartTime(e)$
  $l.EndTime = \max_{\forall e \in List} EndTime(e)$
  $l.location = center_{\forall e \in List}(List)$
  $l.Type = concat_{\forall e \in List}(e.Type)$
  $\mathcal{L} = \mathcal{L} - List$
end

Output: $\mathcal{L}$

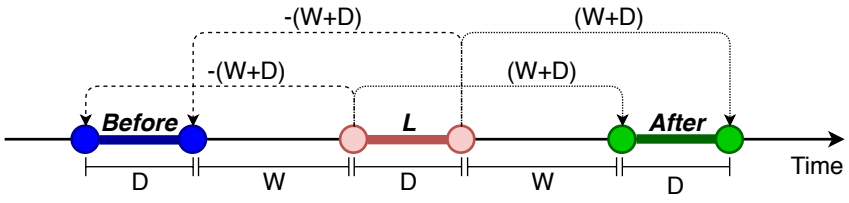


Figure 3.9: Definition of *before* and *after* time intervals with respect to long entity L. Duration of L is shown by "D" (days), and "W" stands for one week time interval.

Extracting R for long Traffic entities. As the location of each traffic entity is specified by latitude and longitude coordinates, we can define R as a "distance value" from the location of a long traffic entity. In order to determine the value of R, we use a random sample of two million traffic entities from the original set described in Section 3.4, and call it set S_1. Then, we apply DBSCAN algorithm[43] to cluster entities in set S_1. In resulting clusters, we find the radius of each cluster as the maximum distance from the center and report the average radius across all the clusters as R. However, before performing DBSCAN clustering, we need to define two parameters ϵ and *minPts* as DBSCAN inputs. The first parameter shows the maximum distance between two entities to be considered as neighbors, and the second parameter shows the minimum required number of neighbor entities for

an entity to not be considered as an outlier. To find these two parameters, we follow the approach suggested in [148], which is a data-driven process. First, we take a sample of size of 0.5 million traffic entities from the original dataset and call it set S_2. Then, we find the closest neighbor to each entity in this set and report a distance threshold which based on that, a majority of entities have their closest neighbor by that distance. To do this, we calculate the distance between each pair of entities in S_2, and report the 99 percentile of the distribution of the closest neighbor distance across the whole set which is obtained as 4.09 *miles*. We use this value as our ϵ for DBSCAN. Now, using the distance value 4.09 miles, we calculate the number of neighbors for each traffic entity within this distance in set S_2. Then, we report the 99 percentile of the distribution of the number of neighbors in ϵ-neighborhood, which we obtained that as 463. At this point, we obtained $\epsilon = 4.09$ *miles* and $minPts = 463$ as DBSCAN parameters. Applying DBSCAN on S_1 using the aforementioned parameters, we found 191 clusters, and the average radius of these clusters is obtained as 14.03 *miles*. Figure 3.8b shows the distribution of the radius for 191 clusters, and the red dashed line demonstrates the average radius $R = 14.03$ *miles*.

Extracting R for long Weather entities. For long weather entities, R is more like a symbolic representation, which is large enough to identify weather entity w and traffic t as neighbors, if the airport station which w is reported from is same as the station which zipcode t can be mapped to, as the closest airport station. In such a case, R can range from 0 to tens of miles, depending on the city and area of coverage by an airport station

Statistical Significance Testing

Given the set of all long entities, we first categorize them into disjoint buckets based on some criteria (e.g., location, type, or duration). Then, for each bucket, we compare the values of $\mathcal{S}_R$, $\mathcal{S}$–$before_R$, and $\mathcal{S}$–$after_R$ for all the existing long entities, to determine whether there is any significant impact of a long entity or not. For this purpose, we design six different testing scenarios and use *two-sample t-test* to test the difference between sample means. For a bucket B, we first calculate the following mean values. Using $\mathcal{S}_R$ for all the long entities in the bucket, we calculate μ_L by taking their average. Similarly, using $\mathcal{S}$–$before_R$ and $\mathcal{S}$–$after_R$, we calculate μ_{before} and μ_{after}, respectively, based on all the long entities in bucket B. Last, we take the average of $\mathcal{S}$–$before_R$ and $\mathcal{S}$–$after_R$ for each long

entity in bucket B, and take the average of average values and report that as μ_{avg}. Now, we define the following tests for bucket B:

- T_1: $\mu_{avg} = \mu_L$ versus $\mu_{avg} < \mu_L$. A one-sided test which examines whether or not the number of geospatial entities during a long entity is larger than this number when there is not such long entity.
- T_2: $\mu_{avg} = \mu_L$ versus $\mu_{avg} > \mu_L$. Similar to previous one, but with the opposite alternative hypothesis.
- T_3: $\mu_{before} = \mu_L$ versus $\mu_{before} < \mu_L$. A one-sided test which examines whether or not the number of geospatial entities during a long entity is larger than when the long entity is not started yet.
- T_4: $\mu_{before} = \mu_L$ versus $\mu_{before} > \mu_L$. Similar to previous one but with the opposite alternative hypothesis.
- T_5: $\mu_{after} = \mu_L$ versus $\mu_{after} < \mu_L$. A one-sided test which examines whether or not the number of geospatial entities during a long entity is larger than when the long entity is ended.
- T_6: $\mu_{after} = \mu_L$ versus $\mu_{after} > \mu_L$. Similar to previous one but with the opposite alternative hypothesis.

Note that in all of the above tests, the first condition is always considered as *null hypothesis* and the second one as *alternative hypothesis*.

Bucketing Criteria. Regarding our application domain as an example of geospatial data, we use three different criteria to create disjoint buckets of long entities, which are *State*, *Duration*, and *Type*. For "State", each bucket contains all the long entities which happened in the same State. For "Duration", we first define several duration buckets (intervals) and then assign each long entity to its bucket based on its duration. For "Type", we create buckets of long entities, where each bucket contains all the entities of the same type.

Positive and Negative Impacts. The positive (negative) impact refers to the cases where the value of $\mathcal{S}_{\mathcal{R}}$ is larger (lower) than $\mathcal{S}$–$before_R$, $\mathcal{S}$–$after_R$, or their average. A significant positive impact can be determined by tests T_1, T_3, or T_5. A significant negative impact can be determined by tests T_2, T_4, or T_6.

Results and Analyses

By applying the above process on our illustration dataset of traffic and weather entities, we extracted 280,649 long entities, and after merging the overlaps, we ended up with 148,237 long entities. To merge the overlaps by Algorithm 3, we used $\rho = 14.03$ miles, that is, the same value as parameter R to define the spatial neighborhood for a long-term entity. In this way, we ensure that after the merge, there is no pair of long entities with spatial neighborhood overlap. Table 3.5 provides the details on top-15 types of long entities, before and after the merge. Note that after the merge process, some of the types are combined and generated new type labels (e.g., Rain_Event). Next, we present the result of long-term pattern discovery in terms of studying the magnitude of impact for long entities. In this way, we first present the results for different bucketing criteria and then conclude the section by a discussion on important observations and results.

Table 3.5: Top 15 long entity types and their frequency.

Before Merge		**After Merge**	
Type	Frequency	Type	Frequency
Construction	113,984	Rain	38,253
Rain	41,668	Snow	25,820
Event	32,144	Construction	22,373
Snow	27,723	Fog	19,553
Fog	20,847	Event	16,753
Congestion	17,314	Severe-Cold	11,671
Flow-Incident	13,099	Congestion	3,206
Severe-Cold	12,083	Flow-Incident	2,381
Storm	733	Construction_Event	1,332
Other	440	Construction_rain	758
Lane-Blocked	253	Storm	709
Accident	226	Congestion_Flow-Incident	675
Precipitation	72	Congestion_Event	516
Broken-Vehicle	49	Event_Rain	514
Hail	14	Congestion_Construction	359
total	280,649	total	144,873

Bucketing by State. By bucketing the long traffic and weather entities based on their location which is demonstrated by State, we first show the ratio of these entities for different States in Figure 3.10. Besides weather and traffic entities, we also have a mixed category,

which is the result of the merging process. Next, we compare the value of $\mathcal{S}_R$ with the average of $\mathcal{S}$–$before_R$ and $\mathcal{S}$–$after_R$ for each long entity in each State, and present the distribution ratio of *increase*, *decrease*, or *no-change* as result of the comparisons. Figure 3.11 shows the results of this comparison. Finally, we use significance tests T_1 and T_2 to examine hypotheses for each bucket (i.e., State), and Figure 3.12 demonstrates the results of this comparison. Note that here we represent $1 - p\text{-}value$, and we also show three confidence levels of significance, that is, 90%, 95%, and 99% by three red lines. The results of other tests for State-level bucketing are not presented as we do not observe any significant difference between their results and what we obtained using tests T_1 and T_2.

Bucketing by Duration. The next part of the experiment is to perform bucketing of long entities based on duration criteria. In this way, we create a series of disjoint time intervals as our buckets. Then, assign each long entity to one of these buckets based on their duration. Figure 3.13 shows the results of six different significance tests based on duration bucketing. Here, x-axis shows those time intervals which for them we have at least 200 instances of long entities. Three red lines show three significance levels 90%, 95%, and 99%.

Bucketing by Type. Last, we bucket the long entities based on their type and perform testing for each bucket. Figure 3.14 shows the results of the six significance tests for this experiment, where the x-axis shows the label of each bucket which is the type of the entities in that bucket. Also, note that here we just represent the top 15 types of entities as described in Table 3.5, and omit the rest in the interests of space.

Given the above results, we summarize the important observations as follows:

– *Distribution of Long Entities*: based on Figure 3.10, we can see the difference between States given the distribution of long entities. As examples, we have states with a dominant distribution of long traffic entities such as DC and NJ, and State with the opposite situation, such as MN, ND, SD, and VT. Such observations can be justified based on the properties of these States, from the population density to the span of civilized places and free lands.

– *State-level Distributions*: comparing Figures 3.10 and 3.11, one can see a correlation pattern, where for those States with more long traffic entities, we usually see higher ratio of "increase" in the value of $\mathcal{S}_R$ in comparison to the average of $\mathcal{S}$–$before_R$ and $\mathcal{S}$–$after_R$

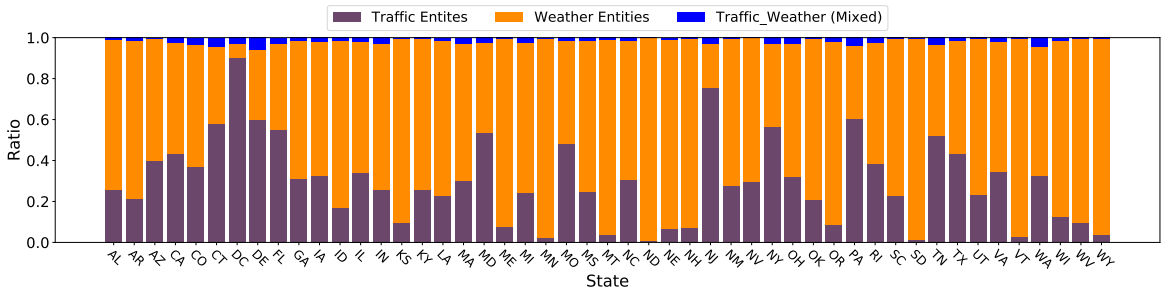


Figure 3.10: Distribution of long traffic, weather, and mixed long-term entities per State.

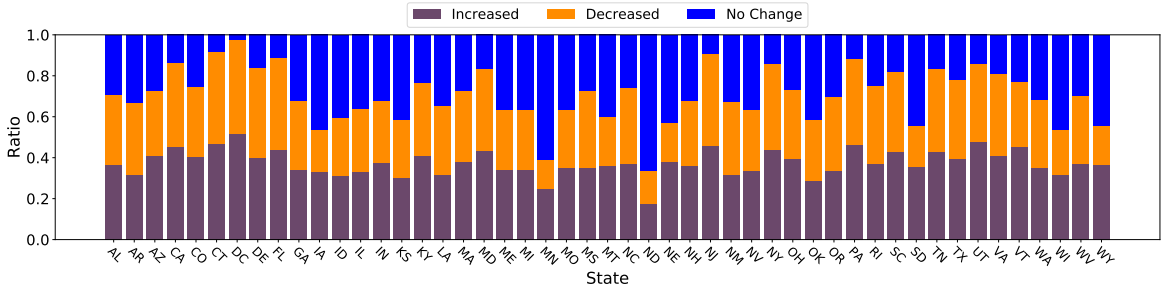


Figure 3.11: Ratio of increase, decrease, or no-change in number of traffic entities during long traffic or weather entities for different States.

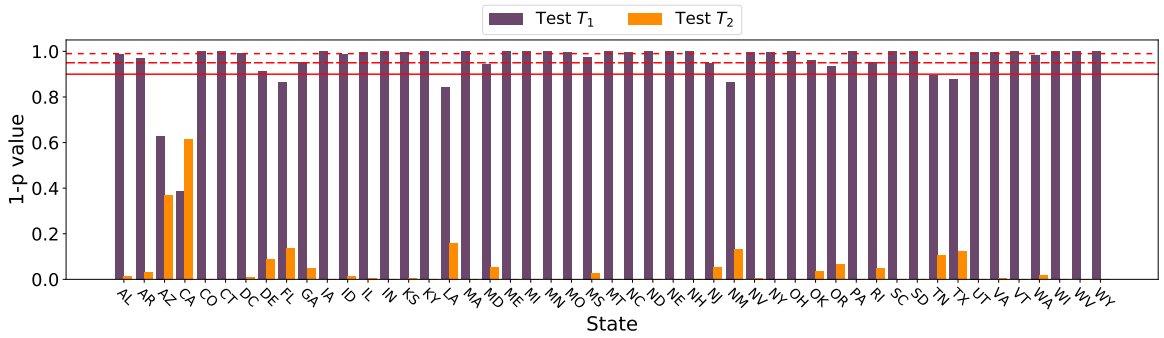


Figure 3.12: Statistical significance testing by test T_1 and T_2 for *state* buckets. Three red lines show different levels of significance confidence 90%, 95%, and 99%.

(e.g., DC, NJ, and PA). On the other hand, for the States with a very high frequency of weather entities, we observe a higher ratio of "no change" (e.g., MN and ND).

– *State-level Impacts*: based on the results of statistical significance testing, for a majority of States we observe that the existence of a long-term weather or traffic entity has a

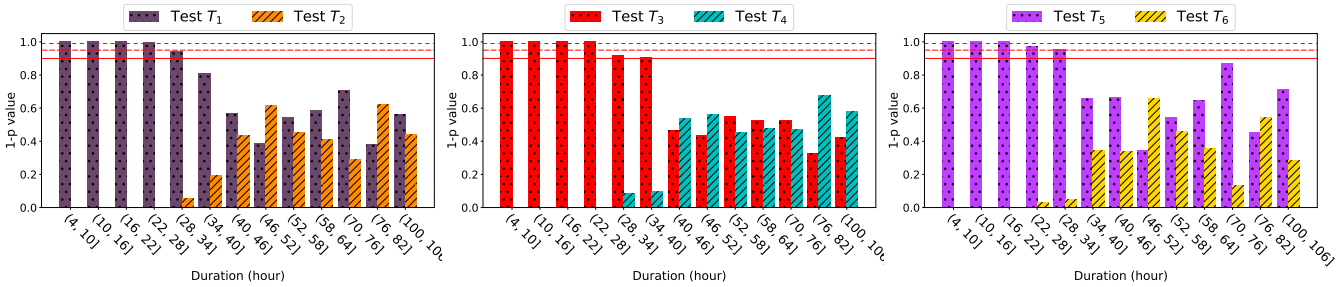


Figure 3.13: Statistical significance testing by tests (a) T_1 and T_2, (b) T_3 and T_4, and (c) T_5 and T_6, for *duration* buckets. Three red lines show different levels of significance confidence 90%, 95%, and 99%.

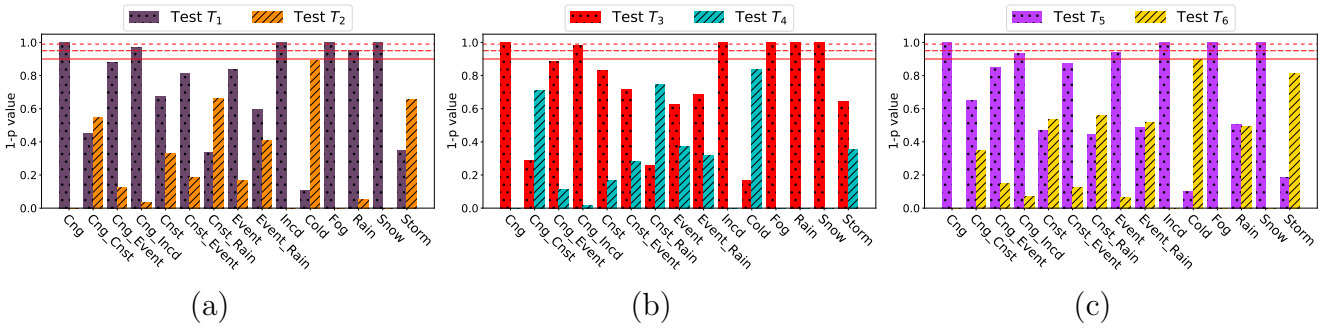


Figure 3.14: Statistical significance testing by tests (a) T_1 and T_2, (b) T_3 and T_4, and (c) T_5 and T_6, for *type* buckets. Three red lines show different levels of significance confidence 90%, 95%, and 99%. Cng, Cnst, Incd, and Cold are short for Congestion, Construction, Flow-Incident, and Severe-Cold, respectively.

significant impact on traffic flow. Also, in the majority of cases, we found that the result of test T_1 to be significant, which means a positive impact. Out of 49 States in our experiment, we found 30 to be significant by confidence 99%, 8 by confidence 95%, and 5 by confidence 90%. In our experiments, we also found the existence of long-term traffic or weather entities have not much impact on traffic flow for States AZ, CA, FL, LA, NM, and TX, where three of these States (i.e., CA, FL, and TX) are the top-3 States with the most observation of traffic entities in two years. Such observation, in some sense, discloses that in a State with usually more traffic issues, the existence of a long-term traffic or weather entity has not much impact on traffic flow. Moreover, CA is the only State which for that the p-value is found to be lower for Test T_2, which, although is not significant,

shows a unique condition where a long-term weather or traffic entity causes fewer traffic issues in comparison to the time when we have not such a long-term entity.

– *Impact by Duration of Entities*: based on the result of all the six tests in Figure 3.13, we conclude that the shorter the duration of a long-term entity, the more significant the impact is. Also, the common trend shows for those long-term entities which lasted for more than 40 hours, we usually do not observe any significant impact. Such observation might be due to adaptation to the new condition and common driving habits. Also, comparing the results of tests T_3 and T_4 with tests T_5 and T_6, provides evidences to usually expect more positive impacts based on *after* interval, rather *before* interval, for long entities which are lasted more than 28 hours. Given that a majority of such long entities are construction projects (about 75% in our data), we can derive two potential interpretations. First, after a long construction project, we might observe a more smooth traffic flow (that is, fewer entities), even in comparison to the time when it is not started. Second, this observation can be also related to the fact that after a long construction project, there might be a significant group of drivers who are still using alternative routes which they were using during the construction.

– *Impact by Weather Entities*: given the results of all the six tests in Figure 3.14, we observe the significant impact of all available types but "storm". However, we have an interesting diversity among the impacts of different types of weather entities. Usually for "fog", "snow" and "rain", based on Tests T_1 and T_2, we see a positive impact on traffic, while for "severe-cold" we observe a negative impact. This observation reveals that in extremely cold temperatures, we should expect to see more smooth traffic flow probably because of fewer vehicles on the roads. Tests T_3 through T_6 also support such conclusion. Comparing rain and snow, based on tests T_3 through T_6, we can see long-term snow has a positive impact in comparison to both time intervals, before and after the event. However, rain is only found to have a positive impact in comparison to the time when the entity is not started yet. This observation reveals that after a long rain event, we expect to see even more traffic issues, while in case of snow, we have most of the traffic issues happened during the event. Also, we can see fog and snow are similar in this regard. Although none of the tests provide significant results for "storm", we observe the probability of having more traffic issues after the storm is higher, based on tests T_5 and T_6.

– *Impact by Traffic Entities*: among long-term traffic entities, based on Figure 3.14, we observe significant impacts by "congestion", "event", and "flow-incident". In the case of long-term "congestion", we have a positive impact in comparison to both time intervals, before and after. For "flow-incident", we also observe a similar situation. However, for a long-term "event", we only observe a positive impact in comparison to the time when the "event" is terminated (test T_5). This is interesting to see that long-term construction almost has no significant impact on traffic flow. However, based on tests T_3 and T_4, we probably expect to see more traffic issues during a long-term construction in comparison to the time that it is not started yet.

3.6 Summary and Conclusion

To overcome the shortcomings posed by the existing general-purpose spatiotemporal pattern discovery frameworks, we present a new framework to extract *propagation* as well as *influential* patterns in geo-spatiotemporal data using improved and novel techniques. To extract propagation patterns, which indicate immediate impacts, we use a strict definition of spatial collocation and co-occurrence relationships to create relation trees, and then perform tree-pattern-mining in a forest of relation trees. Influential patterns, which show lagging impacts, explore the impact of long-lived geospatial entities on their neighborhood, and we use statistical techniques to identify such patterns. Using a new and unique geo-spatiotemporal dataset of traffic and weather entities, which is collected, processed, and augmented for the contiguous United States over two years, we explore 90 prevalent propagation patterns, where 50 of them are initiated by weather (mostly observed in morning) and the rest by traffic entities (mostly observed in afternoon). We also study the lagging impact of long-term traffic or weather entities with respect to location, duration, and type of entities. Interestingly, we identify a positive impact of long-term entities in a majority of the states, except a few ones such as CA, FL, and TX. In general, long-term entities that last for at most 28 hours have the maximum impact on traffic flow. Long-term congestion, snow, rain, fog, severe-cold, and flow-incident cause the most significant lagging impact on traffic flow.

Chapter 4: Characterizing Driving Style

In chapters 2 and 3, we proposed solutions to derive characteristics of driving contexts using telematics as well contextual data. In this chapter, we propose a solution to capture variations in driving behavior that will serve to discriminate different drivers from each other. Such variations serve as the *identity* or *style* of a driver. This task has become a prerequisite for a variety of applications in which driving style encodes essential information, including: usage-based insurance, driver coaching, driver action prediction, and designing autonomous vehicles. We present a deep-neural-network architecture, we term *D-CRNN*, for building high-fidelity representations for driving style, that combines the power of convolutional neural networks (CNN) and recurrent neural networks (RNN). Using CNN, we capture semantic patterns of driver behavior from trajectories (such as a turn or a braking event). We then find temporal dependencies between these semantic patterns using RNN to encode driving style. We demonstrate the effectiveness of these techniques for *driver identification* by learning driving style through extensive experiments conducted on several large, real-world datasets, and comparing the results with state-of-the-art solutions. These experiments also demonstrate a useful example of bias removal, by sampling *dissimilar* trajectories for each driver to prevent *spatial memorization*. Finally, we present an analysis of the contribution of different attributes for driver identification; we find the best combination comprises *Engine RPM*, *Speed*, and *Acceleration*.

4.1 Motivation

Analysis of telematics data with the goal of learning *driving style* for an individual driver has become feasible thanks to the availability of different means to collect and store large amounts of driving data. Learning driving style is the task of capturing *variations* in driving behavior for different drivers. This task is specifically important for driver risk prediction

used by insurance companies. The shift towards more personalized insurance products has created a need for understanding driving style at an individual level. Learning driving style is also useful for driver coaching, in order to help drivers improve their skills [31]; driver action prediction, to prevent dangerous events such as accidents [115, 98]; and designing autonomous vehicles that mimic actual drivers [80].

Several approaches have been proposed recently for learning driving style [93, 95, 117, 123, 124]. To be formalized as a pattern recognition task, learning driving style is about constructing a model that seeks to predict the identity of a *driver* for an input *trajectory*, using "style and behavioral variation information" extracted from trajectory data. In other words, this task resembles *trajectory classification*, where labels are drivers' identities. However, unlike studies such as [131], we aim to use driving behavior information instead of spatial information to label trajectories, which is a more complicated task and provides more benefits and insights for the aforementioned applications.

Limitations to prior work include: using a small set of drivers and trajectories; not correcting for bias in data (such as high spatial similarities between the trajectories for a given driver), which leads to utilizing spatial information instead of learning style; using over-simplified models that do not fully extract and utilize driving behavior data; and incurring greater cost for data collection than necessary such as by using specially equipped vehicles to collect data. Mitigating the impact of spatial similarity is to *correctly* assess the ability of different models to encode "driving style information," an important objective in this study. Suppose that R_1 and R_2 represent the set of prevalent distinct routes that drivers d_1 and d_2 take every day. Given these sets, we can train a classifier to properly distinguish between trajectories taken from d_1 and d_2. This is because trajectories taken from d_1 (which comprise routes in set R_1) are significantly different from those taken from d_2 (which comprise routes in set R_2). Such a classifier would largely utilize spatial similarity information, which is not an objective when we seek to derive and utilize driving style information.

In this chapter, we present a new framework that addresses these limitations. Our data are extensive and are collected from the *CAN-bus*[15]. The data include a wide range of data elements regarding the operation of a vehicle, as well as sensory data from the GPS,

[15]Controller Area Network (or CAN-bus) is a communication mechanism to transfer a variety of information related to operation of a vehicle.

accelerometer, and magnetometer. We preprocess the input data for each driver to limit the *spatial similarity* between trajectories, and to have a diverse set of trajectories for each driver. We also study the importance of different attributes to explore driving style and show the greater discriminative power of a selected set of attributes coming from CAN-bus in comparison to the GPS and other sensory data. To identify drivers via encoding driving style, we propose using *D-CRNN*, which combines CNN, RNN, and a fully connected (FC) component. Using the CNN component, we derive semantic information about driving patterns (a turn, a braking event, etc.). Using the RNN component, we capture temporal dependencies between different driving patterns to encode behavioral information. Finally, the FC component uses the encoded behavioral information to build a model that properly predicts the driver's identity when given a trajectory as input. We show the effectiveness of our proposal in comparison to state-of-the-art techniques, through extensive experiments on several large, real-world datasets provided to us via our industry collaborators.

This chapter makes the following contributions:

- We present a deep-neural-network architecture for driver identity prediction. We show the superiority of our results in comparison to state-of-the-art methods.
- We demonstrate a useful example of bias removal by sampling dissimilar trajectories for each driver, in order to ensure our model is not simply using *spatial memorization* to discriminate drivers. Note that mitigating the impact of spatial information is important to fairly assess the ability of a model to encode driving style information, an important objective in this study.
- We explore the differential discriminative power of various attributes for characterizing driving style. Determining which attributes are most relevant to driver identification may serve to achieve application goals, such as driver coaching.

4.2 Problem Statement

Suppose we have a database $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$ of N trajectories, where each trajectory T is a time-ordered sequence of data points $\langle p_1, p_2, \ldots, p_{|T|} \rangle$. We represent each data point p by a tuple $\big(t, Speed, Accel, Rpm, Lat, Lng, Head, AclX, AclY, AclZ\big)$, where t is the timestamp; *Speed* shows the ground velocity of the vehicle (metric is m/s); *Accel* is the

acceleration or the rate of change of velocity (metric is m/s^2); Rpm represents engine's revolutions per minute; Lat and Lng represent positional latitude and longitude data; $Head$ shows the heading (or bearing) of the vehicle which is a number between 0 and 359 degrees[16]; and $AclX$, $AclY$, and $AclZ$ represent the data coming from the accelerometer sensor which show the acceleration of the vehicle toward different axes in a three dimensional (3D) coordinate system. Additionally, suppose that we have a look-up table which maps each trajectory $T \in \mathcal{T}$ to a driver id d, where $d \in \mathcal{D}$, and $\mathcal{D}$ is the set of all drivers. Given the preliminaries, we formalize our problem as follows:

Input: A trajectory T.

Model: A predictive model M to capture variations in driving behavior to derive driving style information.

Goal: Predict identity of driver for trajectory T based on driving style information.

Optimization Objective: Minimize prediction error.

4.3 Related Work

Driver identification by learning driving style from trajectory data has been the topic of much prior research work [50, 93, 124, 95, 117, 123, 127]. Dong et al. [93] presented a deep-learning framework to learn driving style, which uses GPS data to encode trajectories in terms of statistical feature matrices. Then, using two existing models, one CNN and one RNN, driver identity is predicted for a given trajectory. Using the same type of input, Dong et al. [111] proposed an auto-encoder-based neural network model with a shared RNN component to perform representation learning for trajectories, which can be used for driver identification and trajectory clustering. Likewise, Kieu et al. [131] proposed a similar auto-encoder-based architecture, but using CNN as the shared component, and transforming a trajectory to a location-based 3D image as input to perform trajectory clustering and driver identification. Aim of such frameworks is to use both spatial and driving behavior information for the task of driver identification, as opposed to our objective which is to only encode driving style information.

Driver prediction based on information collected for a *turn* segment is proposed by Hallac et al. [95], using a hand-crafted rule-based classifier, which at max classifies a group

[16]Heading value 0 shows the *north* and 180 shows the *south* direction.

of 5 drivers. Driver prediction for a group of 30 drivers using a random forest model, as proposed by Wang et al. [117], achieved an accuracy of 100% for predicting the actual driver, where such result might be due to using high spatially similar trajectories taken from a small group of drivers [117]. Data in both [95] and [117] came from the CAN-bus. Using equipped vehicles with a variety of sensors and cameras installed inside and outside the vehicle, as well as information coming from the CAN-bus, Ezzini et al. [124] used a set of tree-based models for driver prediction. In [123], the authors proposed a driver prediction model only by using GPS data. They used GPS data to create a statistical feature vector of size 138 for each trajectory. Then, using a Random Forest classifier they classified small groups of 4 to 5 drivers [123]. Using a large set of input features coming from CAN-bus communication which includes 665 features, Hallac et al. [127] proposed a deep-learning model to build embedding for input trajectory data. Then, they used the embedding for a variety of tasks, including driver prediction using a regression model [127].

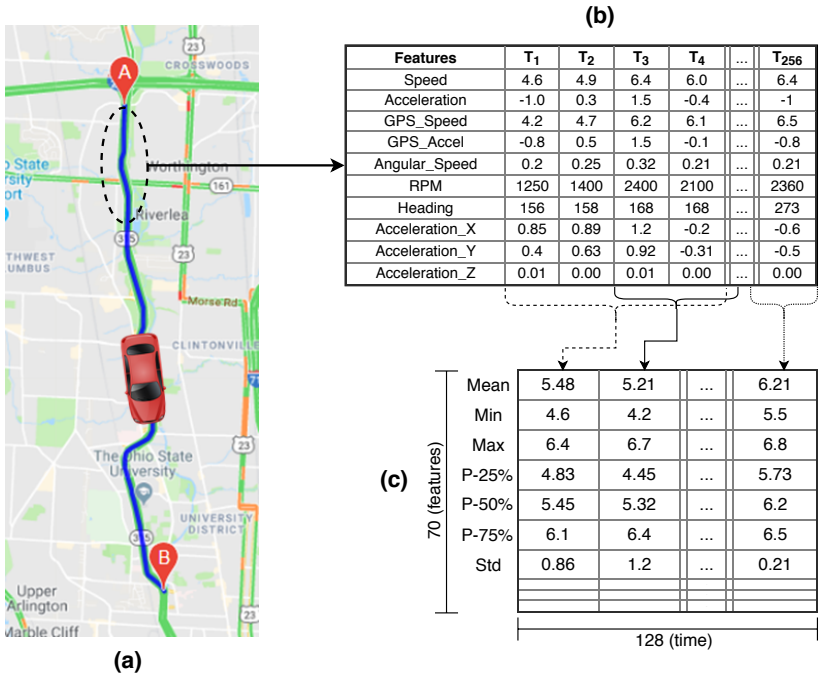
Unlike several of existing work (such as [95, 117, 124, 123, 127]), we model and predict based on a large set of drivers. Also, for the first time in literature, we do trajectory filtering based on the spatial similarity between input trajectories for each driver, to prevent spatial memorization and perform driver identification solely based on driving style representation learning. Moreover, we specifically design a deep-neural-network architecture to better capture the driving style for the task of driver identification. Lastly, similar to [95, 117, 127], we use data collected from the CAN-bus, and several other sensory data, which are rather easy to collect in large-scale, unlike the data which is used by [124].

4.4 Proposed Architectures

In this section, we first describe the feature encoding process. Then, we describe our deep-neural-network architecture to learn driving style, called Deep Convolutional Recurrent Neural Network or D-CRNN for short.

4.5 Feature Encoding

Feature encoding is the process of transforming a trajectory into the form of an input for a machine-learning model. To do so, we adapt a method applied in [93, 111], which is illustrated by Figure 4.1 and its main steps are described below.



Features	T_1	T_2	T_3	T_4	...	T_{256}
Speed	4.6	4.9	6.4	6.0	...	6.4
Acceleration	-1.0	0.3	1.5	-0.4	...	-1
GPS_Speed	4.2	4.7	6.2	6.1	...	6.5
GPS_Accel	-0.8	0.5	1.5	-0.1	...	-0.8
Angular_Speed	0.2	0.25	0.32	0.21	...	0.21
RPM	1250	1400	2400	2100	...	2360
Heading	156	158	168	168	...	273
Acceleration_X	0.85	0.89	1.2	-0.2	...	-0.6
Acceleration_Y	0.4	0.63	0.92	-0.31	...	-0.5
Acceleration_Z	0.01	0.00	0.01	0.00	...	0.00

Mean	5.48	5.21	...	6.21
Min	4.6	4.2	...	5.5
Max	6.4	6.7	...	6.8
P-25%	4.83	4.45	...	5.73
P-50%	5.45	5.32	...	6.2
P-75%	6.1	6.4	...	6.5
Std	0.86	1.2	...	0.21

Figure 4.1: Feature Encoding Process: from (a) raw trajectory, to (b) basic feature map, and to (c) aggregate feature map. Here we set $L_1 = 256$ and $L_2 = 4$.

Fixed-Length Segments. Given a trajectory $T = \langle p_1, p_2, \ldots, p_n \rangle$, we first partition that to smaller sub-trajectories (or segments) by a window of size L_1, with a shift of $L_1/2$. The overlap between neighboring segments helps to prevent information loss during the partitioning.

Basic Feature Map. After partitioning T to a set of segments $S_T = \{s_1, s_2, \ldots, s_k\}$, where $k = 2n/L_1 - 1$, the next step is to generate a feature map (matrix) for each segment $s \in S_T$ which encodes several attributes for each data point. Potentially, we use the following attributes to describe each data point: (1) Speed, (2) Acceleration, (3) GPS_Speed, (4) GPS_Acceleration, (5) Angular_Speed, (6) RPM, (7) Head, (8) AclX, (9) AclY, and (10) AclZ. We call these *basic features*. Assuming that we select a subset $\mathcal{F}$ of basic features, for a segment of length L_1, we represent that by a basic feature map of size $|\mathcal{F}| \times L_1$ (see Figure 4.1–(b) as an example).

Aggregate Feature Map. The last step is to transform a basic feature map to an *aggregate* feature map which encodes more high level driving data instead of point-wise, low-level

data, and also deals with outliers. In this way, we put each L_2 ($L_2 < L_1$) columns of a basic feature map into a frame with shift of $L_2/2$, and for each frame we calculate seven statistics *mean*, *minimum*, *maximum*, 25^{th}, 50^{th}, 75^{th} *percentiles*, and *standard deviation*. Describing each data point with a subset $\mathcal{F}$ of basic features, the result of this process is a matrix of size $7|\mathcal{F}| \times 2L_1/L_2$. See an example in Figure 4.1–(c).

4.5.1 D-CRNN

Deep Convolutional Recurrent Neural Network – or D-CRNN – comprises CNN and RNN models, which aims to derive effective representations of driving style. The idea is to use CNN to extract semantic driving patterns (e.g., a turn) from the input trajectory, and RNN to leverage sequential properties to encode dependencies between the patterns. Figure 4.2 illustrates the overall D-CRNN architecture, and we describe its major components as follows.

- **Input Layer**: The layer uses the aggregate feature map which is described in Section 4.5. This feature map is of size $7|\mathcal{F}| \times \big(2L_1/L_2\big)$, where $\mathcal{F}$ is a subset of basic features, and $2L_1/L_2$ is size of the time-axis.

- **Convolutional Component**: This component seeks to extract semantic driving patterns (e.g., a turn, braking event, etc.). Regarding the input which is represented as an image (or matrix), we believe that these semantic patterns could be characterized as distinguishing visual patterns. Hence, a convolutional network is an appropriate choice to extract such patterns. This component includes two convolutional layers, each followed by a max-pooling to downsample, and dropout (see [71]) on top of the pooling layer. The feature maps generated by the second pooling layer are stacked and used as input for the recurrent component. The first convolutional layer has 16 filters, each with a kernel of size $7|\mathcal{F}| \times 5$, which means convolution only on time axis. The second convolutional layer also has 16 filters, each with a kernel of size 3×3, and both convolutional layers use stride of size 1. The pooling operations in both layers are performed on feature axis, each with pool size 8×1 and stride 1. By using zero-padding, we maintain the size of the time-axis the same across convolutional and pooling layers. Thus, the output of the second max-pooling layer after the stacking is of size $16|\mathcal{F}'| \times \big(2L_1/L_2\big)$, where $\mathcal{F}'$ is size of the feature axis after the last max-pooling, and $|\mathcal{F}'| < |\mathcal{F}|$. The activation function for

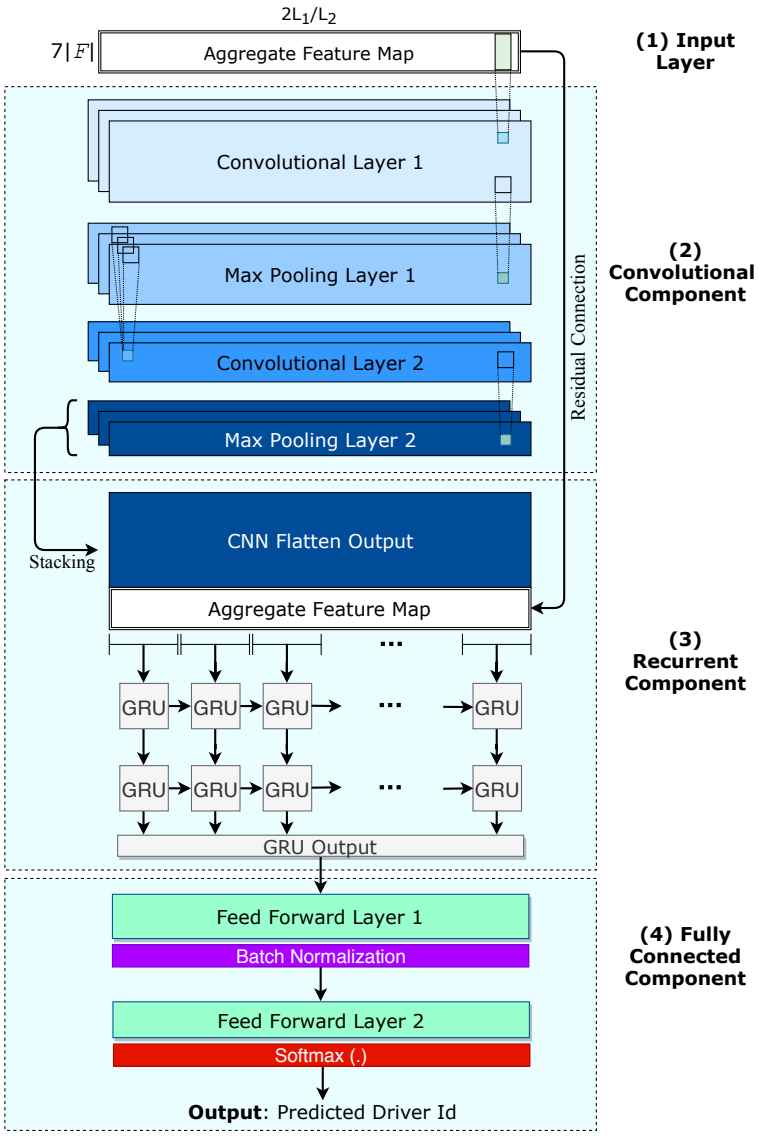


Figure 4.2: D-CRNN architecture overview: the model consists of four components: 1) Input Layer, 2) Convolutional (CNN), 3) Recurrent (RNN), and 4) Fully Connected (FC) component.

both convolutional layers is the rectified linear unit (or ReLU), and we use a dropout probability of 50% after each pooling layer.

- **Recurrent Component**: This component aims to leverage the sequential properties of the input over time to better encode dependencies between driving patterns. The stacked

output of the convolutional component is concatenated with the aggregate feature map (i.e., the input of the model) along their feature axis, to make a *residual connection* (see [96]), which is used by recurrent component as input. In this way, the input for the recurrent component is of size $\big(16|\mathcal{F}'| + 7|\mathcal{F}|\big) \times \big(2L_1/L_2\big)$. In the recurrent component, we have two layers of Gated Recurrent Unit (GRU) cells, stacked on each other. Each layer has 100 GRU cells with a dropout probability of 50%. Through our hyper-parameter tuning, we found GRU cells to provide the best results in comparison to the vanilla RNN and Long Short Term Memory (LSTM) cells.

- **Fully Connected Component**: The output of the second GRU layer is used as input to the two fully connected feed-forward layers, where we have batch normalization [78] and dropout (with probability 50%) after the first fully connected layer. The first layer has 100 hidden neurons and uses *sigmoid* as its activation function. The output of the second fully connected layer provides probability values for different class labels (i.e., driver ids) and a softmax is used to make the final prediction.

The usage of dropout is to prevent overfitting [71], especially when we have a deep network, which has proven to be effective in many deep learning applications [49]. Residual connection helps to better train a deep network [96], and we found it beneficial in our application. Assuming that the convolutional component extracts semantic patterns of trajectory, the usage of residual connection provides the context for such patterns and further helps the recurrent component to identify sequential dependencies between the patterns and extract high-level driving style information. Moreover, the usage of batch normalization is to speed up the training process [78]. Given a set of trajectories T as training data, we use back-propagation with *cross-entropy* loss function to train the network:

$$\mathcal{L}_{ce} = \sum_{m \in \mathcal{M}} \sum_{d \in \mathrm{D}} - \mathbb{1}(d, m) \times log\left(prob(\, d \,|\, m \,)\right) \tag{4.1}$$

Here $\mathcal{M}$ is the set of all feature maps extracted from trajectory set T; D is the set of all drivers whom trajectories T are taken from; $\mathbb{1}(d, m)$ is a binary indicator function which returns 1 if m belongs to d and 0 otherwise; and $prob(\, d \,|\, m \,)$ is the probability of predicting d as driver for m, which is the output of our D-CRNN model prior to applying softmax.

Table 4.1: Details on trajectory dataset, collected from Aug 2017 to Feb 2018. P50 is 50^{th} percentile.

Drivers	Trajectories	Total Duration	Total Distance	P50 of Duration	P50 of Distance
4,476	835,995	221,895 hr	11,174,400 km	11 min	6 km

Grid-search is employed to determine the best architecture settings (see Section 1.1 of Supplementary Material). It is worth noting that convolutional recurrent neural network is rather a new concept which has recently seen notable interests in literature [90, 89, 84, 108, 109, 91, 118, 142]. The major differences between our proposal and existing models include the usage of residual connection from input to the recurrent component, which helps the recurrent component to simultaneously utilize semantic patterns as well as low-level behavioral data; the usage of batch-normalization, after the first fully connected layer; and properties of different layers in each component.

4.6 Dataset

The data used in this research is a large, private dataset provided by an insurance company. Data was collected in a major city, from August 2017 to February 2018, using designated devices connected to the OBD-II port of vehicles which decode *CAN-bus* data. Additionally, each device encapsulates several sensors, including *gps*, *accelerometer*, and *magnetometer*. The data collection rate is one record per second (i.e., 1Hz). In total, we have about 4,500 drivers and 836K trajectories in our dataset. Table 4.1 provides some additional details about the data, including 50^{th} percentile (P50) on the duration and traveled distance of each trajectory. Before using the data, we performed several preprocessing, filtering, and sampling steps which we describe them next. Also, we elaborate on why we require each step.

4.6.1 Data Preprocessing

Several data preprocessing steps are performed to ensure the quality of data. To prevent memorization of origin and destination, we removed the first and the last two minutes

of each trajectory. Additionally, we set the minimum duration of a trajectory to be 10 minutes and the maximum to 30 minutes. This step will not limit the generality of methods, and it just ensures to have enough data for each trajectory and helps to avoid the long running-time based on extremely long trajectories[17]. Finally, we remove those trajectories with any missing attribute to preserve consistency in data.

4.6.2 Data Filtering and Sampling

One important step which is overlooked in literature is to filter input trajectories based on *spatial similarity*. We define the similarity between two trajectories T_1 and T_2 in terms of a spatial, point-wise matching function. We call two points p_i and p_j ($p_i \in T_1$ and $p_j \in T_2$) to be matched, if their haversine distance [153] is lower than a predefined threshold τ. To find similarity between two trajectories T_1 and T_2, we propose Algorithm 1. In this algorithm, p_{1i} and p_{2j} are two data points taken from trajectories T_1 and T_2, respectively, and we use distance threshold $\tau = 100$ (meters). Based on this algorithm, a data point from one of the trajectories can be matched with at most one data point from the other trajectory. Note that similarity measure, as defined here, is a spatial concept that discloses the spatial similarity of two trajectories. Having a dataset with (potential) high spatial similar trajectories for each driver is not an appropriate source of data to learn driving style; because the models can easily learn to memorize geolocation data. Instead, we perform similarity-aware sampling to create our train and test sets. Here we employ three sampling strategies, *Threshold-based*, *Stratified*, and *Random* sampling.

Threshold-Based Sampling

The idea is to create several subsets of the same size but using different *similarity thresholds* to sample trajectories per drivers. We use Algorithm 5 for this purpose. By the choice of maximum similarity threshold (ν) from the set $\{0.2, 0.25, 0.3\}$, we generate three sample sets, where the one which is produced by the lowest threshold (i.e., 0.2) is the most *strict* set. As shown by Table 4.2, the 90^{th} percentile (i.e., $P90$) on the pairwise similarity between trajectories taken from a driver significantly increases as we increase the similarity threshold.

[17]In our dataset, the 90^{th} percentile of trajectory duration is about 30 minutes.

Algorithm 4: Spatial Similarity Scoring

Input: Trajectories T_1 and T_2, Distance threshold τ

```
matched_set = []
for i = 1 to |T_1| do
    for j = 1 to |T_2| do
        dist = haversine(p_1i, p_2j)
        if (dist < τ) and (p_2j ∉ matched_set) then
            matched_set.append(p_2j)
            break
        end
    end
end
score = |matched_set| / min(|T_1|, |T_2|)
```

Output: score

Algorithm 5: Threshold-based Trajectory Sampling

Input: a dataset of drivers and trajectories (D, T), and similarity threshold ν

1 For each driver $d \in \mathrm{D}$, calculate similarity score between each pair of trajectories taken from d using Algorithm 1.

2 Sample a subset of trajectories for each driver $d \in \mathrm{D}$, which in that the maximum similarity between any pair of trajectories is lower than threshold ν. Call this set $(\mathrm{D}', \mathrm{T}')$, where $\mathrm{D}' \subset \mathrm{D}$ and $\mathrm{T}' \subset \mathrm{T}$.

3 Subset: randomly select 50 drivers in set D' which for them we have at least 50 trajectories in set T'.

Output: Subset

Stratified Sampling

A slightly different idea is to create sets with a larger number of drivers while keeping the distribution of pairwise similarity between trajectories the same. Here we employ Algorithm 6, to first create different buckets of trajectories for a driver (based on pairwise similarity), and then uniformly sample from each bucket. In terms of input, we set $N = 50$, draw M from set $\{50, 100, 150, 200\}$, and use threshold set $\{0.2, 0.25, 0.3\}$. Therefore, we generate four different sets as shown in Table 4.2. As one could expect, the 90^{th} percentile

of pairwise trajectory similarity values are almost the same across different sample sets created by this sampling method.

Algorithm 6: Stratified Trajectory Sampling

Input: a dataset of drivers and trajectories (D, T), required number of trajectories N, required number of drivers M, and similarity threshold values $\{\nu_1, \nu_2, \ldots, \nu_m\}$.

1 For each driver $d \in$ D, calculate similarity score between each pair of trajectories of driver d using Algorithm 1.

2 For each driver $d \in$ D and for each trajectory T of this driver, calculate the average pairwise similarity value.

3 For each driver $d \in$ D, create m subsets of her trajectories, where the i^{th} subset contains trajectories with their average similarity bounded by interval $[\nu_{i-1}, \nu_i)$. Assume $\nu_0 = 0$.

4 Select a subset D$'$ of drivers D that for them we have at least $\frac{\mathrm{N}}{m}$ trajectories in each subset.

5 Subset: randomly select M drivers from D$'$, and for each driver randomly select $\frac{\mathrm{N}}{m}$ trajectories from each subset.

Output: Subset

Random Sampling

To show the impact of high spatial similarity in trajectory data for the task of driver identification, we create several random sample sets. Here, using the original dataset, we first sample four groups of drivers of size 50, 100, 150, and 200. Then, for each group, we randomly sample 50 trajectories for each driver. The 90^{th} percentile of pairwise trajectory similarity for these sets is assumed to be the same as the original data which is about 0.607.

In total, we created 11 sample sets; notation and a short description for each set are provided in Table 4.2.

4.7 Result and Discussion

In this section, we present experimental settings and results. We start with a description of baseline methods, then describe how to perform trajectory-level prediction. Next, we

Table 4.2: Sampled trajectory datasets, using *Threshold-based*, *Stratified*, and *Random* sampling strategies. P90 is the 90^{th} percentile on pairwise similarity between trajectories.

Notation	Drivers	P90 of Similarity	Max Similarity	Strategy
Tb–50_0.2	50	0.116	0.2	Threshold-based
Tb–50_0.25	50	0.166	0.25	Threshold-based
Tb–50_0.3	50	0.212	0.3	Threshold-based
St–50	50	0.177	0.3	Stratified
St–100	100	0.171	0.3	Stratified
St–150	150	0.168	0.3	Stratified
St–200	200	0.167	0.3	Stratified
Rd–50	50	0.607	1.0	Random
Rd–100	100	0.607	1.0	Random
Rd–150	150	0.607	1.0	Random
Rd–200	200	0.607	1.0	Random

continue with feature analysis, followed by comparing different architecture and models, using the best combination of basic features. Finally, we study learned driving style based on several use-cases. All implementations are in Python using Tensorflow [77], Theano [104], and scikit-learn [45] libraries; and experiments were run on Ohio Supercomputer nodes [4].

We use *accuracy* as our evaluation metric. Given a set of trajectories $\{T_1, T_2, \ldots, T_n\}$, where T_i is taken from driver d_i, and $prediction(.)$ is a function which predicts identity of driver for T_i, we define "accuracy" as:

$$Accuracy = \frac{\sum_{i=1}^{n} \mathbb{1}_{(prediction(T_i)=d_i)}(T_i)}{n} \tag{4.2}$$

where $\mathbb{1}$ represents the indicator function.

4.8 Baseline Models

As baseline state-of-the-art models, we use several existing architectures to re-train their model and report their prediction results. Among the selected models, three of them are deep-neural network based architectures and one of them is a Gradient Boosting Decision Tree (GBDT) model. We briefly describe each model in following sub-sections and provide the reasoning on why we chose them for comparison.

CNN Model

This is a convolutional neural network model proposed by Dong et al. [93] for learning driving style. Having the input as aggregate feature map (see Section 4.5), the model employs three convolutional layers on top of the input, each followed by a max-pooling layer. Here, both convolution and pooling operations are over the time axis, hence, both are 1-Dimensional operations. On top of the last max-pooling layer, there are three fully-connected (or dense) layers of size 128, 128, and *number of drivers*, respectively.

RNN Model

This is a recurrent neural network model proposed by Dong et al. [93]. The input of the model is the aggregate feature map, where this model utilizes that as a sequential input over time. There are 2 recurrent layers, each with 100 LSTM cells. On top of the output of the second recurrent layer, there are two fully connected layers, where the first layer has 100 neurons and the second layer has as many neurons as the number of drivers which are our labels [18].

4.8.1 VRAE Model

The Variational Recurrent Auto-Encoder (VRAE), proposed by Fabius and Amersfoort [66], is a generative model, used for learning stochastic latent representations of the input sequences. Being stochastic, the model maps an input to a different latent representation each time. This is a useful property, since it allows the model to associate the input sequence with not just one point, but with nearby points as well. For our task, we treat the raw trajectory data as the sequential input to the VRAE model. However, we train our model to not just be a good generator of trajectories, but learn representations which are highly predictive of the drivers as well. This is done by employing a loss function consisting of two parts - the loss due to likelihood error and the loss due to driver misclassification:

$$\mathcal{L}_{total} = \lambda \mathcal{L}_{LL} + (1 - \lambda)\mathcal{L}_{pred} \tag{4.3}$$

Here, $\mathcal{L}_{LL}$ is the likelihood loss defined in the same way as in [66], and $\mathcal{L}_{pred}$ is the cross-entropy loss on misclassification of the driver of the trajectory. λ is a hyper-parameter

[18] In [93] authors suggested to use vanilla recurrent units, with ReLU as activation function, and identity matrix to initialize the weight matrix, which provides comparable results to LSTM.

which controls the relative importance given to each of the loss terms. Let y_i be the one hot representation of the driver's identity, we define the cross-entropy loss as:

$$\mathcal{L}_{pred} = \sum_{i=1}^{\text{num\_drivers}} -y_i log(\hat{y_i}) \tag{4.4}$$

where $\hat{y_i}$, the vector representing model's driver prediction, is calculated as follows:

$$\hat{y_i} = \text{softmax}(W_{driver} h_{enc} + b_{driver}) \tag{4.5}$$

The model employs a GRU encoder to transform the sequential input trajectory into an encoded representation h_{enc}. This encoded representation is used for sampling the stochastic latent representation, z, as follows:

$$\mu_z = W_\mu^t h_{enc} + b_\mu$$
$$log(\sigma_z^2) = W_\sigma^t h_{enc} + b_\sigma$$
$$z \sim N(\mu_z, \sigma_z^2)$$

The decoder is a GRU as well, whose initial state h_{dec} is generated by applying a linear transformation followed by $tanh$ activation on the latent representation z:

$$h_{dec} = tanh(W_{dec} z + b_{dec}) \tag{4.6}$$

The input for VRAE model is the aggregate feature map.

4.8.2 GBDT Model

This model is known to be effective for the task of driver identification [79]. Here we use the set of hand-crafted features as input, which includes *global* and *local* features [93]. Global features include trip duration (in seconds), trip length (distance), average speed, area of the minimal rectangle containing the trip shape, length of the two edges of the minimal rectangle; and a set of 35 values as result of applying seven statistics (i.e., *mean*, *minimum*, *maximum*, 25^{th}, 50^{th}, 75^{th} *percentiles*, and *standard deviation*) on the following basic features: GPS_Speed, GPS_Speed_Change[19], GPS_Acceleration, GPS_Acceleration_Change, and Angular_Speed. For the local features, we first calculate the moving angle of each point

[19]Change with respect to previous observation.

of the trajectory (a value between 0 and 180). Then, by defining 8 bins $[0, 10)$, $[10, 20)$, $[20, 30)$, $[30, 45)$, $[45, 60)$, $[60, 90)$, $[90, 120)$, and $[120, 180]$ for the angle value, the seven statistics for each basic feature are calculated for the data points in each bin. In total, we have 321 features to represent each trajectory. A slightly different idea is to replace the basic features with Speed, Acceleration, Acceleration_Change, RPM, RPM_Change, and Angular_Speed. By keeping the rest of the feature encoding process the same, we would have 384 features to encode a trajectory. The model based on the original set of basic features is termed *GBDT-original*, and the one based on the modified set of basic features is termed *GBDT-modified*.

The model used in here is GBDT, which uses the local and global features to predict the driver. The model based on the original set of features is termed *GBDT-V1*, and the one based on the modified set of features is termed *GBDT-V2*.

4.8.3 Trajectory Level Prediction

All models, except GBDT models, perform segment-level prediction. To obtain trajectory-level prediction, we first obtain the *probability vector* for each segment. The probability vector for segment j of trajectory T can be represented as $\langle \rho_{j1}, \rho_{j2}, \ldots, \rho_{jm} \rangle$, where m is the number of drivers in a set (i.e., potential labels). This vector is the output of a model before the last softmax layer. Then, for a trajectory T which has k segments, we create a trajectory-level average probability vector as follows:

$$Prob\_Vector(T) = \frac{1}{k} \sum_{j=1}^{k} \langle \rho_{j1}, \rho_{j2}, \ldots, \rho_{jm} \rangle \tag{4.7}$$

Then, $softmax(Prob\_Vector(T))$ provides the trajectory-level prediction. Note that in our experiments we only report trajectory-level accuracy, and omit the segment-level results in favor of space.

4.8.4 Feature Analysis

This experiment explores the contribution of the basic features, extracted from trajectory data to the learning of driving style. In total, we have 10 basic features to describe a data point (see Section 4.5). We leverage two models – RNN-model [93] and our D-CRNN architecture. For training of both models, we use mini-batches of size 256; 85% of trajectories

taken from each driver in train and 15% for test; and Root Mean Square Propagation optimizer[53] (RMSProp for short) with initial learning rate $5e-5$, momentum 0.9, and epsilon $1e-6$ as optimizer function. Here we compare different methods based on *Accuracy* (see Equation 4.2). In terms of data, we used data sample sets Tb–50_0.2, St–200, Rd–50, and Rd-200. Table 4.3 provides the results of this experiment, using 13 different subsets of basic features, including the entire set, and we summarize our observations as follows.

- *Best Subset*: The best combination of features is found to be "Speed", "Acceleration", and "RPM", across both models and all four datasets. Even using the entire set of basic features did not result in better accuracy. Based on this observation, we argue that having these three features is enough to learn the driving style of an individual for use in driver identification.

- *Most Effective Feature*: The most effective feature is found to be RPM. While this observation may be related to the uniqueness of RPM observations for different vehicle brands, we reject this hypothesis for two reasons. First, using the set "Tb–50_0.2" we have the least similarity among trajectories of a driver, which means that it is very unlikely to observe the same series of RPM values for different trajectories. Second, in set "St–200" we used a much larger set of vehicles to decrease the possibility of having vehicles with unique characteristics. *We also note that RPM is the outcome of an essential driving behavior (i.e., pressing the gas pedal), therefore we expect it to be a robust predictor of driver identity.*

- *GPS Related Features*: This experiment revealed a significant difference between Speed and Acceleration generated based on GPS coordinates, versus the ones reported directly by the vehicle using CAN-bus. We believe that this observation is due to calculation errors (e.g., distance) when using GPS coordinates.

- *Memorization versus Learning Driving Style*: The low prediction accuracy for a feature like "Head" is an indicator of the sampling quality to avoid memorization. If prediction by heading was high, as it is by random sample sets Rd–50 and Rd-200, this would have potentially indicated that the model is doing spatial memorization instead of learning driving style. Since for a given driver, we could have only a few directions based on their commute patterns. Likewise, a comparison between results obtained using random sets

Table 4.3: Driver prediction based on different combinations of basic features using RNN-model and D-CRNN, tested on data sample sets *Tb–50_0.2*, *Rd–50*, *St–200*, and *Rd–200*. Results are reported based on Accuracy metric (see Equation 4.2). RPM is the outcome of an essential driving behavior, therefore we expect it to be a robust predictor of driver identity.

	Tb–50_0.2		**Rd–50**		**St–200**		**Rd–200**	
Basic Features	RNN-mdl	D-CRNN	RNN-mdl	D-CRNN	RNN-mdl	D-CRNN	RNN-mdl	D-CRNN
Entire Feature Set	46.93%	49.07%	66.58%	70.96%	52.08%	46.27%	68.25%	61.77%
RPM	40.50%	54.75%	59.52%	75.00%	50.27%	54.37%	64.31%	65.34%
Head	2.22%	2.49%	26.72%	23.66%	1.14%	1.00%	20.00%	14.84%
Speed	8.10%	20.90%	21.23%	34.92%	12.99%	16.68%	24.70%	31.62%
RPM and Head	28.57%	45.61%	63.69%	70.89%	45.31%	47.37%	62.20%	62.02%
RPM, Head, AclX, AclY, and AclZ	30.22%	40.38%	56.92%	66.84%	55.34%	51.17%	66.02%	61.83%
GPS_Speed, GPS_Accel, and Angular_Speed	6.98%	8.66%	15.05%	15.86%	2.40%	4.08%	10.46%	12.26%
RPM, GPS_Speed, and GPS_Accel	25.98%	45.57%	55.90%	71.40%	46.50%	48.28%	59.18%	61.02%
Speed, Accel, and Angular_Speed	25.90%	27.27%	27.25%	32.39%	15.11%	17.48%	28.07%	28.24%
Speed, Accel, RPM, Angular_Speed, and Head	46.77%	50.00%	68.82%	73.39%	54.35%	52.09%	66.45%	63.81%
Speed, Accel, and RPM	**52.29**%	**56.06**%	**73.10**%	**75.63**%	**57.72**%	**58.49**%	**68.66**%	**70.90**%
Speed, Accel, AclX, AclY, and AclZ	25.78%	27.76%	42.82%	50.00%	23.94%	28.39%	43.41%	43.14%
AclX, AclY, and AclZ	8.18%	14.78%	21.64%	34.83%	13.83%	22.33%	31.26%	37.52%

and other sample sets further validates our initial hypothesis on the significant impact of spatial information.

We also note that D-CRNN provides better results for a majority of cases in Table 4.3.

4.8.5 Data–Similarity Effect

This experiment studies the impact of similarity among trajectories, and to assess the ability of different models to learn driving style instead of spatial memorization. Here, we compare different models using following sets: "Tb–50_0.2", "Tb–50_0.25", "Tb–50_0.3", and "Rd–50". We use speed, acceleration, and RPM as basic features. For RNN-model and CNN-model, we use the same hyper-parameters as suggested in [93]. For VRAE we use

Table 4.4: Studying the impact of Spatial Similarity on driving style learning and driver identity prediction.

Model	Tb–50_0.2	Tb–50_0.25	Tb–50_0.3	Rd–50
GBDT-original [79]	5.70%	3.09%	6.50%	25.13%
GBDT-modified	25.65%	32.87%	29.54%	49.24%
CNN-model [93]	39.81%	44.48%	46.88%	59.90%
RNN-model [93]	50.69%	57.02%	58.36%	70.73%
VRAE [66]	26.50%	42.20%	54.50%	63.67%
D-CRNN	**59.50**%	**64.42**%	**66.76**%	**78.00**%
D-CRNN-W/O-BR	54.40%	59.93%	63.41%	75.63%

$\lambda = 0.3$, $|h_{enc}| = |h_{dec}| = |z| = 128$ (see Section 2 of Supplementary Material). As before, we use 85% of trajectories taken from each driver in train and 15% for the test[20], the size of mini-batches is set to 256, and RMSProp with the same setting as the previous section is used as the optimizer. Table 4.4 presents the results of this experiment in terms of average accuracy over three runs. Here, *D-CRNN-W/O-BR* shows a variation of D-CRNN without batch-normalization and residual connection.

Based on Table 4.4, we make the following observations:

- *Best Model*: The best prediction results are obtained by D-CRNN. In comparison to the state-of-the-art, our proposed architecture achieves about 8.8% improvement based on the set of trajectories with the least similarity between them (i.e., Tb–50_0.2).

- *Learning versus Memorization*: Increasing the similarity threshold simplifies the driver identification task, as we include more hints in terms of spatial similarity between trajectories. This scenario becomes clearer when using the random sample set, which on-average yields 20.50% difference in accuracy when compared to the set "Tb–50_0.2". This is an important observation because it justifies why data preprocessing is required to have a sound comparison framework. Also, we claim the consistent trend of better results shows the superiority of our model to better learn the driving style.

[20]The same train and test sets are used for all the models.

- *Non-deep Learning Models*: Given the results, we conclude that the modified set of basic feature provides significant improvement in comparison to the case of using the original features. However, the best accuracy obtained by a GBDT model is still lower than the worst accuracy by a deep-neural-network-based architecture (i.e., VRAE model).

- *Results with VRAE*: VRAE generates stochastic latent vectors, which is the reason why it is unable to capture the high-level structure in trajectories as well as other deep learning models. Consequently, the VRAE model suffers in driver prediction.

Our experimental results also reveal the importance of batch normalization and residual connection.

4.8.6 Data–Size Effect

Next, we use sample sets of larger sizes, in terms of the number of drivers. Using the same hyper-parameters and setting as before, the results of this experiment are shown in Table 4.5, in terms of accuracy based on three experiments for each model and dataset. Note that we used speed, acceleration, and RPM as basic features. We make the following observations:

- *Best Model*: D-CRNN is the best model when we use more drivers. However, as we increase the number of drivers, our results become closer to RNN-model.

- *Effect of Data Size*: By increasing the number of drivers, we see a decreasing trend of accuracy. However, the slope of drop for some of the models is slower than the others, which demonstrates the ability of those models to better learn driving style, including D-CRNN, RNN-model, and CNN-model.

- *Deep versus Non-deep*: Regarding the results, as we increase the number of drivers, we see more struggle by non-deep methods such as GBDT. However, most of the deep-models successfully handle more drivers and provide noticeably better results.

Note that comparing the results of random samples with the sets with uniform similarity distributions, once again, reveals the impact of spatial similarity which leads to significant overestimation of the capability of different models to learn driving style.

Table 4.5: Studying the effect of Data Size on driving style learning and driver identity prediction.

Model	St–50	Rd–50	St–100	Rd–100	St–150	Rd–150	St–200	Rd–200
GBDT-original [79]	6.58%	25.13%	2.23%	23.24%	2.61%	20.67%	2.18%	19.53%
GBDT-modified	30.26%	49.24%	20.18%	36.62%	16.28%	31.84%	13.37%	29.81%
CNN-model [93]	42.37%	59.90%	44.17%	52.66%	39.14%	53.27%	40.27%	53.49%
RNN-model [93]	58.07%	70.73%	57.23%	67.48%	55.29%	68.48%	57.34%	67.32%
VRAE [66]	48.58%	63.67%	9.33%	44.80%	8.00%	11.50%	3.00%	11.00%
D-CRNN	**65.35**%	**78.00**%	**61.64**%	**73.56**%	**56.92**%	**71.89**%	**57.50**%	**69.66**%
D-CRNN-W/O-BR	61.58%	75.63%	57.97%	68.42%	54.13%	66.75%	54.51%	68.15%

4.8.7 Real-World Scenario

In a real-world application, we might perform offline pre-processing to filter trajectory data prior to train and apply the model on random test data to identify the identity of drivers in real-time. For this scenario, we modify the datasets used in Section 4.8.5 as follows: given a threshold-based sample set, we keep the train data, but replace the test data with a random sample of trajectories taken from each driver. Again, for each driver, we have 50 trajectories, where 85% of them are used for train and 15% for the test. Using the same models and hyper-parameters, after running each model for three times, the average accuracy results are shown in Table 4.6. As one might expect, using random test sets, we obtained significantly better accuracy results (about 6% on average). Such an outcome can be attributed to simplifying the testing condition by increasing the chance of having spatially-similar trajectories in train and test sets. Another interesting observation is the proximity of results when using different threshold-based training sample sets using RNN-model and D-CRNN. This shows the ability of these models to better derive style information instead of memorizing location data, which leads to stable prediction results.

4.8.8 Driving Style Representation

The main objective of this work is to create models for driver identification by learning driving style. To further assess the accomplishment of this objective, in this section we provide some visualizations using the learned latent representation of trajectories for several drivers. These trajectories are those which we used in the test set. To make the task and

Table 4.6: Studying the effect of training on threshold-based sample set and testing on random sets.

Model	**Tb–50_0.2** (Random test)	**Tb–50_0.25** (Random test)	**Tb–50_0.3** (Random test)
GBDT-original [79]	6.25%	8.75%	10.00%
GBDT-modified	28.25%	32.50%	36.00%
CNN-model [93]	43.92%	52.00%	54.13%
RNN-model [93]	59.75%	60.67%	64.33%
D-CRNN	**73.25**%	**76.17**%	**75.75**%

evaluation more rigorous, we choose set "Tb-50_0.2" which has the least spatially similar trajectories for each driver. Here we compare two models, D-CRNN and RNN-model. In terms of latent representation, we use the output of the first fully connected layer for both models. As such output is for a segment of a trajectory, thus, given a trajectory, we use the *average vector* of representation of its segments as trajectory latent representation. For both models, this representation is a 100-dimensional vector. We use t-SNE method [34] to perform feature selection and show the latent representations in a 2-dimensional space. We randomly sample two sets of 10 drivers. Figure 4.3 illustrates the results of this experiment, where the latent representation of trajectories based on D-CRNN and RNN models is shown for both sample sets. Here, the expected outcome is to see trajectories taken from different drivers to be far from each other, while trajectories taken from the same driver to be sufficiently close. Given the results, both methods provide reasonable latent representations, while D-CRNN provides better separation for multiple cases. For example, D-CRNN provides better separation for trajectories taken from drivers D-1, D-4, D-5, D-8, D-10, D-11, D-13, and D-18. Note that closeness of the trajectories based on their latent representation cannot be a result of spatial similarity, as we use the set of least similar trajectories for each driver. Instead, this is a demonstration of the drivers' fingerprint (or style) which discriminates them from each other.

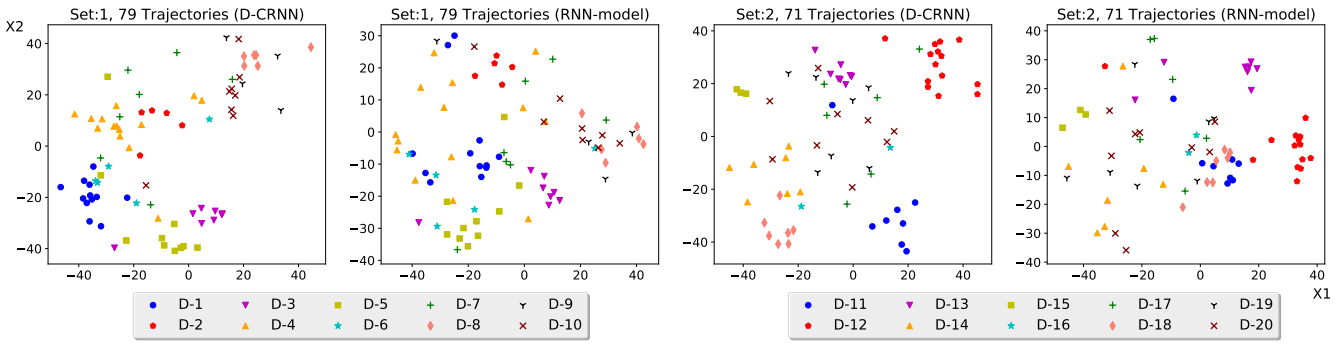


Figure 4.3: Demonstration of learnt driving style by RNN-model and D-CRNN, using two random sample sets of 10 drivers. Original latent vectors are 100-D, and we used t-SNE [34] for feature selection to represent them in a 2-D space.

4.9 Conclusion

In this chapter, we present a deep-neural-network architecture to learn driving style from trajectory data. The proposed model is inspired by the idea of convolutional recurrent neural networks, to empower the model by extracting semantic patterns from trajectories, and by encoding the sequential dependencies between the patterns. We also propose to sample dissimilar trajectories on a large scale to prevent spatial memorization, which could mislead the task and invalidate the results. Our analyses and results demonstrate the superiority of the proposed architecture for driver identification in comparison to state-of-the-art techniques, based on different test scenarios and datasets. We conclude that using deep-learning models for the task of learning driver style yields tremendous improvement in comparison to the non-deep-learning solutions, especially when using a proper combination of deep-neural-network components. Moreover, geolocation bias is an important challenge when we design frameworks based on data that has spatial information, while utilizing such data may not be always an objective. Finally, we note that a subset of data elements collected from the CAN-bus is the best combination to represent driving style.

Chapter 5: Context-aware Traffic Accident Prediction

In this chapter, we propose a context-aware solution for traffic accident prediction. Traffic accidents are important public safety challenges and explicit indicators of driving risk. Accident prediction has been a topic of much research over the past few decades. Shortcomings of existing studies include using small-scale datasets with limited coverage, being dependent on an extensive set of data, and lacking applicability for real-time purposes. To address these challenges, we propose a new solution for real-time traffic accident prediction using easy-to-obtain but sparse data. Our solution relies on a deep-neural-network model (which we have named *DAP*, for Deep Accident Prediction), which uses a variety of contextual data such as *traffic events*, *weather data*, *points-of-interest*, and *time*. DAP incorporates a recurrent component (for time-sensitive data), two fully connected components (for time-insensitive data), and a trainable embedding component (to capture spatial heterogeneity). To fill the data gap, we have - through a comprehensive process of data collection, integration, and augmentation - created a large-scale publicly available database of accident information named *US-Accidents*. By employing the US-Accidents dataset and through an extensive set of experiments across several large cities, we have evaluated our proposal against several baselines on the important task of macro-level driving risk prediction. Our analysis and results show significant improvements in the prediction of rare accident events. Further, we have shown the impact of traffic information, time, and points-of-interest data for real-time accident prediction.

5.1 Motivation

Reducing traffic accidents is an important public safety challenge around the world. A global status report on traffic safety [83] notes that there were 1.25 million traffic deaths in 2013 alone, with deaths increasing in 68 countries compared to 2010. Accident prediction

is important for optimizing public transportation, enabling safer routes, and improving transportation infrastructure, all to make roads safer. Given its significance, accident analysis and prediction have been topics of much research in the past few decades. The major related research categories include analyzing the impact of environmental stimuli (e.g., road network properties, weather, and traffic) on traffic accident occurrence patterns [16, 67, 103], predicting the frequency of accidents within a geographical region [21, 30, 114, 132, 136], and predicting the risk of accidents [119, 120, 82, 92].

The main shortcomings of existing studies are the following: employing small-scale datasets with limited coverage (e.g., a small number of road-segments, or just one city) [20, 21, 30, 82, 119]; being dependent on a wide range of data that may not be available for all regions (e.g., satellite imagery, traffic volume, and properties of road-network) [120, 114, 136]; lacking applicability for real-time applications regarding the modeling constraints and prerequisites (e.g., prediction for longer time intervals such as one day or one week, or requiring extensive set of data) [30, 114, 132, 136]; and employing oversimplified methods for traffic accident prediction [30, 82, 130].

To address these challenges and provide a reasonable solution for real-time traffic accident prediction, we propose the aforementioned *DAP*, a deep-neural-network-based accident prediction model. DAP uses a variety of contextual data including *traffic events* (e.g., congestion, construction, and road hazards), *weather* (e.g., temperature, visibility, and wind speed), *points-of-interest* (e.g., traffic signal, stop sign, and junction), and *time* (e.g., day of week, hour of day, and period of day) to provide real-time prediction for a geographical region of reasonable size (i.e., a square of size $5km \times 5km$ on a map) and during a fine-grained time period (i.e., a 15-minute interval). To our knowledge, this is the first work that has employed traffic events and points-of-interest data for accident prediction. For *time-sensitive* data (e.g., traffic, weather, and time data), DAP employs a recurrent component with Long Short Term Memory (LSTM) cells. For *time-insensitive* data (e.g., points-of-interest), DAP employs feed-forward neural-network layers. Further, to better capture spatial heterogeneity, which is effective for accident prediction [120], DAP employs trainable latent representation for each geographical region to encode essential spatiotemporal information.

In order to mitigate the impact of data size on analysis and prediction, we use *US-Accidents*, a new dataset including about 2.25 million traffic accidents recorded in the

contiguous United States[21] between February 2016 and March 2019. US-Accidents offers a comprehensive set of attributes to describe each accident including *location data*, *time data*, *natural language description of event*, *weather data*, *period-of-day information*[22], and *relevant points-of-interest data*. Importantly, we also present our *process* for creating the above dataset from streaming traffic reports and heterogeneous contextual data (weather, points-of-interests, etc.), so that the community can validate it and with the belief that this process can serve as a model for dataset creation. Our analyses of the US-Accidents dataset demonstrated that about 40% of accidents took place on or near high-speed roadways (highways, interstates, etc.) and about 32% took place on or near local roads (streets, avenues, etc.). We also derived various insights with respect to the correlation of accidents with time, points-of-interest, and weather conditions.

Using US-Accidents, and through extensive experiments across several large cities, we compared our proposal against several neural-network-based and traditional machine-learning models (such as logistic regression and gradient boosting classifier). Our analysis and results show the superiority of our model in terms of improvement of $f1$-*score* for the case of positive examples (i.e., cases labeled as accidents), by about 16% in comparison to the best traditional model, and about 7% in comparison to the best neural-network-based model. When considering both positive and negative cases (negative cases are labeled as non-accidents, which are the majority), our proposal achieves results comparable to the best baselines. Nevertheless, we note that positive cases are far more important, given their rare nature, and need to be predicted properly. Further, we assesses the ability of different categories of attributes for real-time traffic accident prediction using multiple testing scenarios. Our findings indicate the importance of time, points-of-interest, and traffic data for this task.

The main contributions of this chapter are therefore as follows.

- We develop a new methodology for heterogeneous data collection, cleansing, and augmentation to prepare a unique, large-scale dataset of traffic accidents. This dataset contains 2.25 million traffic accidents recorded in the contiguous United States over three years. The dataset is available publicly at `https://smoosavi.org/datasets/us_accidents`.

[21]The contiguous United States excludes Alaska and Hawaii and considers District of Columbia (DC) as a separate state.

[22]Period-of-day is associated with daylight, thus it is represented as "day" or "night".

- We compile a variety of insights gleaned through analyses of accident hot-spot locations, time, weather, and points-of-interest correlations with the accident dataset. These insights may be used directly for applications such as urban planning, exploring flaws in transportation infrastructure design, traffic control and prediction, and personalized insurance.
- We propose a new deep-neural-network-based solution for traffic accident prediction using heterogeneous sparse data. To the best of our knowledge, this is the first work that uses information from *traffic flow*, fused with other available sources of contextual data such as "weather" and "points-of-interest," to perform accident prediction. Furthermore, our methodology predicts future accidents at the fine-grained time interval of 15 minutes.

5.2 Definitions and Problem Statement

Definition 5.2.1 (Traffic Event) *We define a traffic event e by* $e = \langle lat, lng, time, type, desc \rangle$*, where lat and lng represent the GPS coordinates, type is a categorical classification of the event, and desc provides a natural language description of the event. A traffic event is one of the following types:* accident*,* broken-vehicle*,* congestion*,* construction*,* event*,* lane-blocked*, and* flow-incident*. Table 5.1 describes these events.*

Table 5.1: Definition of Traffic Events.

Type	Description
Accident	A collision event which may involve one or more vehicles.
Broken-vehicle	Refers to the situation when there is one (or more) disabled vehicle(s) in a road.
Congestion	Refers to the situation when the speed of traffic is slower than the expected speed or speed-limit.
Construction	Refers to maintenance project on a road.
Event	Situations such as *sports event*, *demonstrations*, or *concerts*, that could potentially impact traffic flow.
Lane-blocked	Refers to the cases when we have blocked lane(s) due to traffic or weather condition.
Flow-incident	Refers to all other types of traffic events. Examples are *broken traffic light* and *animal in the road.*

Definition 5.2.2 (Weather Observation Record) *A weather observation w is defined by* $w = \langle lat, lng, time, temperature, humidity, pressure, visibility, wind\text{-}speed, precip, rain, snow, fog, hail \rangle$. *Here lat and lng represent the GPS coordinates of the weather station which reported w;* precip *is the precipitation amount (if any); and rain, snow, fog, and hail are binary indicators of these events.*

Definition 5.2.3 (Point-of-Interest) *A point-of-interest p is defined by* $p = \langle lat, lng, type \rangle$. *Here, lat and lng show the GPS latitude and longitude coordinates, and available types for p are described in Table 5.2. Note that several of definitions in this table are adopted from* `https://wiki.openstreetmap.org`.

Table 5.2: Definition of Point-Of-Interest (POI) annotation tags based on Open Street Map (OSM).

Type	Description
Amenity	Refers to particular places such as restaurant, library, college, bar, etc.
Bump	Refers to speed bump or hump to reduce the speed.
Crossing	Refers to any crossing across roads for pedestrians, cyclists, etc.
Give-way	A sign on road which shows priority of passing.
Junction	Refers to any highway ramp, exit, or entrance.
No-exit	Indicates there is no possibility to travel further by any transport mode along a formal path or route.
Railway	Indicates the presence of railways.
Roundabout	Refers to a circular road junction.
Station	Refers to public transportation station (bus, metro, etc.).
Stop	Refers to stop sign.
Traffic Calming	Refers to any means for slowing down traffic speed.
Traffic Signal	Refers to traffic signal on intersections.
Turning Loop	Indicates a widened area of a highway with a non-traversable island for turning around.

Definition 5.2.4 (Geographical Region) *We define a geographical region r as a square of size $l \times l$ over the map of a city. The choice of l is related to the application domain, and in this work, we set $l = 5km$.*

Given the preliminaries, we formulate the problem as follows:

Given:

- A spatial grid $\mathrm{R} = \{r_1, r_2, \ldots, r_n\}$, where each $r \in \mathrm{R}$ is a geographical region of size $5km \times 5km$.
- A set of fixed-length time intervals $\mathrm{T} = \{t_1, t_2, \ldots, t_m\}$, where we set $|t| = 15$ *minutes*, for $t \in \mathrm{T}$.
- A database of traffic events $E_r = \{e_1, e_2, \ldots\}$ for each geographical region $r \in \mathrm{R}$.
- A database of weather observation records $W_r = \{w_1, w_2, \ldots\}$ for each geographical region $r \in \mathrm{R}$.
- A database of points of interest $P_r = \{p_1, p_2, \ldots\}$ for each geographical region $r \in \mathrm{R}$.

Create:

- A representation F_{rt} for a region $r \in \mathrm{R}$ during a time interval $t \in \mathrm{T}$, using E_r, W_r, and P_r.
- A binary label L_{rt} for F_{rt}, where 1 indicates at least one traffic accident happened during t in region r, and 0 otherwise.

Find:

- A model $\mathcal{M}$ to predict L_{rt} using $\langle F_{rt_{i-8}}, F_{rt_{i-7}}, \ldots, F_{rt_{i-1}} \rangle$, which means predicting the label of current time interval using observations from the last 8 time intervals to

Objective:

- Minimize the prediction error.

5.3 Related Work

Accident analysis and prediction has been the topic of much research during the past few decades, where we study three categories of these work as follows.

Analysis of Environmental Stimuli on Accidents. This category of work investigates the impact of environmental stimuli (e.g., weather, traffic flow, and properties of road-network) on possibility or severity of traffic accidents. Studying the impact of weather

factors (e.g., precipitation) on road accidents [16, 67, 103, 116]; applying data mining techniques to extract association rules to perform causality analysis [81, 54]; and statistical analysis of unobserved heterogeneity to explore the impact of unavailable variables (e.g., missing data) on severity of traffic accidents [102] are some examples of this category. These studies usually provide significant insights, however, may not be directly used for real-time prediction and planning.

Accident Frequency Prediction. Prediction of the expected number of traffic accidents for a specific road-segment or geographical region is the target of this group of studies [21]. Early work in this area by Chang et al. [20] used information such as road geometry, annual average daily traffic (AADT), and weather data to predict the frequency of accidents for a highway using a neural network model. Caliendo et al. [30] used a set of road-related attributes such as length, curvature, AADT, sight distance, and presence of junction to predict the frequency of accidents. The usage of satellite imagery to predict the frequency of accidents by a convolutional neural network model using large scale accident and imagery data was proposed by Najjar et al. [114]. Further, Ren et al. [132] recently used a Long Short Term Memory (LSTM) model to predict the frequency of accidents, given the history of the past 100 hours, for grid cells of size $1km \times 1km$. Similarly, Chen et al. [122] proposed to use a stacked denoising convolutional autoencoder model to predict the frequency of accidents for grid cells using traffic flow (collected using plate recognition systems), past traffic accidents, and time data. Yuan et al. [136] proposed hetero-ConvLSTM to predict the frequency of traffic accidents using several sources of environmental data such as traffic volume, road condition, rainfall, temperature, and satellite images. They evaluated their model using large-scale data of traffic accidents from the state of Iowa, performed predictions for grid cells of size $5km \times 5km$, and showed the importance of capturing spatial heterogeneity and temporal trends to better predict traffic accidents [136]. Studies in this category usually make use of many pieces of information that may not be available in real-time applications.

Accident Risk Prediction. This category of work is very much similar to the previous one, unless prediction here is defined as a binary classification task which better fits real-time applications [119, 120]. Using data for a single segment of I-64 in Virginia (US), Lin et al. [82] leveraged a decision tree model to separate pre-crash records from normal ones, using information such as weather, visibility, traffic volume, speed, and occupancy information.

However, their limited size of data might weaken their solution or findings. In another study, Chen et al. [92] used human mobility data in terms of 1.6 million GPS records and a set of 300,000 accident records in Tokyo (Japan) to predict the possibility of accident occurrence on grid cells of size $500m \times 500m$ in an hourly basis. They leveraged a stack denoising autoencoder model to extract latent features from human mobility, and then used a logistic regression model to predict accidents. Finally, Yuan et al. [120] used a heterogeneous set of urban data such as road characteristics (AADT, speed limit, etc.), radar-based rainfall data, temperature data, and demographic data to predict the probability of accident for each road-segment in the state of Iowa. They leveraged eigen-analysis to capture and represent spatial heterogeneity. Their analyses and results suggest the importance of time, human factors, weather data, and road-network characteristics for this task.

Our proposal belongs to the last category as we seek to perform accident risk prediction. Further, our solution is more suitable for real-time applications as we provide a prediction for a much shorter time interval (i.e., 15 minutes) in comparison to the literature. Besides, our usage of real-time traffic events and points-of-interest, to the best of our knowledge, is not discussed before. Lastly, the type of input data that we use for prediction is rather easy-to-collect and available to the public, in contrast to those work that used extensive datasets for modeling and prediction.

5.4 Accident Dataset

This section describes the process of constructing a country-wide traffic accident dataset, which we named *US-Accidents*. An overview of this process is shown in Figure 5.1. US-Accident contains 2.25 million cases of traffic accidents that took place within the United States from February 2016 to March 2019. The following sub-sections provide a detailed description of each step of the data preparation process. The dataset is publicly available at `https://smoosavi.org/datasets/us_accidents`.

5.4.1 Traffic Data Collection

Realtime Traffic Data Collection

We collected streaming traffic data using two real-time data providers, namely "MapQuest Traffic" [151] and "Microsoft Bing Map Traffic" [138], whose APIs broadcast traffic events

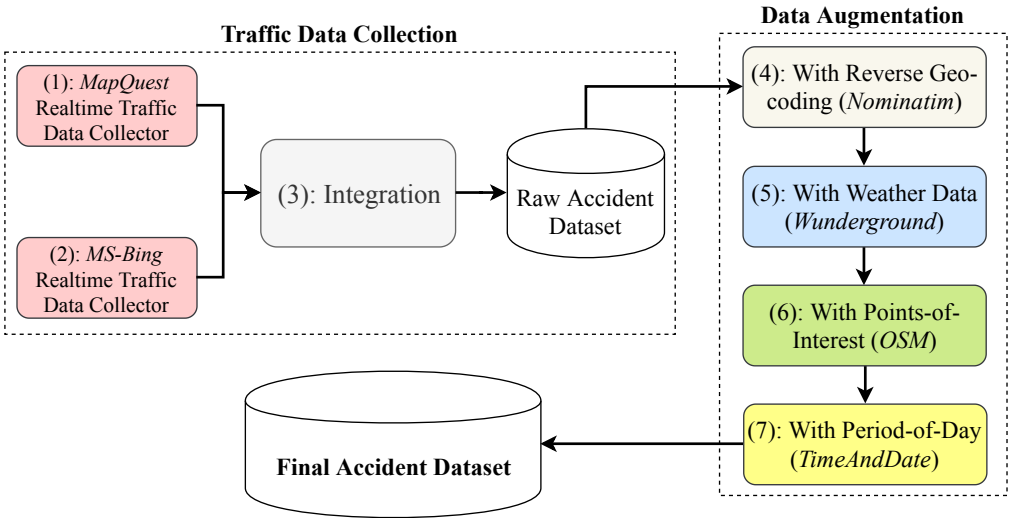


Figure 5.1: Process of Creating Traffic Accident Dataset

(accident, congestion, etc.) captured by a variety of entities - the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. We pulled data every 90 seconds from 6 am to 11 pm, and every 150 seconds from 11 pm to 6 am. In total, we collected 2.27 million cases of traffic accidents between February 2016 and March 2019; 1.73 million cases were pulled from MapQuest, and 0.54 million cases from Bing.

Integration

The integration of the data consisted of removing cases duplicated across the two sources and building a unified dataset. We considered two events as duplicates if their Haversine distance and their recorded times of occurrence were both below a heuristic threshold (set empirically at 250 meters and 10 minutes, respectively). We believe these settings to be conservative, but we settled on them in order to ensure a very low possibility of duplicates. Using these settings, we found about $24,600$ duplicated accident records, or about 1% of all data. The final dataset after removing the duplicated cases comprised 2.25 million accidents.

5.4.2 Data Augmentation

Augmenting with Reverse Geo-Coding

Raw traffic accident records contained only GPS data. We employed the *Nominatim* tool [145] to perform reverse geocoding to translate GPS coordinates to addresses, each consisting of a *street number*, *street name*, *relative side (left/right)*, *city*, *county*, *state*, *country*, and *zip-code*. This process is same as *point-wise map-matching*.

Augmenting with Weather Data

Weather information provides an important context for traffic accidents. Thus, we employed *Weather Underground* API [152] to obtain weather information for each accident. Raw weather data was collected from 1,977 weather stations located in airports all around the United States. The raw data comes in the form of observation records, where each record consists of several attributes such as *temperature*, *humidity*, *wind-speed*, *pressure*, *precipitation* (in millimeters), and *condition*[23]. For each weather station, we collected several data records per day, each of which was reported upon any significant change in any of the measured weather attributes.

Each traffic event e was augmented with weather data as follows. First the closest weather station s was identified. Then, of the weather observation records which were reported from s, we looked for the weather observation record w whose reported time was closest to the start time of e, and augmented it with weather data. In our integrated accident dataset, the average difference in report time for an accident record and its paired weather observation record was about 15 minutes.

Augmenting with Points-Of-Interest

Points-of-interest (POI) are locations annotated on a map as *amenities*, *traffic signals*, *crossings*, etc. These annotations are associated with *nodes* on a road-network. A node can be associated with a variety of POI types, however, in this work, we only use 13 types as described in Table 5.2. We obtained these annotations from Open Street Map (OSM) [147] for the United States, using its most recently released dataset (extracted in April 2019). The applicable POI annotations for a traffic accident a are those which are located

[23]Possible values are *clear*, *snow*, *rain*, *fog*, *hail*, and *thunderstorm*.

within a distance threshold τ from a. We determine this threshold by evaluating different values to find the value that is best able to associate a POI with an accident. Essentially, the objective is to find the best distance for which a POI annotation can be identified as *relevant* to an accident record. Therefore, we need a mechanism to measure the relevancy. To begin with, we note that the natural language descriptions of traffic accidents follow a set of regular expression patterns, and that a few of these patterns may be used to identify and use as an annotation for the location type (e.g., intersection or junction) of the accident.

Regular Expression Patterns. Given the description of traffic accidents, we were able to identify 27 regular expression patterns; 16 of them were extracted based on MapQuest data, and 11 from Bing data. Among the MapQuest patterns, the following expression corresponds to *junctions* (see Table 5.2): "... **on** ... **at exit** ...", and the following pattern mostly[24] determines an *intersection*: "... **on** ... **at** ...". An intersection is associated with *crossing*, *stop*, or *traffic signal* (see Table 5.2). Among Bing regular expression patterns, two of them identify junctions: "**at** ... **exit** ..." and "**ramp to** ...". Table 5.3 shows several examples of accidents, where the regular expression pattern (in bold face) identifies the correct POI type[25].

Table 5.3: Examples of traffic accidents with their *annotation type* assigned using their natural language description by regular expression patterns.

Source	Description	Type
MapQuest	Serious accident **on** 4th Ave **at** McCullaugh Rd.	Intersection
MapQuest	Accident **on** NE-370 Gruenther Rd **at** 216th St.	Intersection
MapQuest	Accident **on** I-80 **at Exit** 4A Treasure Is.	Junction
MapQuest	Accident **on** I-87 I-287 Southbound **at Exit** 9 I-287.	Junction
Bing	**At** Porter Ave/**Exit** 9 - Accident. Left lane blocked.	Junction
Bing	**At** IL-43/Harlem Ave/**Exit** 21B - Accident.	Junction
Bing	**Ramp to** I-15/Ontario Fwy/Cherry Ave - Accident.	Junction
Bing	**Ramp to** Q St - Accident. Right lane blocked.	Junction

[24]Using 200 randomly sampled accidents cases which were manually checked on a map, about 78% of matches using this pattern were actually occurred on intersections.

[25]These cases were manually checked on a map to ensure the correctness of the annotation.

Algorithm 7: Find Annotation Correlation

Input: A dataset of traffic accidents $\mathcal{A}$, a database of points-of-interest $\mathcal{P}$, and a distance threshold τ

1 Extract and create a set of regular expression patterns RE to identify a specific POI ν

2 **Create set** S_1: for each traffic accident $a \in \mathcal{A}$, we add it to S_1 if its natural language description $a.desc$ can be matched with at least one regular expression in set RE

3 **Create set** S_2: for each traffic accident $a \in \mathcal{A}$, we add it to S_2 if there is at least one POI $p \in \mathbf{P}$ of type ν, where *haversine_distance* $(a, p) \leq \tau$

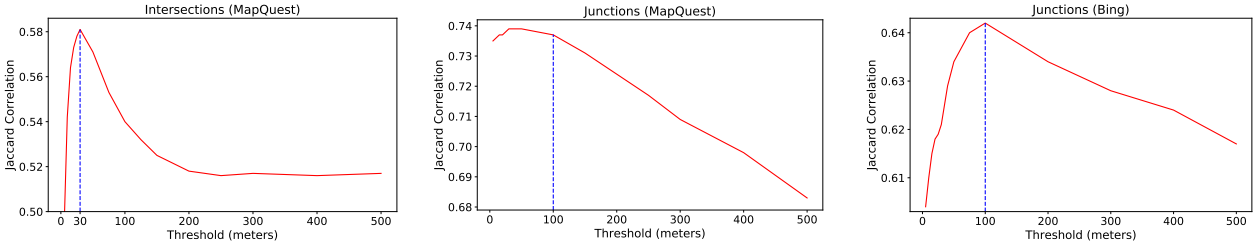
Output: *Jaccard* (S_1, S_2)

The fundamental idea is to find a threshold value that maximizes the correlation between annotations from POI and annotations derived using regular expression patterns. Thus, for a set of accident records, we annotate their location based on both methods, regular expression patterns as well as OSM-based POI annotations (using a specific distance threshold). Then, we measure the correlation between the annotations derived from these methods to find which threshold value provides the highest correlation (i.e., the best choice). Note that we employ the regular expression patterns as *pseudo* ground truth labels, to evaluate OSM-based POI annotations using different threshold values. We propose Algorithm 7 to find the best distance threshold. We use a sample of $100,000$ accidents as set $\mathcal{A}$ (the input). For step 1, we consider either "intersection" or "junction", and use the set of relevant regular expressions (see Table 5.3) in terms of RE. Next we create set S_1 by annotating each traffic accident $a \in \mathcal{A}$ using the regular expression patterns in RE (step 2). Then we annotate each traffic accident $a \in \mathcal{A}$ based on points-of-interests in $\mathcal{P}$, using the distance threshold τ to create S_2 (step 3). Finally, we calculate the Jaccard similarity score using Equation 5.1 (the output):

$$Jaccard\ (S_1, S_2) = \frac{\left| S_1 \cap S_2 \right|}{\left| S_1 \cup S_2 \right|} \tag{5.1}$$

We examined the following candidate set to find the optimal threshold value (all values in meters): $\{5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 125,\ 150, 200, 250, 300, 400, 500\}$. We separately studied samples from Bing and MapQuest, and employed corresponding regular expression

patterns for "intersection" and "jucntion". Figure 5.2 shows the results for each data source and each annotation type. From Figure 5.2a, we see that the maximum correlation for intersections is obtained for a threshold value of 30 meters. Figures 5.2b and 5.2c show that 100 meters is an appropriate distance threshold for annotating a junction.



(a) Using MapQuest for Intersection (b) Using MapQuest for Junction (c) Using Bing for Junction

Figure 5.2: Correlation study between regular-expression and OSM-based extracted annotations to find the best distance threshold values.

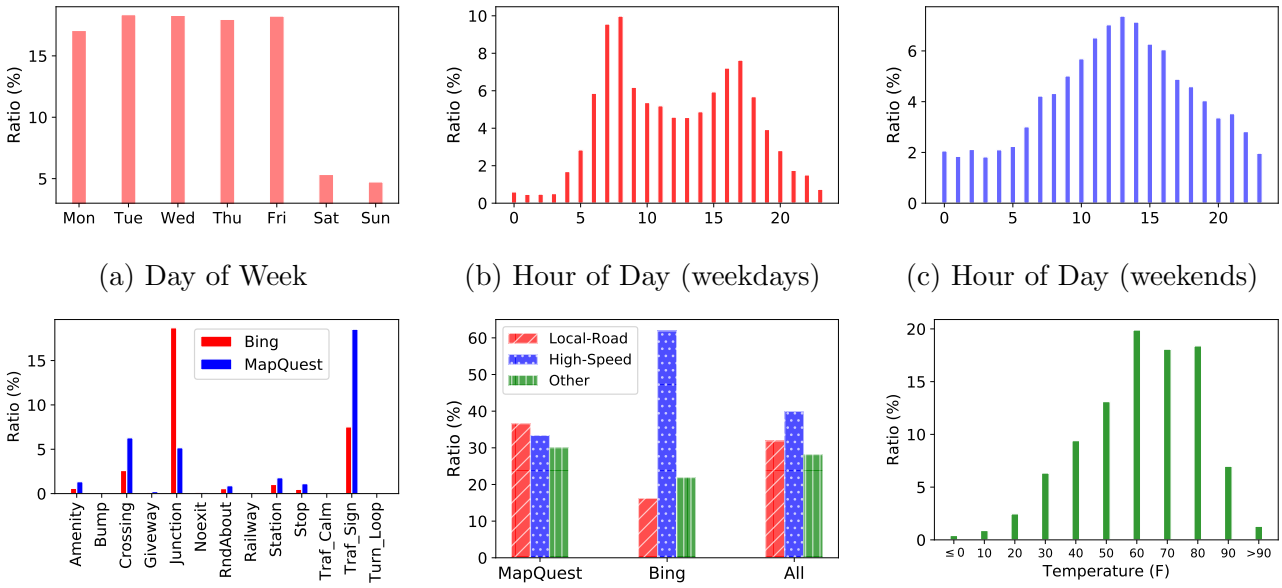
Thresholds for the other available annotations in Table 5.2 are derived from the thresholds for junction and intersection as described below:

- **Junction-based threshold.** Given the definition of a junction (i.e., a highway ramp, exit, or entrance), we used the same threshold (100 meters) for the following types: amenity and no-exit.
- **Intersection-based threshold.** Given the definition of an intersection, we used the same threshold (30 meters) for the following annotation types: bump, crossing, give-way, railway, roundabout, station, stop, traffic calming, traffic signal, and turning loop.

Using these thresholds, we augmented each accident record with points-of-interest. In summary, 27.5% of accident records were augmented with at least one of the available POI types in Table 5.2. Further discussion on annotation results are presented in Section 5.4.3.

Augmenting with Period-of-Day

Given the start time of an accident record, we used "TimeAndDate" API [149] to label it as *day* or *night*. We assign this label based on four different daylight systems, namely



(a) Day of Week (b) Hour of Day (weekdays) (c) Hour of Day (weekends)

(d) Points-of-interest Distribution (e) Road-type Distribution (f) Temperature Distribution

Figure 5.3: Characteristics of US-Accidents dataset, in terms of time analysis (a)–(c), points-of-interest-based augmentation distribution analysis (d), map-matching-based road type coverage analysis (e), and temperature analysis (f).

Sunrise/Sunset, *Civil Twilight*, *Nautical Twilight*, and *Astronomical Twilight*. Note that these systems are defined based on the position of the sun with respect to the horizon, and each provide a different definition for period-of-day[26].

5.4.3 US-Accidents Dataset

Using the process described above, we created a countrywide dataset of traffic accidents, which we name *US-Accidents*. US-Accident contains about 2.25 million cases of traffic accidents that took place within the contiguous United States from February 2016 to March 2019. Table 5.4 shows the important details of US-Accidents. Also, Figure 5.3 provides more details on the characteristics of the dataset. Figure 5.3-(a) shows that significantly more accidents were observed during the weekdays than weekends. Based on parts (b) and (c) of Figure 5.3, it can be observed that the hourly distribution during weekdays has two peaks

[26]See `https://en.wikipedia.org/wiki/Twilight` for more details.

(8 am and 5 pm), while the weekend distribution shows a single peak (1 pm). Figure 5.3-(d) demonstrates that most of the accidents took place near junctions or intersections (crossing, traffic signal, and stop). MapQuest tends to report more accidents near intersections, while Bing reported more cases near junctions. This shows the complementary behavior of these APIs, and hence the comprehensiveness of our dataset. Figure 5.3-(e) describes distribution of road types, extracted from the map-matching results (i.e., street names). We used street names to identify the type of the road. Here we note that about 32% of accidents happened on or near local roads (e.g., streets, avenues, and boulevards), and about 40% took place on or near high-speed roads (e.g., highways, interstates, and state roads). We also note that Bing reported more cases on high-speed roads. Finally, the period-of-day data shows that about 73% of accidents happened after sunrise (or during the day).

Table 5.4: US-Accidents: details as of March 2019.

Total Attributes	45
Traffic Attributes (10)	id, source, TMC [150], severity, start_time, end_time, start_point, end_point, distance, and description
Address Attributes (8)	number, street, side (left/right), city, county, state, zip-code, country
Weather Attributes (10)	time, temperature, wind_chill, humidity, pressure, visibility, wind_direction, wind_speed, precipitation, and condition (e.g., rain, snow, etc.)
POI Attributes (13)	All cases in Table 5.2
Period-of-Day (4)	Sunrise/Sunset, Civil Twilight, Nautical Twilight, and Astronomical Twilight
Total Accidents	2,243,939
Total MapQuest Accidents	1,702,565 (75.9%)
Total Bing Accidents	516,762 (23%)
Total Reported by Both	24,612 (1.1%)
Top States	California (485K), Texas (238K), Florida (177K), North Carolina (109K), New York (106K)

5.5 Accident Prediction Model

In this section, we describe our traffic accident prediction framework. We start with a description of feature vector representation, and then present our proposal for real-time traffic accident prediction.

5.5.1 Feature Vector Representation

Regarding the problem description in Section 5.2, we create a feature vector representation for each geographical region r of size $5km \times 5km$ during a time interval $t = 15\,minutes$. Such representation includes the following feature categories:

- **Traffic**: a vector of size 7 representing the frequency of available traffic events (i.e., accident, broken-vehicle, congestion, construction, event, lane-blocked, and flow-incident) during the current 15 minutes interval. We obtain traffic events from [143].

- **Time**: includes *weekday* (a binary value to show weekday or weekend), *hour-of-day* (a one-hot vector of size 5 to show belonging to a specific time interval as defined in [112])[27], and *daylight* (an attribute to show period-of-day: day or night). We obtain daylight data from [149].

- **Weather**: a vector representing 10 weather attributes including temperature, pressure, humidity, visibility, wind speed, precipitation amount; and four indicator flags for special events rain, snow, fog, and hail. We obtain weather data from [152].

- **POI**: a vector of size 13 to represent the frequency of POIs within r, for amenity, speed bump, crossing, give-way sign, junction, no-exit sign, railway, roundabout, station, stop sign, traffic calming, traffic signal, and turning loop. We obtain POI data from [147].

- **Desc2Vec**: given a historical set of traffic events in the region r, we use their natural language description, and by employing the GloVe pre-trained distributed word vectors [70], we create a description to vector (Desc2Vec) representation for r. Such representation is the average representation of words in the description of all events which took place within r during a particular time period. The size of this vector is 100. The choice of

[27]These time intervals are [6am – 10am], [10am – 3pm], [3pm – 7pm], [7pm – 10pm], and [10pm – 6am].

GloVe among the existing models is because of its well-known applicability for generic applications and also reasonable dictionary size (i.e., 400K terms). We obtain traffic events from [143].

In this way, we represent r during time interval t by 24 time-variant (i.e., traffic, time, and weather) and 113 time-invariant (i.e., POI and Desc2Vec) attributes. In order to predict the label of r during t, we use a vector representing the last 8 time intervals (last two hours), including one instance of time-invariant attributes (113 features) and 8 instances of time-variant attributes (8×24 features)[28].

5.5.2 Deep Accident Prediction (DAP) Model

To better use heterogeneous sources of data and perform real-time traffic accident prediction, we propose a deep neural network model, named the Deep Accident Prediction (DAP). This model is shown in Figure 5.4, and we describe its components as follows.

- **Recurrent Component**: Regarding the definition of our prediction framework, we use a set of 8 vectors, each of size 24 (i.e., time-variant attributes), which can be treated as a sequence of such vectors (given their temporal order); therefore, we may benefit from the recurrent neural network models. Specifically, we use a long-short-term-memory (LSTM) model [9] as represented in Figure 5.4, which includes two recurrent layers, each with 128 LSTM cells. Thus, the output is a vector of size 128.

- **Embedding Component**: Given the index of a grid-cell, this component provides a distributed representation of that cell that encodes essential information in terms of spatial heterogeneity, traffic characteristics, and impact of other environmental stimuli on accident occurrence. This distributed representation will be derived as we train the entire pipeline. We feed this representation to a feed-forward layer of size 128 that uses the *sigmoid* activation function. Note that the embedding matrix is of size $|R| \times 128$, where R is the set of all grid-cell regions in the input dataset.

- **Description-to-Vector Component**: This component uses the natural language description of historical traffic events in a grid-cell, that is, Desc2Vec data. We feed Desc2Vec of a grid-cell to a feed-forward layer of size 128 using the *sigmoid* activation function.

[28]See Section 5.2 for formulation of prediction task.

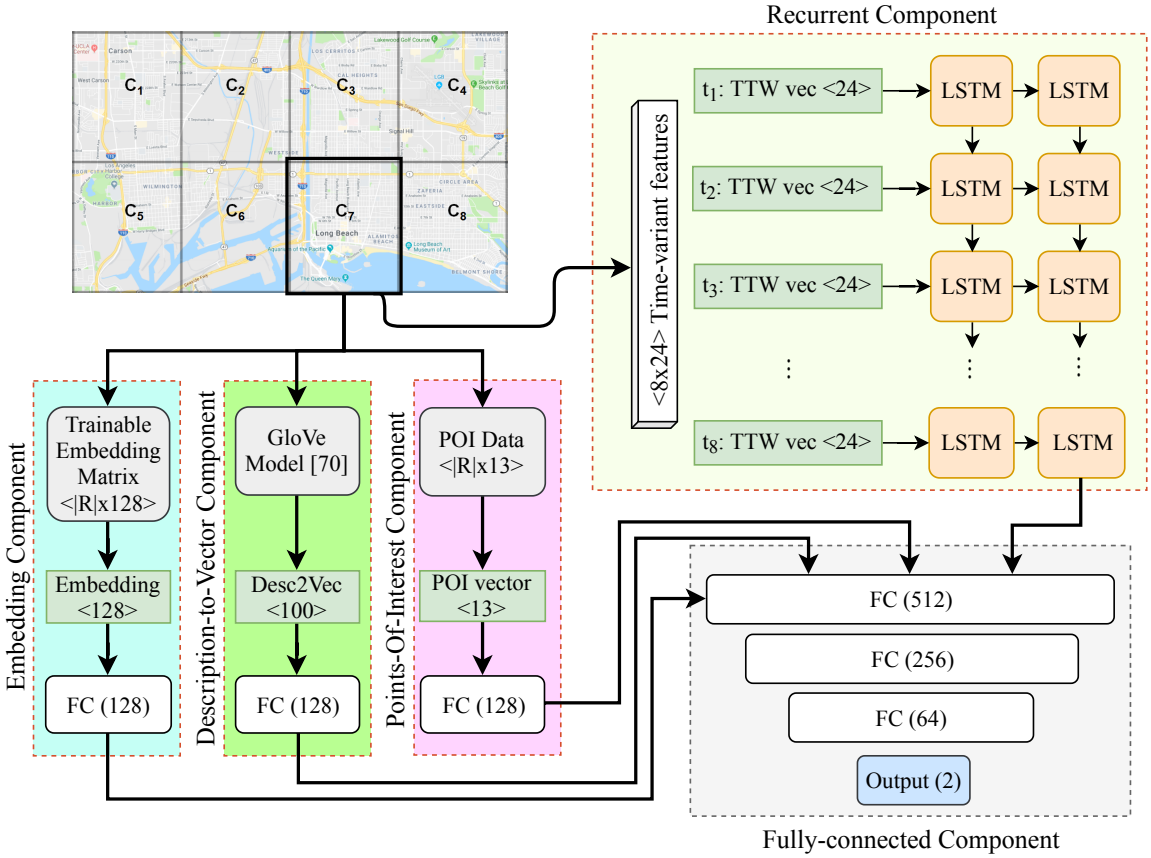


Figure 5.4: DAP: A Deep neural-network-based Accident Prediction model. Here, R is the set of all regions; each C_i is a grid-cell (or region); and FC, TTW, and POI stand for fully connected, time-traffic-weather, and point-of-interest, respectively.

- **Points-of-Interest Component**: This component uses points-of-interest data (a vector of size 13), which is a representation of spatial characteristics. We feed a POI vector to a feed-forward layer of size 128 which also uses the *sigmoid* activation function.

- **Fully-connected Component**: This component uses the output of the above components to make the final prediction. Here we have four dense layers of size 512, 256, 64, and 2, respectively. Additionally, to speed-up the training process, we use batch normalization [78] after the second and the third layer. We use ReLU as the activation function of the first three layers, and apply *softmax* on the output of the last layer.

The DAP model uses inputs of various types to better capture temporal and spatial heterogeneity. Using DAP we can extract latent spatio-temporal features in terms of embedding representations, whose impact we show through our real-world experiments. We employed grid-search to perform hyper-parameter tuning to find the optimal number of recurrent layers (choices of $\{1, 2, 3\}$); the best type of recurrent cells (choices of $\{Vanilla\text{-}RNN, GRU, LSTM\}$); size of the embedding vector for grid-cells (choices of $\{50, 100, 150\}$); sizes of the different fully connected layers (choices of $\{64, 128, 256, 512\}$); and activation function for each fully connected layer (choices of $\{sigmoid, ReLU, tanh\}$). We employed the Adam optimizer [68] with an initial learning rate of 0.01 to train the model.

5.6 Experiments and Results

In this section, we first describe the data which is used for prediction and analysis. Then, we describe baseline models. Next, we compare different models using a variety of metrics, followed by analyses of data attributes. All implementations are in Python using Tensorflow [77], Keras [76], and scikit-learn [45] libraries; and experiments were run on nodes at the Ohio Supercomputer Center [4] [29].

5.6.1 Data Description

To evaluate our accident prediction framework, we chose six cities: *Atlanta*, *Austin*, *Charlotte*, *Dallas*, *Houston*, and *Los Angeles*; primarily so as to achieve diversity in traffic and weather conditions, population, population density, and urban characteristics (road-network, prevalence of urban versus highway roads, etc.). We sampled a subset of data (traffic, weather, etc.) collected from June 2018 to August 2018 (i.e., 12 weeks) for each city. We chose this time period to prevent any noises as result of seasonality in weather and traffic patterns. To create Desc2Vec for each grid cell region, we used traffic events that took place within that region from June 2017 to May 2018 (i.e., a one-year time frame), where data obtained from the *Large-Scale Traffic and Weather Events dataset* [143]. From the traffic, time, weather, POI, and Desc2Vec data for each grid cell, and by scanning through the data with a window of size 2 hours and 15 minutes and a shift of 15 minutes

[29] Code and sample data is available at `https://github.com/mhsamavatian/DAP`.

(see Figure 5.5), we built a *sample entry* using data of the first two hours (see Section 5.5.1). Each entry is represented by 113 time-invariant and 8×24 time-variant features. The last 15 minutes interval is used to label the sample entry as an accident or N-Accident case.

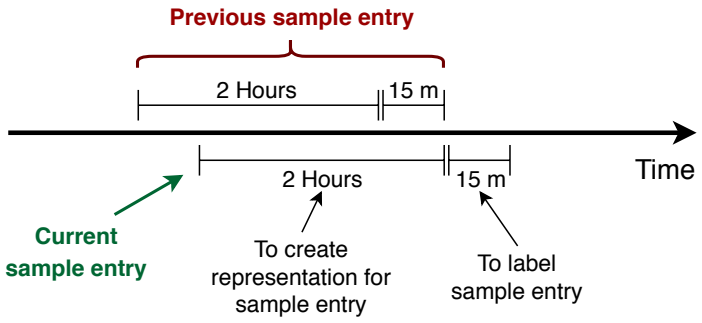


Figure 5.5: Creating a Sample Entry (see Section 5.6.1).

Since accidents are rare and because our dataset is sparse [30], we performed *negative sampling* to balance the frequency of samples between accident and N-Accident classes. Specifically, we uniformly sampled from the N-Accident class with a probability of 2.0%. Table 5.5 summarizes the number of samples for each class (Acc versus N-Acc), for each city, after negative sampling. As can be seen, the maximum ratio of accident to N-Accident is about 27% for *Los Angeles* (which is still lower than the ratio which is employed by previous studies; e.g., Yuan et al. [120] employed 33%). Table 5.5 also shows the number of all other traffic events (except accidents) which took place during the selected 12 weeks time frame. We use data from the first 10 weeks to train and data from the last two weeks as the test set, for each city.

5.6.2 Baseline Models

We chose logistic regression (LR), gradient boosting classifier (GBC), and a Deep Neural Network (DNN) model as baselines.

[30]Our data is a result of streaming data with the possibility of missing records.

Table 5.5: Distribution of accident (Acc) and N-Accident (N-Acc) classes, and traffic events (except accidents).

City	#Acc	#N-Acc	Acc/N-Acc	#Traffic Events
Atlanta (GA)	2,630	11,970	22%	24,396
Austin (TX)	4,274	23,280	18%	16,313
Charlotte (NC)	5,295	20,192	26%	14,030
Dallas (TX)	3,363	28,537	12%	28,098
Houston (TX)	5,859	43,762	13%	40,735
Los Angeles (CA)	7,974	29,020	27%	97,090

- **Logistic Regression (LR)**: A significant number of previous studies leveraged regression-based models to perform accident prediction [20, 92, 122]. Therefore, we employ logistic regression as a reasonable baseline to perform our binary classification task.

- **Gradient Boosting Classifier (GBC)**: GBC is a popular general-purpose classification model, with useful boosting characteristics and a suitable learning process. In practice, GBC usually provides superior results for binary or multi-class classification tasks, when compared to the other models such as Random Forest or Support Vector Machine; our preliminary experiments also confirmed this.

- **Deep Neural Network (DNN)**: This is a four-layer feed-forward neural network, with three hidden layers of size 512, 256, and 64, respectively. *ReLU* was used as the activation function of the hidden layers, and *softmax* was applied on the output of the last layer. To speed-up the training process, we used batch normalization [78] after the second and third hidden layers. We employed the Adam optimizer [68] with an initial learning rate of 0.01 to train this model.

As input, the baseline models use vectors of size 305, that includes 113 time-invariant and 192 time-variant attributes (see Section 5.5.1). The output is the prediction probability for "accident" and "non-accident" classes. Using grid-search over heuristic choices of parameters, we found the best parameter setting for each model. For LR, we performed the grid search over choices of regularizations: $\{L1, L2\}$, maximum iterations: $\{100, 100, 10000, 100000\}$, and solvers: $\{newton\text{-}cg, lbfgs, sag, liblinear\}$. For GBC, the grid search was performed over choices of learning rates: $\{0.01, 0.05, 0.1, 0.15\}$, number of estimators: $\{100, 200, 300, 400\}$,

Table 5.6: Accident prediction results based on F1-score for class of accidents (Acc), non-accidents (N-Acc), and weighted average (W-avg).

Model / City	LR			GBC			DNN			DAP-NoEmbed			DAP		
	Acc	N-Acc	W-Avg	Acc	N-Acc	W-Avg	Acc	N-Acc	W-Avg	Acc	N-Acc	W-Avg	Acc	N-Acc	W-Avg
Atlanta	0.54	0.91	0.83	0.57	0.91	0.84	0.62	0.89	0.83	0.62	0.91	0.84	**0.65**	0.89	0.84
Austin	0.58	0.93	0.87	0.61	0.93	0.87	0.62	0.92	0.87	0.62	0.93	0.87	**0.64**	0.91	0.87
Charlotte	0.56	0.91	0.83	0.60	0.91	0.84	0.61	0.87	0.82	0.61	0.87	0.81	**0.63**	0.87	0.82
Dallas	0.30	0.94	0.87	0.32	0.94	0.87	0.36	0.94	0.87	0.43	0.88	0.83	**0.50**	0.93	0.88
Houston	0.49	0.94	0.88	0.51	0.94	0.88	**0.59**	0.93	0.88	0.58	0.92	0.88	0.58	0.93	0.88
Los Angeles	0.41	0.88	0.78	0.45	0.88	0.79	0.53	0.81	0.75	0.53	0.77	0.72	**0.56**	0.84	0.78

and maximum depth: $\{3, 4, 5, 6\}$. For DNN, the grid search was performed over choices of initial learning rates: $\{0.001, 0.01, 0.05, 0.1\}$, activation functions: $\{sigmoid, ReLU\}$, number of hidden layers: $\{2, 3, 4\}$, and size of hidden layers: $\{128, 256, 512\}$.

5.6.3 Exploring Models

In this section, we evaluate different models based on their ability to predict traffic accidents. That is, we compare different models based on $F1$-*score* (defined by Equation 5.2), reported for each class separately, as well as the *weighted average* $F1$-*score* (the relative frequency of each class is used as its weight).

$$
\begin{aligned}
Precision &= \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \\
Recall &= \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \\
F1\text{-}Score &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\end{aligned}
\tag{5.2}
$$

We used logistic regression (LR), gradient boosting classifier (GBC), and a deep neural network (DNN) model as baselines. We report the result of our DAP model, as well as a variation of DAP without the embedding component (DAP-NoEmbed). We ran each model three times and reported the average results. As mentioned before, we used the grid search to find the optimal parameters. For LR and GBC, we performed this for each city, but for the neural-network-based models, we employed grid search for one city and used the best architecture setting for the other cities. DNN, DAP, and DAP-NoEmbed were trained for

60 epochs, and using *early stopping* based on the validation set (i.e., 10% of the training set), we used the best model for prediction on the test set. It is worth noting that each model is separately trained and tested for each city and we do not train a single model for all cities. Table 5.6 presents the results of this experiment. In this table, we report $F1$-*score* for the class of accident (Acc), non-accident (N-Acc), and the weighted average (W-Avg). We note that the class of accidents is usually more important, while we seek to provide reasonable results for the other class (non-accident) as well. LR and GBC usually provide better results for non-accident class, and given the frequency of this class, their weighted average score is also reasonably high. However, when considering the accident class, we note that neural-network-based models provide more satisfactory results, where our proposed DAP model provides superior results for 5 of the 6 cities (DNN provided the best result for Houston). Considering the weighted average on $F1$-*score*, we note that DAP provides better results when compared to the other neural-network-based models.

To better compare different models, Figure 5.6 shows the average results of different models across all six cities, by separately reporting $F1$-*score* for class of accident and non-accident, and the weighted average $F1$-*score*. As one can see, our proposed model provides a significant improvement for the class of accidents, while LR and GBC provide slightly better results for the non-accident class. When considering the weighted average, we observe LR, DAP and GBC slightly outperform the other models. Once again note that the "accident class" is the one of most importance, given that accidents are rare events. Hence we should pay more attention to false negatives (i.e., predicting an accident as a non-accident) rather than false positives (i.e., predicting a non-accident as an accident).

While we cannot directly compare our proposal with the state-of-the-art models such as [120, 92, 82] (due to inconsistency between input types, unavailability of input data used by those models, inconsistency between reported metrics, etc.), we note that their reported results based on $F1$-*score* show similar trend and values (see [120] for example). Further, we believe that separately reporting prediction results for different classes (i.e., accident versus non-accident) provides a better context to compare different solutions.

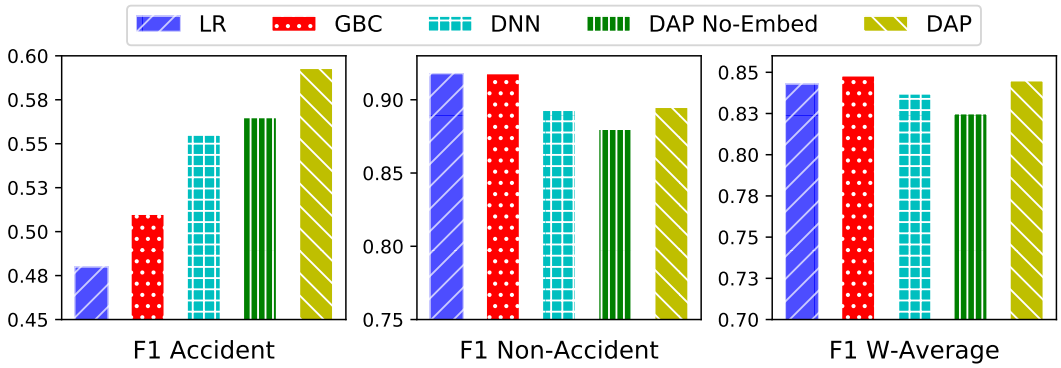


Figure 5.6: Comparing different models based on average $F1$-*score* (across all six cities) for class of accident, non-accident, and weighted average.

5.6.4 Exploring Features

Our next experiment was to examine the importance of different feature categories for the task of accident prediction. For this exploration, we designed two testing scenarios as follows:

- **Only One**: This scenario means we only use one category of features (traffic, POI, time, etc.) to perform accident prediction.
- **All But One**: This scenario means to remove only one category of features and perform the prediction task.

For this experiment, we only report the result of GBC and DAP-NoEmbed, and omit the results of other models for the interest of space. Also, because of having the trainable embedding component, we choose DAP-NoEmbed over DAP to exclude the effect of the embedding component when studying the impact of other features[31]. Figure 5.7 demonstrates the results, where we report weighted average $F1$-*score*, and $F1$-*score* on accident class. Based on parts (a), (b), (e), and (f), we generally observe weather (WE) and time (TM) are the least important categories of attributes to be used alone[32]. However, parts (c), (d),

[31]Since DAP uses an embedding component, we cannot fairly study the impact of several categories of features (in isolation), such as traffic, weather, and points-of-interest; given the correlation between these categories and the latent representation which will be derived for each region.

[32]Note that we could not use categories D2V and POI for DAP-NoEmbed, regarding the architecture of this model.

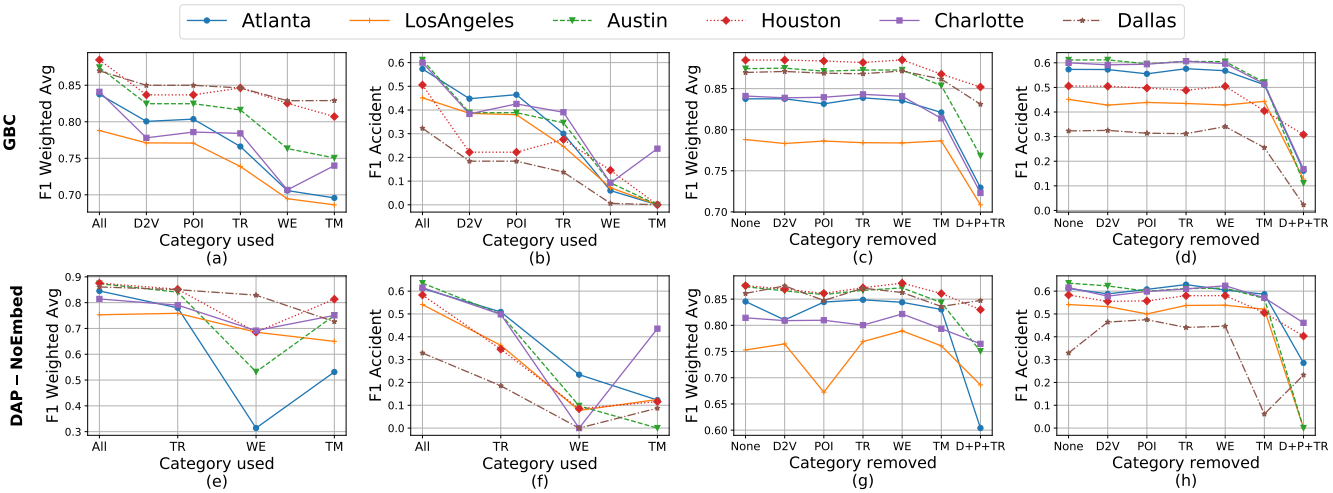


Figure 5.7: Prediction results using only one category (a, b, e, and f), and all but one category (c, d, g, and h). Here D2V, TR, WE, and TM stand for Desc2Vec, traffic, weather, and time, respectively. Also, "D+P+TR" means removing Desc2Vec, POI, and traffic from input features.

(g), and (h) reveal that removing time attributes would significantly hurt the prediction performance. Based on these figures, when we remove Desc2Vec, POI, and Traffic attributes (i.e., D+P+TR), the prediction performance drops significantly, which shows the importance of these categories. We may also note that these categories might be correlated, such that removing one of them does not significantly change the prediction results (see (c) and (d)). Therefore, when we remove all three, then we observe a significant drop.

It is worth noting that among the POI types, we found "crossing", "junction", "stop", and "traffic signal" to be more effective than the others for the task of accident prediction.

5.7 Summary and Conclusion

Traffic accidents are a major public safety issue, with much research devoted to the analysis and prediction of these events. However, most studies suffer from using small-scale datasets, relying on extensive data that is not easily accessible to other researchers, and lacking applicability for real-time purposes. To address these challenges, we introduce a new framework for real-time traffic accident prediction based on easy-to-obtain but sparse contextual data. Our prediction model incorporates several neural-network-based

components that use data attributes such as traffic events, weather data, points-of-interest, and time information. We also create a publicly available countrywide traffic accident dataset, named US-Accidents, through a comprehensive process of data collection, cleansing, and augmentation. Using the data from US-Accidents, we compare our work against several neural-network-based and traditional machine learning models, and show its superiority through extensive experiments. Further, we study the impact of different categories of attributes for traffic accident prediction, and find time, traffic events, and points-of-interest as having significant value.

Chapter 6: Context-aware Driving Risk Prediction

This chapter describes a micro-level, context-aware driving risk prediction framework based on telematics data. Most existing telematics-based driving risk prediction studies leverage demographic data in addition to telematics data, which is not the case in this study. Instead, our objective is to *maximize* the use of telematics as well as contextual data to provide a risk prediction framework with applicability for real-world applications. The first step is to *contextualize the telematics data*. We achieve this by representing raw trajectory data in different contexts, and by generating comprehensive views of the telematics data using various attributes (e.g., speed, acceleration, and turn speed). The second step is to identify the major risk cohorts in the data and build a *risk cohort prediction classifier*. This step used the contextualized telematics data and weak risk labels, and employs a data-driven process to refine the weak risk labels prior to building the classifier. The last step is to build a *driving risk prediction process* to be used for real-time purposes, which can be utilized for applications such as usage-based insurance and driver coaching.

6.1 Motivation

A recent trend in the auto insurance industry is to use *telematics data* (i.e., data collected by OBD-II devices or smartphone applications) together with *demographic information* (e.g., age, gender, and marital status) in the underwriting process[33] to determine driving risk and insurance policy rates. The important benefit of using telematics data is to better demonstrate *naturalistic driving behavior*, which cannot be achieved if only demographic information is considered. However, understanding how to utilize telematics data to best effect remains a challenging problem. While one group of existing studies has employed telematics data by summarizing them as *coarse-grained* representations [133, 129, 137, 141],

[33]See `https://en.wikipedia.org/wiki/Underwriting` for more details about this process.

another group of studies sought to use such information in a more *fine-grained* form [57, 69, 74, 99, 121, 125, 128, 135, 139], with neither set of studies having made the case for which way leads to a best practice.

In addition to telematics data, we also need *contextual information* to better evaluate someone's behavior in order to rate their auto insurance policy. Examples of contextual information are location data, time information, traffic conditions, and weather conditions. Driving behavior will be affected by these contextual properties. As an example, a safe driver on dry days may be seen as a risky driver on rainy days. Nevertheless, several existing studies have failed to incorporate proper contextual information when predicting future risk [57, 69, 74, 121, 125, 134, 135, 139].

Another significant challenge is to *define* and *measure* driving risk. Many previous studies defined driving risk as the likelihood of involvement in an accident, a serious traffic offense (e.g., driving under the influence or DUI), or, in general, filing an *at-fault* insurance claim [3, 6, 8, 15, 57, 69, 74, 99, 121, 128, 125, 135, 139]. Other recent studies defined the risk based on the driver taking dangerous actions (with the actions being specified by human experts) [26, 57, 69, 74]. Still other recent studies have employed human experts to label drivers with appropriate risk scores given their history of driving as derived from telematics data [135]. We define risk as a factor of an individual's previous driving record, including accidents and traffic violations. This information is readily available for a large group of drivers, it can be used to assess driving behavior, and it can be used as *partial ground-truth labels*[34].

In this dissertation, we propose a new driving risk prediction framework that utilizes telematics data together with contextual information. This framework uses a finer-grained way of representing telematics data in comparison to previous studies [99, 121, 125, 126, 128, 129, 134, 135]. Our risk prediction framework consists of three major parts described as follows. The first part derives fine-grained representations of trajectories within context. We directly encode contextual information while creating these representations. Here we employ road type (e.g., residential, urban, and highway area) and road shape (e.g., turn segments) as contextual information. Our representation comprises trajectory information, that is, attributes such as speed, acceleration, angular-speed, and change-of-angle.

[34]Having traffic conviction records is not a sufficient indicator of being a risky driver; not having a record does not endorse safeness; therefore, we refer to such information as partial ground-truth labels.

The second part of the framework employs these contextualized telematics representations and builds a classifier to predict *risk cohorts* (e.g., low-risk, high-risk, etc.). This step includes refining the risk labels by a data-driven process that uses partial risk labels and contextualized telematics data to explore major risk cohorts in data; and then training a classifier such as a Gradient Boosting Classifier.

The third part of the framework is a driving risk prediction process, which uses contextualized telematics data as input and labels a driver by an appropriate risk cohort label.

The main contributions of this section are summarized as follows:

- We propose a new process to contextualize telematics data by employing various sources of contextual data including road type and road shape.
- We propose a data-driven process to refine risk labels and build a classifier based on the contextualized telematics data and the refined label set.
- We describe a driving risk cohort prediction process to be employed for applications such as usage-based-insurance and driver coaching.

6.2 Problem Statement

Let us suppose that telematics data is composed of trajectories. A trajectory T is a time-ordered sequence of data points $\langle p_1, p_2, \ldots, p_{|T|} \rangle$. We represent each data point p by a tuple $\big(t, Speed, Heading, Lat, Lng\big)$, where t is a timestamp, $Speed$ is the ground velocity of a vehicle (in km/h), $Heading$ is the heading (or bearing) of a vehicle (number between 0 and 359 degrees[35]), and Lat and Lng represent latitude and longitude. We can then formulate the risk prediction problem as follows:

Given:

- A set of drivers $\mathcal{D}$, with
- The *observed risk score* for each driver $D \in \mathcal{D}$ denoted by R_D. Here, R_D represents the total number of traffic accidents and violation records for driver D in the past five years.

[35]An H, a heading value, of 0 points to due *north* while a value of 180 points to due *south*.

- A set of trajectories $\mathcal{T}_D$ for each driver $D \in \mathcal{D}$.

Build:

- A contextualized telematics representation $\mathcal{C}_D$ for driver $D \in \mathcal{D}$ based on $\mathcal{T}_D$ and the available contextual information (e.g., road shape, road type, and time information).

- A classifier $\mathbb{C}$ to predict a risk cohort label $\mathcal{L}_D$ for driver $D \in \mathcal{D}$, using her contextualized telematics representation $\mathcal{C}_D$. Here, risk cohorts are categorical labels such as *low-risk* and *high-risk*.

Objective:

- Minimize the average observed risk score for the low-risk cohort and maximize it for the high-risk cohort.

6.3 Related Work

Driving risk prediction has been a topic of much research over the past few decades. While earlier studies mostly employed demographic information (e.g., driver's age, gender, marital status, car's age, and car's value) [3, 6, 8, 10, 11, 12, 15], the recent studies have mostly used telematics data in addition to demographic information for driving risk prediction [57, 69, 74, 134, 125, 135, 137, 129, 141]. In this section. we provide an overview of related work by categorizing them based on the type of their input data.

6.3.1 Risk Prediction based on Demographic Data

A large body of research on driving risk prediction has only employed demographic data. Peck and Kuan [3] proposed a multi-regression model to study the impact of different factors on predicting future accidents for individual drivers. They used data from the past three years on accidents and violations, as well as demographic, annual mileage, type of the license, etc. to predict the number of future accidents for a large group of drivers in California. Important findings were that prior citation frequency, territorial accident rate, and prior accident frequency were the top three important factors in predicting future accidents. In another study, Gebers and Peck [6] examined the risk of older drivers through several experiments. They found that traffic conviction record for older drivers is a stronger

predictive factor for future accidents in comparison to younger drivers. Lourens et al. [11] studied the impact of several factors on the risk of involvement in accidents. Using single and multivariate regression analyses, they found that annual mileage, previous traffic conviction records, age, age-citation[36], and gender-education to be significant predictor factors of risk. Separating experienced and non-experienced drivers in studying the impact of previous traffic convictions on future crashes or serious violations was examined by Elliot et al. [12]. They found that a serious offense in the previous year doubled the odds for a serious offense in the subsequent year, and a previous at-fault record increases the odds by 50% for new at-fault crashes. A common aspect of all of these work is that they used previous conviction records, demographic information, or questionnaire surveys to describe drivers, and then performed correlation or prediction analyses based on such representations. However, the intrinsic (or naturalistic) behavior of a driver may not be well represented in such information. In fact, a proper and fair prediction of driving risk may not be achieved when there is no telematics data available. This is because of inherent biases in applying predictions for a cohort (e.g., novice versus experienced drivers) to individual drivers, and inadequate information about the past history of drivers (i.e., previous traffic conviction records). Additionally, a precise case-by-case prediction of driving risk was not possible using these traditional approaches, due to lack of information on individual drivers. Consequently, most of these traditional studies focused on analysis and prediction for cohorts of drivers [6, 8, 10, 11, 12].

6.3.2 Risk Prediction based on Telematics Data

Using telematics data alone for driving risk prediction is a recent trend given notable attention in the past few years. Joubert et al. [99] proposed using accelerometer (collected at a fine-grained frequency rate of 50 Hz) and speed data as the telematics data for risk prediction. Their representation method is a two-dimensional matrix only using the accelerometer data along with X and Y axes. Given a set of matrices for drivers, they calculated a distribution of values for different matrix cells and defined several risk categories based on these distribution values. In addition to acceleration data, they also used speed and road type information to create matrices that represent acceleration-based driving behavior

[36]This means to consider a driver's age and citation history together as a single factor.

for various types of road and speed ranges. While this work is an example of a context-aware, telematics-based driving risk prediction solution, their usage of telematics and contextual data was limited. In another recent study, Hu et al. [129] used telematics data to predict the future risk of a crash. Given telematics data in terms of trajectories, they performed map-matching to use speed-limit and expected traffic speed data. Then, they extracted patterns from these trajectories that included "fast-acceleration", "hard-braking", and "speeding". By analyzing these patterns in a specific context, they employed regression-based models to predict the future risk of a crash. Although the usage of contextual data is a strong point, the authors did not use finer-grained telematics data. Using accelerometer data to predict driving risk by using a convolutional neural network (CNN) model was proposed by Wang et al. [134]. From raw accelerometer data, they create a scatter map of all observed values for each instance of the data (e.g., two hours of driving), and then convert the scatter map into a 2D image. Next, using a CNN-based auto-encoder, they first initialize their risk prediction pipeline, then use encoder weights to train a second CNN network that makes risk prediction as a binary label, where 0 means risky (or having at least one traffic violation). In a recent study, Zhang et al. [121] proposed a model to create *state graphs* to represent telematics data. Then, the similarity between an aggregate state-graph (created from telematics data collected for a large group of drivers) and an individual state-graph (created for an individual) would reveal any abnormality of behavior for the individual driver. The authors also proposed models to label the type of abnormality by pre-defined labels such as rapid-acceleration, sudden-braking, and rapid-swerving. Similarly, Wang et al. [135] proposed a solution that first converts telematics data into a *state-graph* representation that they feed into an auto-encoder neural network model to derive a condensed representation. They use the condensed representation to make a prediction on driving risk using a Support Vector Regression model. An interesting aspect of this study is the use of expert knowledge to label drivers as risky or safe; however no details about the labeling process were reported in the paper.

Using a dataset which is called the *100-car naturalistic driving study* (NDS) [26], Guo and Fang [57] performed risk assessment for individual drivers, based on crash and near-crash (CNC) events, as well as critical-incident events (CIE). Using a negative binomial regression model, they estimated the number of CNC events and found CIE, age, and personality

of drivers[37] to be the important risk factors. Using K-Means clustering, they performed clustering of CNC rates, and identified three clusters of low, moderate, and high risk drivers. Finally, using a logistic regression classifier, they classified drivers into different classes of risk, using identified important risk factors. Using the same dataset, Klauer et al. [69] studied the impact of secondary tasks on the risk of crash or near-crash events. These tasks are dialing a cell phone, reaching for a cell phone, sending/receiving a text message, reaching for an object rather than cell phone, looking at a roadside object, or eating [26]. Their analyses and results demonstrated that the risk of crash or near-crash among both novice and experienced drivers significantly increased with involvement in many of the secondary tasks. Another study based on the NDS dataset by Chen et al. [139] sought to predict near-crash and crash events, using an auto-encoder model to find the optimal window-size to read the input data, and another deep-neural-network architecture being used to predict CNC events.

6.3.3 Risk Prediction based on Demographic and Telematics Data

The last group of studies employed both telematics and demographic information for driving risk prediction. Gao et al. [125] proposed a framework to employ telematics data in terms of distribution matrices of speed and acceleration values. After representing telematics data in this way, they examined several approaches to condense the telematics representation. Finally, they created a Poisson General Additive Model (GAM) to predict the number of insurance claims; and used condensed telematics data, driver's age, and vehicle's age as input parameters to estimate the lambda parameter of the GAM model. In another study, Gao et al. [126] improved on their previous work by representing telematics data with a wider range of speed values (0 to 80 km/h), and extended their demographic factors to *average monthly drive time*, age, gender, living region, and car's age, to predict the frequency of claims for different drivers using a GAM model. The authors also concluded that speed in lower ranges (i.e., 0 to 20 km/h) provided better predictivity for driving risk. Ayuso et al. [137] proposed leveraging telematics and demographic data to predict the frequency of insurance claims. Their telematics information included several coarse-grained factors such as annual distance driven, distance driven at night, distance over the speed limit, and

[37]A nominal attributed with five possible labels: neuroticism, openness to experience, extroversion, agreeableness, and conscientiousness.

urban distance. They used a Poisson regression model for prediction, and concluded that age, type of vehicle, distance per year, the distance over the speed limit, and urban distance per year are the most significant predictive factors. In another study, Guillen et al. [141] propose a Zero Inflated Poisson regression model to perform future claim prediction using demographic and telematics factors. The usage of this specific Poisson model provides an important bias correction, in that a model does not predict someone as being riskier only because he/she tends to drive more. In terms of telematics, they employed several coarse-grained factors such as distance driven per year over the speed limit, urban distance per year, and distance per year. Verbelen et al. [133] proposed to use a similar set of telematics and demographic data attributes to predict the frequency of claims. Their telematics data mostly comprise coarse-grained attributes such as yearly distance, number of trips, average distance per trip, and coverage of different road types by distance. They employed regression-based models to predict the frequency of claims, and found that a combination of telematics and demographic features provide the best results. Similarly, He et al. [128] proposed a comprehensive framework to leverage telematics and demographic data for risk prediction. Their framework first extracted fine-grained information from trajectories in terms of driving state variables (e.g., sharp-turn, lane-change, abnormal acceleration/deceleration, and speeding with respect to speed-limit data). Then, they used a machine learning model to predict the risk of accidents for a given trajectory, and studied drivers' risk in terms of groups of drivers with similar future risk trends. A common shortcoming of these studies was the limited usage of contextual information.

6.3.4 Related Work Summary

Our risk prediction framework belongs in the second category, where we predict driving risk only using telematics data. The important differences between our work and the existing solutions are: a better use of telematics data by using finer-grained representations, employing contextual data to better represent and evaluate driving behavior in a proper context, and augmenting weak risk labels using a data-driven process before modeling and prediction. The last one is an overlooked aspect when using past traffic records (e.g., accident and claim) as ground-truth labels to train or test a risk prediction model. Note that having traffic conviction records is not a sufficient indicator of being a risky driver, and, vice versa, not having a record does not endorse safeness. This bias must be corrected prior

to using past traffic conviction records as labels if we are to build accurate risk prediction models.

Table 6.1 provides a comparative overview on some of the important studies, including our work.

Table 6.1: A comparison between existing studies on driving risk prediction, based on source of input data, usage of contextual information, granularity of prediction, and utilization of telematics data.

Study	**Demographic**	**Telematics**	**Contextual**	**Prediction Granularity**	**Telematics Representation**
Peck and Kuan [3]	Yes	No	No	Individual	N/A
Gebers and Peck [6]	Yes	No	No	Group	N/A
Rajalin et al. [8]	Yes	No	No	Group	N/A
Hu et al. [10]	Yes	No	No	Group	N/A
Lourens et al. [11]	Yes	No	No	Group	N/A
Elliot et al. [12]	Yes	No	No	Group	N/A
Gebers and Peck [15]	Yes	No	No	Individual	N/A
Joubert et al. [99]	No	Yes	Yes	Individual	Fine-grained
Hu et al. [129]	No	Yes	Yes	Individual	Coarse-grained
Wang et al. [134]	No	Yes	No	Individual	Fine-grained
Zhang et al. [121]	No	Yes	No	Individual	Fine-grained
Wang et al. [135]	No	Yes	No	Individual	Fine-grained
Zheng et al. [74]	No	Yes	No	Individual	Fine-grained
Guo and Fang [57]	No	Yes	No	Individual	Fine-grained
Klauer et al. [69]	No	Yes	No	Individual	Fine-grained
Chen et al. [139]	No	Yes	No	Individual	Fine-grained
Gao et al. [125]	Yes	Yes	No	Individual	Fine-grained
Ayuso et al. [137]	Yes	Yes	Yes	Individual	Coarse-grained
Guillen et al. [141]	Yes	Yes	Yes	Individual	Coarse-grained
Verbelen et al. [133]	Yes	Yes	Yes	Individual	Coarse-grained
He et al. [128]	Yes	Yes	Yes	Individual	Fine-grained
Our Work	No	Yes	Yes	Individual	Fine-grained

6.4 Methodology

We now propose a context-aware driving risk prediction framework that consists of three components. The first component creates a representation for telematics data in

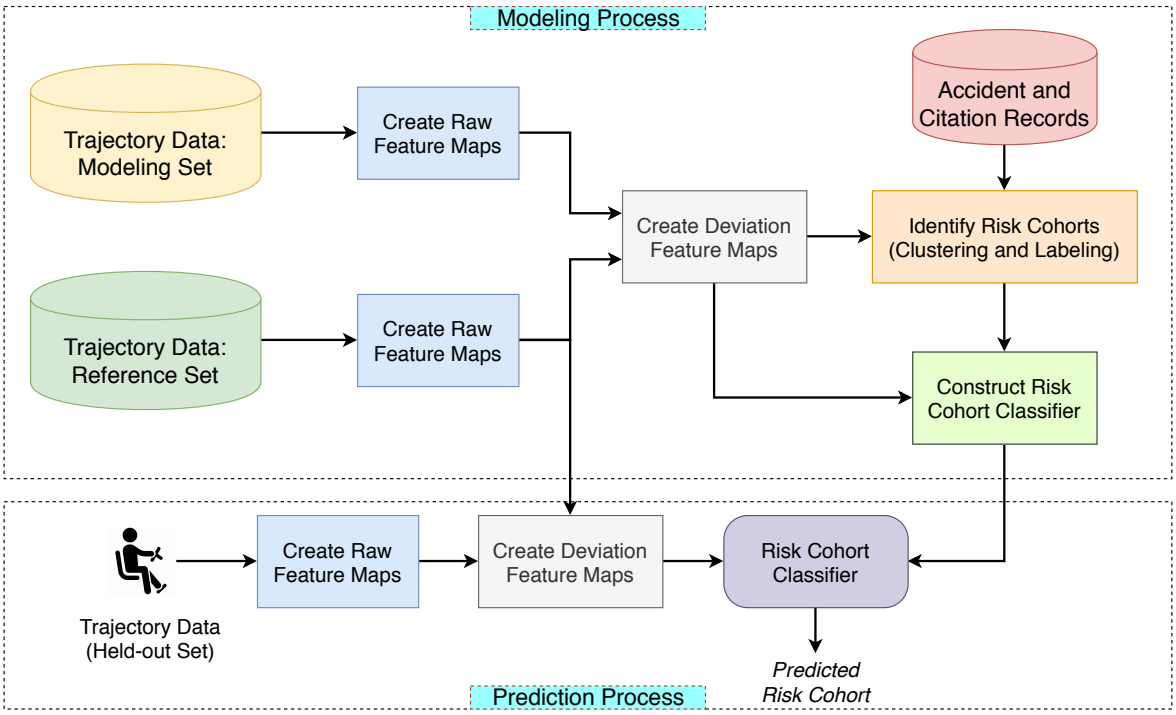


Figure 6.1: The proposed risk prediction framework. The "Modeling Process" builds contextualized telematics representation for drivers and constructs a risk-cohort classifier; and the "Prediction Process" predicts risk cohort for new drivers.

context. The second component builds a risk cohort classifier based on the contextualized telematics information plus weak risk labels. Finally, the third component provides risk cohort prediction for drivers in real-time. The big-picture of our proposal is presented in Figure 6.1, where the first two components are parts of the *Modeling Process* and the third component belongs to the *Prediction Process*.

6.4.1 Contextualized Telematics Representation

In this section, we describe how to use telematics and contextual data to build representations for drivers. The telematics data consists of trajectory datasets and the representations are built based on different contextual conditions (e.g., road type and road shape). We next describe trajectory datasets, and then describe how to build representations for each trajectory. Lastly, we will describe how to build the representation for each driver.

Modeling Trajectory Dataset

The modeling dataset consists of a large set of drivers with a minimum number of trajectories (e.g., 100 trajectories). These drivers must satisfy the following constraints: 1) each driver must own exactly one vehicle; and 2) the driver should not share her vehicle with any other driver. These constraints help ensure that trajectories collected for a vehicle had been generated by a single driver, who only drives that vehicle; thereby reducing the chance of vehicle-related bias[38]. We also refer to this set as the *observation set* in this chapter.

Reference Trajectory Dataset

The reference trajectory dataset consists of a large set of *presumably safe* drivers, also with a minimum number of trajectories for each driver. Drivers in this set have no record of accidents or traffic violations in the past 10 years, and hence can be considered to be safe drivers. Note that having any conviction records is not necessarily an indicator of being risky. Additionally, not having such records does not necessarily endorse safeness. However, we heuristically assume that a driver with no record for a *sufficiently* long period of time is a safe driver.

Raw Feature Map: Representing a Trajectory

The first step to using telematics data is to design a way to represent trajectories. We represent a trajectory using a *feature map representation*. A feature map is a matrix, such that each of its axes shows a specific attribute (e.g., speed, acceleration, or angular speed), while the content of each cell represents the normalized frequency of trajectory data points that belong to that cell. As an example, suppose that we have a feature map where its x-axis is speed and its y-axis is acceleration. A cell in this matrix is denoted by two speed and acceleration ranges (e.g., speed [5, 25] km/h, and acceleration [0.0, 0.2] m/s^2). The content of such a cell will contain the normalized frequency of trajectory data points whose speed and acceleration values fall in these ranges (with the raw frequency is normalized by the length of the trajectory). Figures 6.2, 6.3, and 6.4 are examples of raw feature maps. Figure 6.2 is a feature map for a single trajectory with speed and acceleration as the axes. Figure 6.3 shows another feature map generated for the same trajectory whose

[38]A driver might exhibit different behavior when driving different cars.

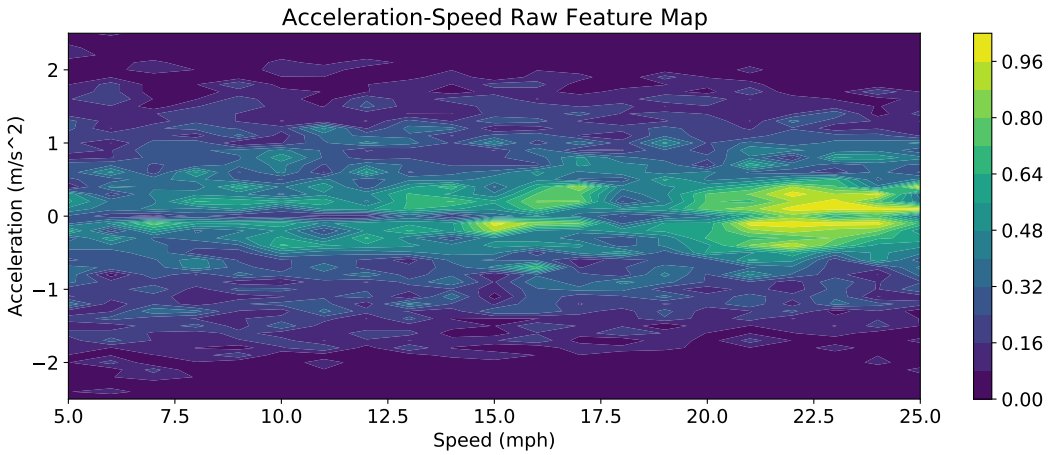


Figure 6.2: Raw feature map based on trajectory's speed and acceleration information, generated for a single trajectory.

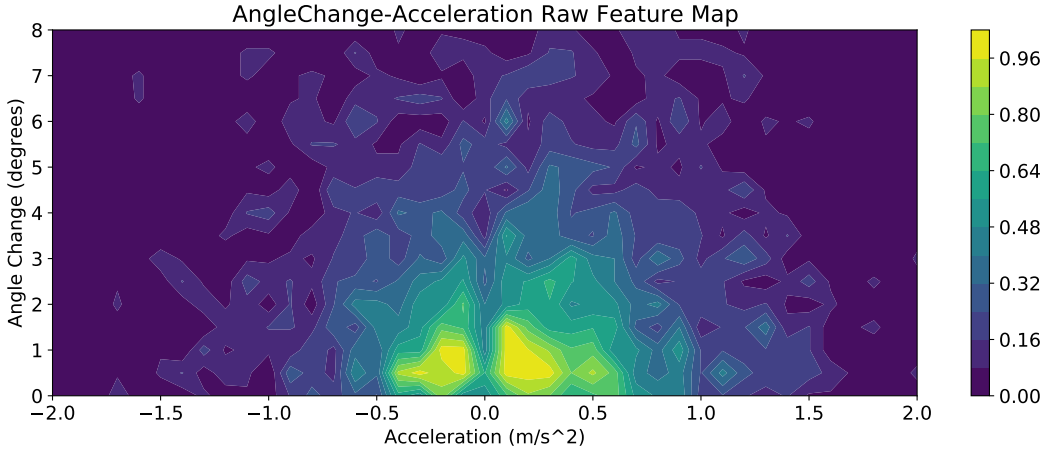


Figure 6.3: Raw feature map based on trajectory's acceleration and angle-change information, generated for a single trajectory.

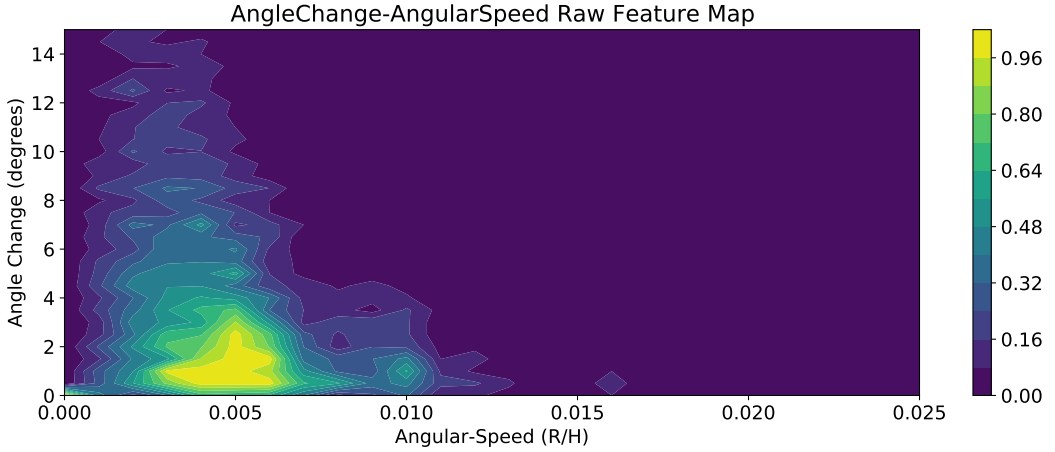


Figure 6.4: Raw feature map based on trajectory's angular-speed and angle-change information, generated for a single trajectory.

axes are acceleration and angle-change (or change of heading). Finally, Figure 6.4 shows another view of the same trajectory with axes as angular-speed and angle-change. Note that the last two raw feature maps are generated only for *turn-segments* extracted from the represented trajectory, and are examples of utilizing contextual information to generate the representation for a trajectory. More details on our turn detection process are provided in Section 6.4.4.

Deviation Feature Map: Representing a Driver

Given a set of raw feature maps generated for a set of trajectories taken from a driver d (sampled from the observation set), we present our novel process to create a new representation for driver d, termed a *deviation feature map*, and use it for predicting whether the driver is safe or risky. The main steps of this process are described by Algorithm 8. For this process, we randomly divide our reference set into *base* and *control* sets; such that the former set contains trajectories taken from 70% of the presumably safe drivers, and the latter contains trajectories taken from the rest of drivers. The process starts with generating raw feature maps for trajectories in the base set (lines 1–7). Then, we generate a *histogram matrix* that shows the distribution of raw feature maps in the base set (line 9). The next step is to generate raw feature maps, and then a histogram matrix for each driver in the control set (lines 11–20). Then, we obtain the difference between the histogram created for each driver in the control set, and the histogram created for the whole base set (lines 22–26), and summarize these differences to obtain a single matrix that we term the *natural deviation* (line 27). Next, we repeat the same process for the observation set by creating raw feature maps (lines 29–33), and then generating a histogram matrix for driver d (line 34), finally obtaining the difference between the matrix generated for driver d and the base histogram matrix (line 35). The output of this algorithm is the difference between the natural deviation and the observed deviation, which we term the *deviation feature map* for driver d (line 36).

The functions used in Algorithm 8 are described below.

- Function *build_raw_feature_map()*: This function creates a raw feature map for an input trajectory. Examples of such a feature map are shown in Figures 6.2, 6.3, and 6.4.

Algorithm 8: Generating the Deviation Feature Map

Input: The base set B, the control set C, and the observation trajectory set T_d

$\mathrm{B}_{FM} = [\,]$ ▷ Set of raw feature maps for the Base set B
for *driver* k ***in*** B **do**
  for *trajectory* t ***in*** T_k **do**
    $fm =$ build_raw_feature_map (t)
    $\mathrm{B}_{FM}.append\ (fm)$
  end
end
▷ Summarize raw feature maps by creating a histogram matrix for the base set
$hist\_matrix_{\mathrm{B}} = obtain\_histogram\_matrix\ (\mathrm{B}_{FM})$
▷ For each driver in control set, obtain raw features maps and their histogram matrix
$\mathrm{C}_{HM} = [\,]$ ▷ Set of histogram matrices for the Control set B
for *driver* $k \in$ C **do**
  $k_{FM} = [\,]$
  for *trajectory* $t \in \mathrm{T}_k$ **do**
    $fm =$ build_raw_feature_map (t)
    $k_{FM}.append\ (fm)$
  end
  $hist\_matrix_k = obtain\_histogram\_matrix\ (k_{FM})$
  $\mathrm{C}_{HM}.append\ (hist\_matrix_k)$
end
▷ Obtain the natural deviation by summarizing deviation of the control set from the base set
$\mathrm{C}_{diff} = [\,]$
for *hist_matrix* $hm \in \mathrm{C}_{HM}$ **do**
  $diff = obtain\_difference\ (hist\_matrix_{\mathrm{B}}, hm)$
  $\mathrm{C}_{diff}.append\ (diff)$
end
$natural\_deviation = summarize\ (\mathrm{C}_{diff})$
▷ Generate the deviation feature map for driver d, based on the natural deviation
$d_{FM} = [\,]$ ▷ Set of raw feature maps for the observation trajectory set
for *trajectory* $t \in \mathrm{T}_d$ **do**
  $fm =$ build_raw_feature_map (t)
  $d_{FM}.append\ (fm)$
end
$hist\_matrix_d = obtain\_histogram\_matrix\ (d_{FM})$
$observed\_deviation_d = obtain\_difference\ (hist\_matrix_{\mathrm{B}}, hist\_matrix_d)$
$deviation\_feature\_map_d = difference(observed\_deviation_d, natural\_deviation)$

Output: $deviation\_feature\_map_d$

- Function *obtain_histogram_matrix()*: Given a set of raw feature maps, we summarize them by obtaining a histogram that shows the distribution of values for each cell. In other words, if we consider a feature map as a matrix, for a set of such matrices we can obtain the distribution of values for each cell across all the input matrices. In this way, we can derive one histogram per each cell to summarize values that we observe for that cell across all the feature maps. We call such a three-dimensional representation a *histogram matrix*.

- Function *obtain_difference()*: This function obtains deviation from the base histogram matrix. Here, we can employ any method to obtain deviation. For instance, one can use *Kullback–Leibler divergence* [1] for this purpose. Note that this function uses two matrices as input, where each matrix comprises a set of probability distributions (i.e., one histogram per cell). The final deviation is not just a single value, but a two dimensional matrix where each cell contains the deviation between the probability distributions for the corresponding cells in the input matrices.

- Function *summarize()*: We summarize deviation matrices of drivers in control set using this function. This function may be implemented using a *summation*, *average*, or any other aggregation function.

- Function *difference()*: This function measures the difference between the natural deviation (i.e., output of the *summarize* function), and the observed deviation for driver d, to deliver a deviation feature map for driver d.

Several examples of deviation feature maps generated for a driver are presented by Figures 6.5, 6.6, and 6.7. Note that the last two maps are generated based on turn segments extracted from the trajectories of the driver. The number of deviation feature maps to represent a driver depends on the number of raw feature maps to represent each trajectory. Further details on various types of raw feature maps used in this work are provided in Section 6.6.1.

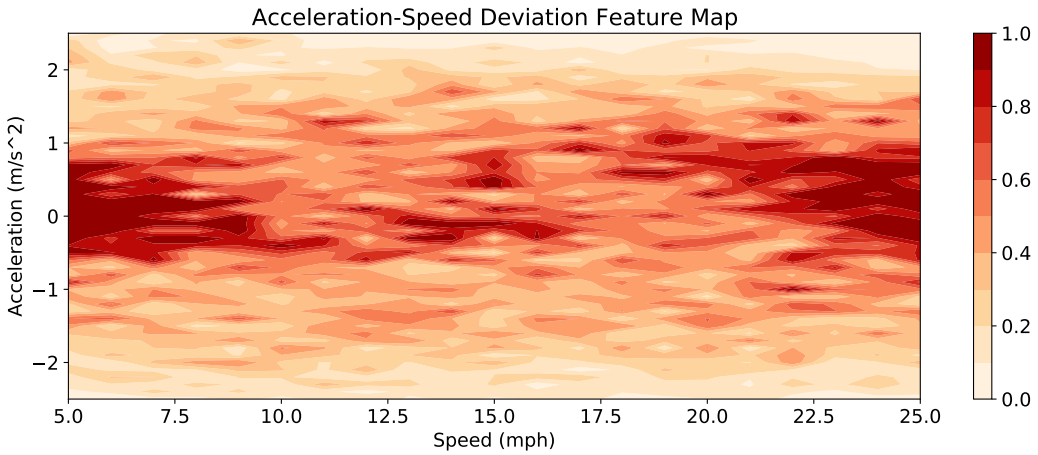


Figure 6.5: Deviation feature map based on speed and acceleration information, generated for a single driver.

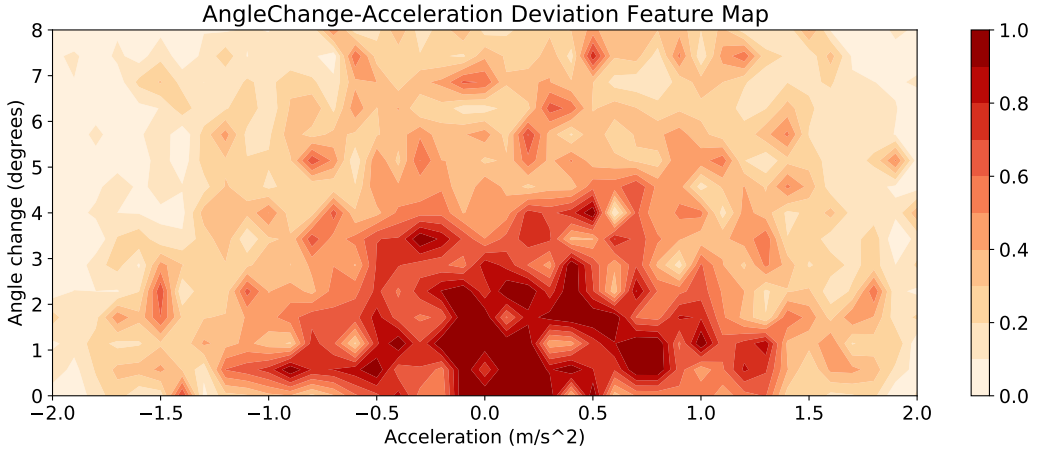


Figure 6.6: Deviation feature map based on acceleration and angle-change information, generated for a single driver.

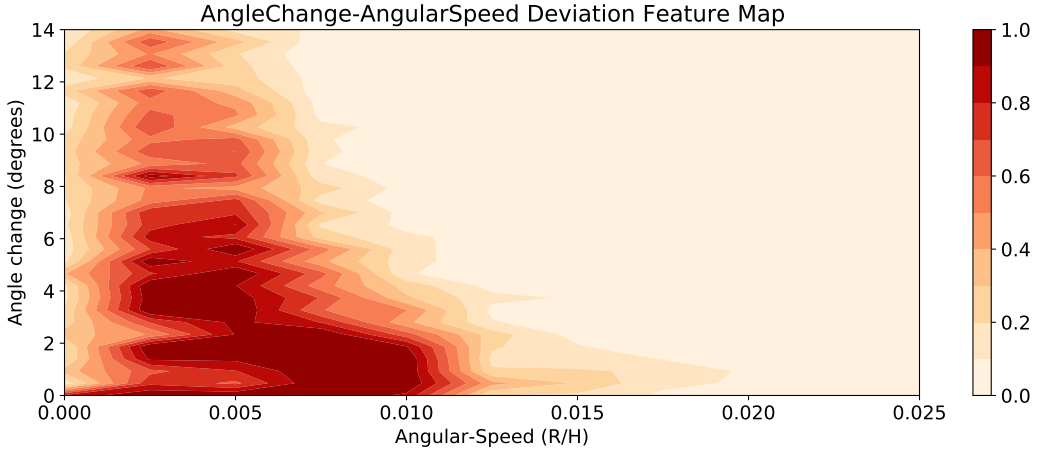


Figure 6.7: Deviation feature map based on angular-speed and angle-change information, generated for a single driver.

6.4.2 Risk Cohort Classifier

As shown in Figure 6.1, the main outcome of the modeling process is a risk cohort prediction classifier. We describe the different requirements and steps of building such a classifier in the following sub-sections.

Accident and Citation Records

This set contains traffic accident and citation records for each driver in the modeling set. In terms of accidents, we only consider the *at-fault* cases. For each driver, we only keep their last five years of accident and citation records.

Identifying Risk Cohorts

This step aims to refine weak risk labels using contextualized-telematics data and a data-driven process. We employ Algorithm 9 to identify risk cohorts for drivers in the modeling set. Each driver is represented by a set of $\mathcal{D}$ deviation maps. We use each of the deviation maps to cluster the drivers (line 3). After each clustering, we perform a post-processing step to associate each cluster with a risk cohort label (line 4). To do this, we employ the partial (or weak) ground truth risk label data (i.e., traffic conviction records), obtain the average number of records in each cluster, and then associate each cluster with a categorical label (e.g., low-risk and high-risk)[39]. Then, for each driver, we update their label set by adding the newly obtained cohort label (lines 5–7). Finally, we find the most frequent label for each driver given a confidence coefficient $\mathcal{C}$ (lines 9–16) as a threshold. Based on this threshold, we identify those drivers who have been confidently assigned to a specific risk cohort. As an example, if a driver is assigned to the *low-risk* cohort (say) 80% of the times, we label her as a *low-risk* driver. If a driver found to be not confidently assigned to any of the cohorts, then her final label would be $None$. The output of this algorithm would be a subset of drivers in the modeling set with a valid final risk cohort label.

While any clustering algorithm may be used in the *do_clustering()* function, *K-means* is a reasonable choice given that the number of clusters must be specified beforehand. In terms of number of clusters, we can pick any value greater than 1, depending on the driving context and application requirements. In this work, we use a value of 2 to test our framework

[39]For instance, if we define our labels as low-risk and high-risk, then the cluster with smaller average of conviction records per driver will be labeled as low-risk, and the other cluster will be labeled as high-risk.

Algorithm 9: Identifying Risk Cohorts

Input: Modeling set drivers $\mathcal{M}$, Deviation feature maps $\mathcal{D}$, number of cohorts k, traffic conviction records $Recs$, and labeling confidence $\mathcal{C}$.

```
driver_to_cohort_label = {}
for deviation_map D ∈ 𝒟 do
    res = do_clustering (ℳ_D, k) ▷ ℳ_D represents the modeling set drivers by deviation map D
    cohort_label = clustering_label_to_risk_cohort_label (res, Recs)
    for driver d ∈ ℳ do
        driver_to_cohort_label [d].append (cohort_label [d])
    end
end
final_labeled_set = {}
for driver d ∈ ℳ do
    labels = driver_to_cohort_label [d] ▷ Labels assigned to driver d during clusterings
    final_label = find_frequent_label (labels, 𝒞)
    if final_label ≠ None then
        final_labeled_set [d] = final_label
    end
end
```

Output: *final_labeled_set*

and derive preliminary results. Also, we determine the labeling confidence $\mathcal{C}$ based on the number deviation maps to represent each driver, because this value determines the number of potential labels for each driver. For instance, if we use 10 deviation maps for each driver, then we can choose any confidence threshold from the following set: $\langle 0.5, 0.6, 0.7, 0.8, 0.9 \rangle$.

Constructing the Risk Cohort Classifier

The last step of the modeling process is to build a classifier that uses deviation maps as input to predict the risk cohort label. The input of this part is the set of labeled drivers and their deviation maps obtained from the previous step. Any off-the-shelf classifier may be used to predict the risk cohort. The choice of the classifier depends on the size of the data and the complexity of the prediction task.

It is worth noting that *refining risk labels* before building the classifier is an important aspect of our framework. To demonstrate its importance, suppose that we build a classifier to predict driving risk using telematics data as input and using past traffic records as risk

labels. Although this seems a straightforward predictive task, it may not result in any satisfactory outcome, due to *unreliability* of the original risk labels. An important practical observation here is that if someone only has records of citation or accident it does not mean that she is a risky driver; and only because someone does not have such records does not necessarily mean that she is a safe driver. Consequently, we need a process to refine our labels prior to building a risk cohort classifier, similar to one that we propose in this work. Our label refinement process primarily uses telematics data, with traffic citation and accident records only as *partial* ground-truth data.

6.4.3 The Prediction Process

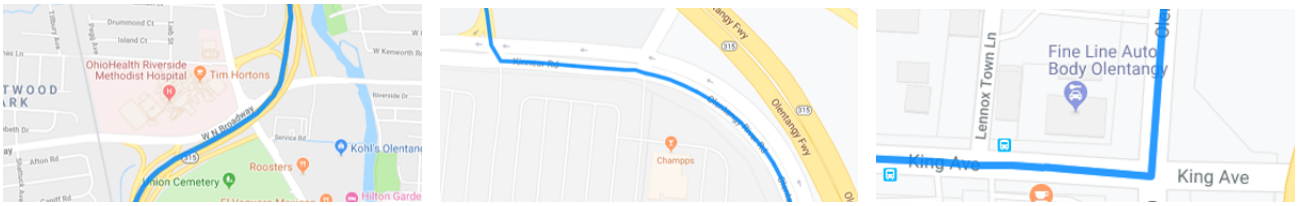
In order to employ the proposed framework for the real-world applications, we describe a *prediction process*. Given a set of trajectories for a driver whom we want to perform risk prediction for, the important steps of this process are as follows:

- Creating raw feature maps: This step is the same as what that we described in the last section to generate feature maps for an input trajectory.
- Creating deviation feature maps: This step is similar to what that we described by Algorithm 8.
- Risk prediction using risk cohort classifier: We employ the classifier that we obtained as outcome of the modeling process to classify a driver to the most relevant risk cohort.

6.4.4 Complementary Material: Turn Detection

One of the important steps in contextualizing telematics data is in studying driving behavior with respect to road shape. Turn segments are interesting road shape patterns that provide useful insights when employed for driving behavior analysis. A reasonable expectation is that risky drivers perform differently than safe drivers on sharp or even smooth turns. Figure 6.8 shows several examples of sharp and smooth turns.

The question is how should we identify these patterns in a trajectory? We propose an intuitive solution (see Algorithm 10) for turn detection. This algorithm scans a trajectory and calculates the heading difference between every two consecutive data points. If the difference is larger than or equal to a pre-defined threshold (i.e., S_{HC}), and the recorded



(a) Smooth turn (b) Sharp and smooth turn (c) Sharp turn

Figure 6.8: Examples of *smooth* and *sharp* turn segments in trajectories (here trajectories are specified by blue line).

speed of the current data point is larger than or equal to another threshold (i.e., M_S), we add the current data point to the current turn candidate segment (lines 5–9). A *violation* could happen if the heading difference is smaller than S_{HC} or if the recorded speed of the current data point is smaller than M_S. Here *violations* records the number of times that at least one of the mentioned conditions are violated. There is a maximum threshold for the number of violations, and when it reached, we check if the current turn segment candidate satisfies the conditions to be added to the set of final turn segment candidates (i.e., *turn_candidates*, lines 16–22). The main requirements here are: 1) the change in vehicle's direction should be higher than a threshold M_{HC} (lines 17 and 19), and 2) the average speed of a vehicle during a turn segment should be higher than a threshold M_{AS} (lines 18 and 19). The latter condition is there to avoid noisy GPS data when the vehicle is stopped or when it is moving at a very low speed. Any turn segments that are separated by a short gap are merged (line 29).

Table 6.2 shows the best parameter (empirically determined) values for Algorithm 10. To test our turn detection algorithm, we randomly selected a set of 100 trajectories sampled from 5 different cities (i.e., Atlanta, Columbus, Memphis, Philadelphia, and Pittsburgh), with 20 trajectories for each city. We applied our algorithm on these trajectories, and manually checked each case on a map. Table 6.3 summarizes the results of this study. Here we reported the *accuracy*. On average, our algorithm detected 9 turn segments per trajectory. Also, we found that about 96% of errors to be due to low vehicle speed, hence GPS drift.

Algorithm 10: Turn Detection Algorithm

Input: Trajectory T, minimum change of heading M_{HC}, significant heading change S_{HC}, minimum speed M_S, minimum average speed M_{AS}, maximum violations M_v, and maximum gap to merge turn segments *Gap*

```
turn_candidates = [ ]
current_candidate = [ ]
violations = 0
for i = 1 to length (T) do
    hc = heading_change (T[i − 1].heading, T[i].heading)
    if hc ≥ S_HC and T[i].speed ≥ M_S then
        current_candidate.append (T[i])
        violations = 0                          ▷ Reset the violation count
    end
    else
        violations+ = 1
        if violations ≤ M_v then
            current_candidate.append (T[i])
        end
        else
            if length (current_candidate) > 1 then
                mhc = obtain_max_heading_difference (current_candidate)
                avs = obtain_average_speed (current_candidate)
                if mhc ≥ M_HC and avs ≥ M_AS then
                    turn_candidates.append (current_candidate)
                end
            end
            current_candidate = [ ]
            violations = 0
        end
    end
end
                    ▷ The following step merges consecutive turn segments
final_turn_segments = merge_consecutive_turns (turn_candidates, Gap)
```

Output: *final_turn_segments*

Table 6.2: The best parameter values for Algorithm 10

Parameter	Description	Value
M_{HC}	Minimum change of angle/heading during a turn segment	40 degrees
S_{HC}	Minimum change of heading between consecutive data points to be considered as a significant angle change	6 degrees
M_S	Minimum speed of each data point within a turn segment	5 km/h
M_{AS}	Minimum average speed for a turn segment	10 km/h
M_v	Maximum number of violations allowed for constraints on speed and heading attributes (i.e., S_{HC} and M_S) in a turn segment	2
Gap	Maximum gap between two turn segments to be considered for merge	1

Table 6.3: Evaluation of the proposed turn detection algorithm based on a set of 100 trajectories.

# Trajectories	# Detected Turns	# Correctly Detected	Accuracy
100	886	861	97.2%

6.5 Dataset

We employed multiple and extensive datasets to build and evaluate the proposed risk prediction framework. Data were collected from the *CAN-bus*[40] and other sensors including GPS, accelerometer, and magnetometer, and for five cities[41]. In addition to trajectory information, we have the demographic data for drivers and also their history of traffic accidents or conviction records. In building the framework, we divided our data into multiple sets, as follows.

- **Modeling Set**: This is the set that we use it for the modeling process to create the risk cohort prediction classifier. In this set, we have about 4,000 drivers, none of whom shared their vehicle with any other driver, and who each owned exactly one vehicle. For each driver, we collected exactly 100 trajectories.

[40]Controller Area Network (or CAN-bus) is a communication mechanism to transfer a variety of information related to the operation of a vehicle.

[41]Atlanta (GA), Columbus (OH), Memphis (TN), Philadelphia (PA), and Pittsburgh (PA).

- **Reference Set**: This is the set of presumably safe drivers, with no records of accidents or traffic citations from 2010 to 2019. In this set, we have 5,000 vehicles. Each vehicle could have been shared with more than one driver, but all drivers who shared the same vehicle satisfied the previous constraint. Here also we collected 100 trajectories for each vehicle.

- **Held-out Set**: This is a smaller set to evaluate the prediction process. In this set, we had about 300 drivers who owned exactly one vehicle, and who did not share their vehicle.

To form the partial ground-truth labels for drivers in the modeling and the held-out sets, we collected their traffic citation and at-fault accident records from 2014 to 2019.

6.6 Experiments and Results

In this section, we evaluate our risk prediction framework based on real-world data. First, we describe the set of raw feature maps that we created for each trajectory. Then, we present details of our experiments and results.

6.6.1 Raw Feature Maps

We represented a trajectory using 19 different feature maps. We categorized these maps in terms of 7 groups as follows.

Group 1: Acceleration-Speed

This group consists of three raw feature maps based on speed and acceleration data calculated from GPS coordinates.

- **T1**: For this case x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[5, 25]$ mph and $[-2.5, 2.5]$ m/s^2, respectively.

- **T2**: For this case x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[25, 45]$ mph and $[-2, 2]$ m/s^2, respectively.

- **T3**: For this case x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[45, 80]$ mph and $[-1, 1]$ m/s^2, respectively.

Group 2: Acceleration-Speed w.r.t Speed Limit

This group is similar to group 1, except that we employ speed limit data and categorize road segments into three categories: *residential* with speed limit less than 25 mph; *urban* with speed limit between 25 mph and 45 mph; and *highway* with speed limit between 45 mph and 90 mph. To obtain the speed limit data, we employed the Graphhopper Map-matching Library [140]. This group comprises three feature maps:

- **T4**: This case only covers residential areas, with its x-axis shows speed and y-axis shows acceleration. We define speed and acceleration ranges as $[3, 50]$ mph and $[-2.5, 2.5]$ m/s^2, respectively.

- **T5**: This case only covers urban areas, with its x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[5, 65]$ mph and $[-2, 2]$ m/s^2, respectively.

- **T6**: This case only covers highway areas, with its x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[30, 85]$ mph and $[-1, 1]$ m/s^2, respectively.

Group 3: Acceleration-Speed for Turn Segments

This group of feature maps is similar to the first group, unless that it only covers turn segments that were detected by our proposed algorithm. This group contains the following feature maps:

- **T7**: For this case x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[5, 25]$ mph and $[-3, 3]$ m/s^2, respectively.

- **T8**: For this case x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[25, 45]$ mph and $[-3.5, 3.5]$ m/s^2, respectively.

- **T9**: For this case x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[45, 80]$ mph and $[-2, 2]$ m/s^2, respectively.

Group 4: Acceleration-Speed for Turn Segments w.r.t Speed Limit

Similar to group 3, here we have feature maps created for turn segments, but with respect to the speed limit data. This group contains the following feature maps:

- **T10**: This case only covers residential areas, with its x-axis shows speed and y-axis shows acceleration. We define speed and acceleration ranges as $[3, 45]$ mph and $[-3, 3]$ m/s^2, respectively.

- **T11**: This case only covers urban areas, with its x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[5, 65]$ mph and $[-3.5, 3.5]$ m/s^2, respectively.

- **T12**: This case only covers highway areas, with its x-axis shows speed, and y-axis shows acceleration. We define speed and acceleration ranges as $[20, 80]$ mph and $[-2, 2]$ m/s^2, respectively.

Group 5: Angle_Change-Speed for Turn Segments

This group contains feature maps based on angle_change (y-axis) and speed (x-axis), collected for the turn segments. The angle_change is a feature that we calculate it based on the change of the heading between two consecutive data points in a trajectory. This group contains the following feature maps:

- **T13**: For this case x-axis shows speed, and y-axis shows angle_change. We define speed and angle_change ranges as $[5, 25]$ mph and $[0, 20$ *degrees*, respectively.

- **T14**: For this case x-axis shows speed, and y-axis shows angle_change. We define speed and angle_change ranges as $[25, 45]$ mph and $[0, 6$ *degrees*, respectively.

- **T15**: For this case x-axis shows speed, and y-axis shows angle_change. We define speed and angle_change ranges as $[45, 80]$ mph and $[0, 4$ *degrees*, respectively.

Group 6: Angle_Change-Speed for Turn Segments w.r.t Speed Limit

This group is similar to group 5, unless that we also employ speed limit information. Following is the list of feature maps in this group:

- **T16**: This case only covers residential areas, with its x-axis shows speed and y-axis shows angle_change. We define speed and angle_change ranges as $[3, 45]$ *mph* and $[0, 20$ *degrees*, respectively.

- **T17**: This case only covers urban areas, with its x-axis shows speed and y-axis shows angle_change. We define speed and angle_change ranges as $[5, 65]$ *mph* and $[0, 15$ *degrees*, respectively.

- **T18**: This case only covers highway areas, with its x-axis shows speed and y-axis shows angle_change. We define speed and angle_change ranges as $[20, 80]$ *mph* and $[0, 7]$ *degrees*, respectively.

Group 7: Angle_Change-Angular_Speed for Turn Segments

This group uses angular_speed and angle_change information from turn segments to form a feature map as follows:

- **T19**: For this case x-axis shows angular_speed and y-axis shows angle_change. We define angular_speed and angle_change ranges as $[0, 0.025]$ *radian per hour (R/H)* and $[0, 15$ *degrees*, respectively.

6.6.2 Risk Cohort Prediction Results

After representing a trajectory using the 19 different raw feature maps, we created the risk cohort classifier from the modeling and reference sets (see Section 6.4). We examined different combinations of raw feature maps, and the set of all feature maps to represent trajectories. Next we describe several settings before presenting the results.

Choice of Deviation Metric

To obtain the deviation between two probability distributions in Algorithm 8, we may employ any deviation calculation method. However, for the sake of simplicity, we simply used *Manhattan Distance*:

$$Manhattan\_Distance(D_1, D_2) = \frac{\sum_{i=1}^{i=N} |D_1(i) - D_2(i)|}{N}$$

Labeling Confidence $\mathcal{C}$

The labeling confidence in Algorithm 9 determines the final labeled set of drivers. We determine this value with respect to the number of deviation maps that we use to represent each driver[42], and also the number of risk cohorts. For example, if we represent each driver using 10 different deviation maps and the objective is to find two risk cohorts, then we use the following confidence set: $\{0.6, 0.7, 0.8, 0.9, 1.0\}$, where $\mathcal{C} = 0.6$ means that 60% of labels found for a driver must belong in the same cohort, say low-risk. As another example, if we use a set of three deviation maps to represent each driver and the objective is to find two risk cohorts, we use the following confidence set: $\{0.7, 1.0\}$. Given a confidence set C, we choose a value $\mathcal{C} \in$ C that provides the best prediction results on the held-out set.

Number of Cohorts

One of the advantages of this framework is that we may define any number of cohorts. In this work, we used two cohorts, which we referred to as *low-risk* and *high-risk*.

Clustering Algorithm

The choice of the clustering algorithm to be employed by Algorithm 9 is another customizable part of this framework. Here, we chose to use K-Means clustering, but any other method that allows the user to set the number of clusters may be used.

Classification Method

For classification we used Gradient Boosting. The input for this classifier was a deviation feature map that represented a driver. The label for each driver is obtained from the clustering (see Section 6.4).

Risk Prediction Results on the Held-out Set

We used different combinations of raw feature maps to build our risk prediction pipeline and predict the binary risk cohort label for drivers in the held-out set. For the drivers in each cohort, we represented the average cohort risk in terms of the average number of traffic citations or accident records in the past five years. Figure 6.9 shows the results of

[42]Which depends on the number of raw feature maps to represent each trajectory

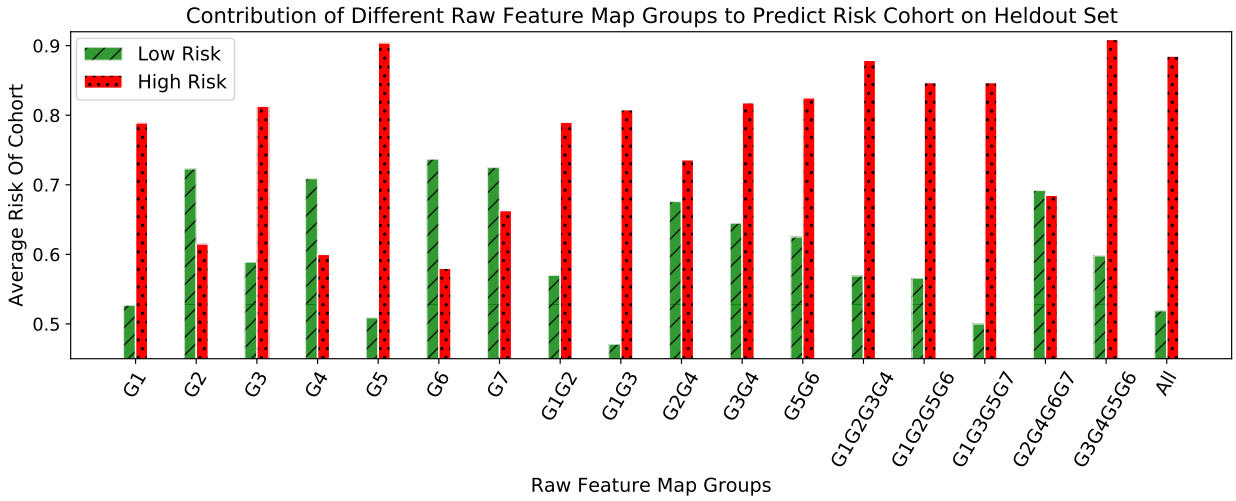


Figure 6.9: Risk cohort prediction results for drivers in the held-out set. X-axis shows the combination of raw feature maps to represent each trajectory, and Y-axis shows the average number of traffic citation or accident records for drivers in each predicted cohort.

this experiment. Table 6.4 also provides further details in terms of the size of each cohort, and the ratio of drivers with no records in the past five years in each cohort.

Based on Figure 6.9, feature maps in group 5 provided the best risk cohort prediction on the held-out set; because there is a large gap between the average risk for the low-risk cohort (specified by green) versus the high-risk cohort (specified by red). Similarly, Table 6.4 confirms this result by showing that the ratio of drivers with no record in the low-risk cohort is significantly higher than the high-risk cohort. This group of feature maps contains angle_change and speed information for the turn segments. This is a very interesting result because the raw feature maps in this group represent how a driver behaves on turn segments.

We noted that employing road type information did not yield satisfactory results (see Groups 2, 4, and 6 in Figure 6.9). This might be because of inaccuracies in the map-matching process, or using inappropriate ranges for data attributes that we employed to generate the corresponding feature maps (see Section 6.6.1).

Analysis of Table 6.4 also reveals interesting insights. First, based on our best risk cohort prediction results (i.e., group 5), we observe that about 54% of drivers were assigned to the low-risk cohort and the rest were assigned to the high-risk one. The proportion of drivers

Table 6.4: Risk cohort prediction results for held-out set. Here we have two risk cohorts: *low-risk* and *high-risk*. The ratio of drivers in a cohort with no traffic accident or citation records is shown by *no records %*. Also, average risk of each cohort shows the average number of records per driver in each cohort.

Feature Groups	**# Low-Risk Drivers**	**# High-Risk Drivers**	**Low-Risk no records %**	**High-Risk no records %**	**Avg. risk of Low-Risk**	**Avg. risk of High-Risk**
G1	112	185	67.86%	58.38%	0.527	0.789
G2	206	91	61.17%	63.74%	0.723	0.615
G3	163	134	62.58%	61.19%	0.589	0.813
G4	247	50	62.35%	60.00%	0.709	0.600
G5	**161**	**136**	**67.08%**	**55.88%**	**0.509**	**0.904**
G6	209	88	63.16%	59.09%	0.737	0.580
G7	131	166	56.49%	66.27%	0.725	0.663
G1G2	135	162	65.19%	59.26%	0.570	0.790
G3G4	220	77	64.09%	55.84%	0.645	0.818
G5G6	200	97	63.00%	59.79%	0.625	0.825
G1G2G3G4	181	116	65.19%	56.90%	0.569	0.879
G1G2G5G6	166	131	65.06%	58.02%	0.566	0.847
G1G3G5G7	134	163	65.67%	58.90%	0.500	0.847
G2G4G6G7	224	73	63.39%	57.53%	0.692	0.689
G3G4G5G6	209	88	64.59%	55.68%	0.598	0.909
All	158	139	65.19%	58.27%	0.519	0.885

with no traffic accident or citation records in the low-risk cohort was significantly higher than this value for the high-risk cohort, however, there were certainly drivers in this cohort with some accident and citation records. This confirms our initial presumption that *having traffic citation or accident records does not necessarily indicate riskiness, and not having such records does not endorse safeness*. Thus, when we build our risk cohorts, we should expect to see some of the drivers with no traffic conviction records to be identified as risky, and some of the drivers with traffic conviction records to be identified as safe. However, we may expect that a majority of drivers with traffic conviction records belong to a high-risk cohort.

6.7 Summary and Conclusions

In this chapter, we presented a new driving risk prediction framework based on telematics and contextual data. The proposed framework consists of a modeling and a prediction process. The modeling process builds a risk cohort classifier based on contextualized telematics data. The important steps of the modeling process are building contextualized

views of trajectories, transforming the raw views to deviation-based views for drivers, employing a data-driven process to refine risk labels, and building a risk cohort classifier based on contextualized telematics data and refined risk labels. The prediction process performs driving risk cohort prediction based on contextualized telematics data by using the previously built risk cohort classifier. One important aspect of this new framework is the *risk label refinement*, which employs a novel data-driven process to refine risk labels and prepare data to train the risk cohort classifier in the modeling process.

In terms of contextual data, we mostly employed road type and road shape information. This may be improved by using other contextual information such as time, traffic, and weather data. In so doing, we can create more raw feature maps and, consequently, more deviation maps to represent drivers. In other words, this framework allows any other contextualized view of a trajectory to be used as input. Further, the framework provides the ability to adjust the number of risk cohorts for different locations. For instance, one could consider only two cohorts for region R_1, but three cohorts for region R_2. Finally, we note that the framework also allows us to choose the clustering algorithm, classification method, and formulation of deviation calculation method.

Chapter 7: Conclusions

With the availability of telematics (i.e., driving data collected using various sensors) and contextual data (e.g., traffic data, weather data, and characteristics of road-network) in large scale, a variety of applications can be defined to use such data and derive insights. The focus of this dissertation is on driving behavior analysis and risk prediction. Thus, we seek to design a framework to use telematics and contextual data to provide reasonable predictions on driving risk. The important objectives in designing such a framework are 1) properly utilizing contextual data to study and analyze a driver's behavior in comparison to the other drivers in the same context, and 2) properly utilizing a driver's skills and personality information to better derive risk-related insights. To achieve these objectives, we propose a comprehensive solution that includes the following components: a) characterizing driving context based on driving behavior and contextual data; b) characterizing driving style based on telematics data; and c) context-aware driving risk prediction based on telematics and contextual data using macro- and micro-level solutions.

7.1 Summary of Key Contributions

7.1.1 Characterizing Driving Context

To explore characteristics of driving contexts, we propose two solutions, one based on *trajectory segmentation and causality analysis* and the other based on *geo-spatiotemporal pattern discovery*. The first solution uses telematics data as input, explores meaningful driving patterns, and performs causality analysis to identify any relationship between observed driving patterns and extrinsic causes (e.g., traffic congestion, weather conditions, and characteristics of road-network). The second solution uses contextual traffic and weather data as input and seeks to explore *propagation* and *influential* patterns. Propagation patterns reveal the common cascading forms of traffic or weather entities in a region, and

influential patterns study the impact of long-term traffic or weather entities on their spatial neighborhood. The output of these solutions provide valuable insights regarding different contexts as they relate to driving behavior.

7.1.2 Characterizing Driving Style

Driving style can be described as variations in driving behavior that discriminate different drivers from each other. Here the objective is to capture such variation for different drivers in terms of driving style, whereby using driving style information we can predict the identity of a driver for an input trajectory. To achieve this objective, we propose a deep-neural-network-based solution to use telematics data. In addition, to avoid any potential biases as a result of the spatial similarity between trajectories taken from the same driver, we propose to preprocess the input data by several similarity-aware sampling methods, and then use the sampled datasets to build and evaluate our model.

7.1.3 Context-aware Driving Risk Prediction

To fulfill the main objective of this research, we propose two solutions for driving risk prediction. The first solution is a macro-level driving risk prediction solution that seeks to predict the possibility of traffic accidents (as explicit indicators of driving risk) in a region during a fine-grained time interval. This solution relies on a novel deep-neural-network model that uses heterogeneous sources of contextual data (e.g., traffic, weather, road-network characteristics, and temporal data) to predict the possibility of accidents on a real-time basis. Our proposal uses easy-to-obtain but sparse data, thus it is scalable and can serve real-world applications.

The second solution is a micro-level driving risk prediction approach that uses telematics and contextual data to achieve risk prediction for individual drivers. To do this, we first build contextualized telematics representations for an input trajectory. Then, we build a risk-cohort prediction classifier using weak risk labels (i.e., traffic conviction records) and employing a data-driven process to augment risk labels. Finally, we build a risk prediction process for real-time applications. To our knowledge, this is the first proposal that maximizes the use of telematics and contextual data, and also provides data relabeling (or label augmentation) based on weak risk labels. Also, our proposal relies on GPS telemetry data that is inexpensive to collect on a large-scale.

7.2 Future Work

We propose the following directions for future research.

- **Additional types of telematics data**: Telematics data used in this research comes primarily from the CAN-bus and GPS sensor. While these sources provide useful information regarding driving behavior, we may also benefit from other sensors such as the accelerometer. High-quality accelerometer data collected on high frequency rates (e.g., 10 Hz) provides another view of driving behavior to capture various interesting patterns such as lane change and sudden jerk while driving straight. These patterns, when contextualized, provide additional insights to better analyze driving behavior and predict driving risk.

- **Additional types of contextual data**: As we have shown in this dissertation, contextual data provides valuable insights regarding driving behavior to be used for driving risk prediction. An important future direction of this research could be using other sources of contextual data to improve modeling and predictions. Examples of additional contextual data are finer-grained traffic flow information (collected using the loop-detector sensors), finer-grained weather data (e.g., precise rainfall and road surface conditions), additional characteristics of the road-network (e.g., number of lanes, precise speed-limit information, and road curvature), and demographic information (e.g., distribution of age, income, ethnicity, and crime for a region).

- **Augmenting the risk label**: A fundamental challenge with telematics-based driving risk prediction is the lack of availability of fine-grained risk labels. Usually, there is no risk data associated with individual trajectories or sub-trajectories. However, a risky driver may not represent a continuous trend of riskiness in all his/her driving trajectories, and a safe driver may not represent a continuous trend of safety as well. Thus, if we have risk data available for every trajectory or sub-trajectory, we may construct better models to predict finer-grained risk labels. This method of modeling and prediction also might be more beneficial for real-time applications to prevent dangerous driving actions that result in accidents.

- **Evaluating models in the real-world**: We mostly evaluated our models based on off-line testing scenarios and using historical data as labels, which is the typical form of evaluating machine learning models and data-science pipelines. However, to prove the applicability of a solution for real-world purposes, one future direction is to design a framework to assess the ability of different risk prediction models based on an *A/B testing* approach. In this way, we can compare different risk prediction frameworks, including the traditional one, based on real-world data in *production* to derive insights and examine different modeling strategies.

Bibliography

[1] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[2] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[3] Raymond C Peck and Jensen Kuan. A statistical model of individual accident risk prediction using driver record, territory and other biographical factors. *Accident Analysis & Prevention*, 15(5):371–393, 1983.

[4] Ohio Supercomputer Center. Ohio supercomputer center, 1987. URL: `http://osc.edu/ark:/19495/f5s1ph73`.

[5] SR Ely. Rds-alert: a drive project to develop a proposed standard for the traffic message channel feature of the radio data system rds. In *Car and its Environment-What DRIVE and PROMETHEUS Have to Offer, IEE Colloquium on*, pages 8–1. IET, 1990.

[6] Michael A Gebers and Raymond C Peck. The identification of high-risk older drivers through age-mediated point systems. *Journal of Safety Research*, 23(2):81–93, 1992.

[7] James Elander, Robert West, and Davina French. Behavioral correlates of individual differences in road-traffic crash risk: an examination of methods and findings. *Psychological bulletin*, 113(2):279, 1993.

[8] Sirpa Rajalin. The connection between risky driving and involvement in fatal accidents. *Accident Analysis & Prevention*, 26(5):555–562, 1994.

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

[10] Patricia S Hu, David A Trumble, Daniel J Foley, John W Eberhard, and Robert B Wallace. Crash risks of older drivers: a panel data analysis. *Accident Analysis & Prevention*, 30(5):569–581, 1998.

[11] Peter F Lourens, Jan AMM Vissers, and Maaike Jessurun. Annual mileage, driving violations, and accident involvement in relation to drivers' sex, age, and level of education. *Accident Analysis & Prevention*, 31(5):593–597, 1999.

[12] Michael R Elliott, Patricia F Waller, Trivellore E Raghunathan, Jean T Shope, and Roderick JA Little. Persistence of violation and crash behavior over time. *Journal of Safety Research*, 31(4):229–242, 2000.

[13] Nobuyuki Kuge, Tomohiro Yamamura, Osamu Shimoyama, and Andrew Liu. A driver behavior recognition method based on a driver model framework. *SAE transactions*, 109(6):469–476, 2000.

[14] Andrew Liu and Dario Salvucci. Modeling and prediction of human driver behavior. In *Intl. Conference on HCI*, pages 1479–1483, USA. ", 2001.

[15] Michael A Gebers and Raymond C Peck. Using traffic conviction correlates to identify high accident-risk drivers. *Accident Analysis & Prevention*, 35(6):903–912, 2003.

[16] Daniel Eisenberg. The mixed effects of precipitation on traffic crashes. *Accident analysis & prevention*, 36(4):637–647, 2004.

[17] Tony X Han, Steven Kay, and Thomas S Huang. Optimal segmentation of signals and its application to image denoising and boundary feature extraction. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 4, pages 2693–2696, USA. IEEE, 2004.

[18] Yan Huang, Shashi Shekhar, and Hui Xiong. Discovering colocation patterns from spatial data sets: a general approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(12):1472–1485, 2004.

[19] Dario D Salvucci. Inferring driver intent: a case study in lane-change detection. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48, pages 2228–2231, USA. SAGE Publications, SAGE, 2004.

[20] Li-Yen Chang. Analysis of freeway accident frequencies: negative binomial regression versus artificial neural network. *Safety science*, 43(8):541–557, 2005.

[21] Li-Yen Chang and Wen-Chieh Chen. Data mining of tree-based models to analyze freeway accident frequency. *Journal of safety research*, 36(4):365–375, 2005.

[22] Hui Yang, Srinivasan Parthasarathy, and Sameep Mehta. A generalized framework for mining spatio-temporal patterns in scientific data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 716–721. ACM, 2005.

[23] Mohammed J Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae*, 66(1-2):33–52, 2005.

[24] Aris Anagnostopoulos, Michail Vlachos, Marios Hadjieleftheriou, Eamonn Keogh, and Philip S Yu. Global distance-based segmentation of trajectories. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 34–43. ACM, 2006.

[25] Shinko Yuanhsien Cheng and Mohan M Trivedi. Turn-intent analysis using body pose for intelligent driver assistance. *IEEE Pervasive Computing*, 5(4):28–37, 2006.

[26] Thomas A Dingus, Sheila G Klauer, Vicki L Neale, Andy Petersen, Suzanne E Lee, JD Sudweeks, Miguel A Perez, Jonathan Hankey, DJ Ramsey, Santosh Gupta, et al. The 100-car naturalistic driving study, Phase II-results of the 100-car field experiment. Technical report, Virginia Tech Transportation Institute, 2006.

[27] John Krumm and Eric Horvitz. Predestination: inferring destinations from partial trajectories. In *International Conference on Ubiquitous Computing*, pages 243–260, USA. Springer, 2006.

[28] Thomas H Maze, Manish Agarwal, and Garrett Burchett. Whether weather matters to traffic demand, traffic safety, and traffic operations and flow. *Transportation research record*, 1948(1):170–176, 2006.

[29] Shirish Tatikonda, Srinivasan Parthasarathy, and Tahsin Kurc. Trips and tides: new algorithms for tree mining. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 455–464. ACM, 2006.

[30] Ciro Caliendo, Maurizio Guida, and Alessandra Parisi. A crash-prediction model for multilane roads. *Accident Analysis & Prevention*, 39(4):657–670, 2007.

[31] Neville A Stanton, Guy H Walker, Mark S Young, Tara Kazi, and Paul M Salmon. Changing drivers' minds: the evaluation of an advanced driver coaching system. *Ergonomics*, 50(8):1209–1234, 2007.

[32] Tom Brijs, Dimitris Karlis, and Geert Wets. Studying the effect of weather conditions on daily crash counts using a discrete time-series model. *Accident Analysis & Prevention*, 40(3):1180–1190, 2008.

[33] Mete Celik, Shashi Shekhar, James P Rogers, and James A Shine. Mixed-drove spatio-temporal co-occurrence pattern mining. *network*, 11:15, 2008.

[34] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[35] Amardeep Sathyanarayana, Pinar Boyraz, and John HL Hansen. Driver behavior analysis and route recognition by hidden markov models. In *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, pages 276–281, USA. IEEE, 2008.

[36] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 352–361, USA. ACM, 2009.

[37] Feng Qian, Qinming He, and Jiangfeng He. Mining spread patterns of spatio-temporal co-occurrences over zones. In *International Conference on Computational Science and Its Applications*, pages 677–692. Springer, 2009.

[38] Maike Buchin, Anne Driemel, Marc van Kreveld, and Vera Sacristán. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 202–211, USA. ACM, 2010.

[39] Mario Cools, Elke Moons, and Geert Wets. Assessing the impact of weather on traffic intensity. *Weather, Climate, and Society*, 2(1):60–68, 2010.

[40] Philippe Fournier-Viger. *Un modèle hybride pour le support à l'apprentissage dans les domaines procéduraux et mal définis*. PhD thesis, Université du Québec à Montréal, 2010.

[41] Pradeep Mohan, Shashi Shekhar, James A Shine, and James P Rogers. Cascading spatio-temporal pattern discovery: a summary of results. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 327–338. SIAM, 2010.

[42] Anna K Andersson and Lee Chapman. The impact of climate change on winter road maintenance and traffic accidents in west midlands, uk. *Accident Analysis & Prevention*, 43(1):284–289, 2011.

[43] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[44] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1010–1018. ACM, 2011.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[46] Leon Stenneth, Ouri Wolfson, Philip S Yu, and Bo Xu. Transportation mode detection using mobile phones and gis information. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 54–63, USA. ACM, 2011.

[47] Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. *Geolife GPS trajectory dataset - User Guide*. 2011. URL: `https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/`.

[48] CY Goh, J Dauwels, N Mitrovic, MT Asif, A Oran, and P Jaillet. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 776–781, USA. IEEE, 2012.

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[50] José Onate López, Andrés C Cuervo Pinilla, et al. Driver behavior classification model based on an intelligent driving diagnosis system. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 894–899. IEEE, 2012.

[51] Pradeep Mohan, Shashi Shekhar, James A Shine, and James P Rogers. Cascading spatio-temporal pattern discovery. *IEEE Transactions on Knowledge and Data Engineering*, 24(11):1977–1992, 2012.

[52] Costas Panagiotakis, Nikos Pelekis, Ioannis Kopanakis, Emmanuel Ramasso, and Yannis Theodoridis. Segmentation and sampling of moving object trajectories based on representativeness. *IEEE Transactions on Knowledge and Data Engineering*, 24(7):1328–1343, 2012.

[53] Tijmen Tieleman and Geoffrey Hinton. Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. Lecture 6.5-rmsprop.

[54] Joaquién Abellán, Griselda López, and Juan De OñA. Analysis of traffic accident severity using decision rules via decision trees. *Expert Systems with Applications*, 40(15):6047–6054, 2013.

[55] Ruth Bergel-Hayat, Mohammed Debbarh, Constantinos Antoniou, and George Yannis. Explaining the road accident risk: weather effects. *Accident Analysis & Prevention*, 60:456–465, 2013.

[56] Chen Chen, Hao Su, Qixing Huang, Lin Zhang, and Leonidas Guibas. Pathlet learning for compressing and planning trajectories. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 392–395, USA. ACM, 2013.

[57] Feng Guo and Youjia Fang. Individual driver risk assessment using naturalistic driving data. *Accident Analysis & Prevention*, 61:3–9, 2013.

[58] Derrick Hambly, Jean Andrey, Brian Mills, and Chris Fletcher. Projected implications of climate change for road safety in greater vancouver, canada. *Climatic Change*, 116(3-4):613–629, 2013.

[59] Tanvi Jindal, Prasanna Giridhar, Lu-An Tang, Jun Li, and Jiawei Han. Spatiotemporal periodical pattern mining in traffic data. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing*, page 11. ACM, 2013.

[60] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[61] Yue Liu. *Weather Impact on Road Accident Severity in Maryland*. PhD thesis, University of Maryland, College Park, 2013.

[62] Raul Montoliu, Jan Blom, and Daniel Gatica-Perez. Discovering places of interest in everyday life from smartphone data. *Multimedia Tools and Applications*, 62(1):179–207, 2013.

[63] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402, 2013.

[64] Sander PA Alewijnse, Kevin Buchin, Maike Buchin, Stef Sijben, and Michel A Westenberg. Model-based segmentation and classification of trajectories. In *Dead Sea, Israel: Proceedings of the 30th European Workshop on Computational Geometry March*, pages 3–5, 2014.

[65] Sander Alewijnse, Kevin Buchin, Maike Buchin, Andrea Kölzsch, Helmut Kruckenberg, and Michel A Westenberg. A framework for trajectory segmentation by stable criteria. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 351–360. ACM, 2014.

[66] Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

[67] David Jaroszweski and Tom McNamara. The influence of rainfall on road accidents in urban areas: a weather radar approach. *Travel behaviour and society*, 1(1):15–21, 2014.

[68] Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] Sheila G Klauer, Feng Guo, Bruce G Simons-Morton, Marie Claude Ouimet, Suzanne E Lee, and Thomas A Dingus. Distracted driving and risk of road crashes among novice and experienced drivers. *New England journal of medicine*, 370(1):54–59, 2014.

[70] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[71] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[72] Athanasios Theofilatos and George Yannis. A review of the effect of traffic and weather characteristics on road safety. *Accident Analysis & Prevention*, 72:244–256, 2014.

[73] Department Of Transportation. Annual average daily traffic (aadt) reports from department of transportation. `https://www.dot.state.oh.us/Divisions/Planning/TechServ/traffic/Pages/Traffic-Count-Reports-and-Maps.aspx`, 2014. Accessed: 2019-12-30.

[74] Yang Zheng, Jianqiang Wang, Xiaofei Li, Chenfei Yu, Kenji Kodaka, and Keqiang Li. Driving risk assessment using cluster analysis based on naturalistic driving data. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 2584–2589. IEEE, 2014.

[75] Mete Celik. Partial spatio-temporal co-occurrence pattern mining. *Knowledge and Information Systems*, 44(1):27–49, 2015.

[76] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[77] Abadi et al. TensorFlow: large-scale machine learning on heterogeneous systems, 2015. URL: `https://www.tensorflow.org/`. Software available from tensorflow.org.

[78] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[79] Kaggle. Driver telematics analysis. `www.kaggle.com/c/axa-driver-telematics-analysis`, 2015. Accessed: 2019-12-30.

[80] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicles from demonstration. In *IEEE International Conference on Robotics and Automation*, pages 2641–2646. IEEE, 2015.

[81] Sachin Kumar and Durga Toshniwal. A data mining framework to analyze road accident data. *Journal of Big Data*, 2(1):26, 2015.

[82] Lei Lin, Qian Wang, and Adel W Sadek. A novel variable selection method based on frequent pattern tree for real-time traffic accident risk prediction. *Transportation Research Part C: Emerging Technologies*, 55:444–459, 2015.

[83] World Health publisher. *Global status report on road safety 2015*. World Health publisher, 2015.

[84] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4580–4584. IEEE, 2015.

[85] Shashi Shekhar, Zhe Jiang, Reem Y Ali, Emre Eftelioglu, Xun Tang, Venkata Gunturi, and Xun Zhou. Spatiotemporal data mining: a computational perspective. *ISPRS International Journal of Geo-Information*, 4(4):2306–2338, 2015.

[86] Stephan Spiegel. Discovery of driving behavior patterns. In *Smart Information Systems*, pages 315–343. Springer, USA, 2015.

[87] Han Su, Kai Zheng, Kai Zeng, Jiamin Huang, Shazia Sadiq, Nicholas Jing Yuan, and Xiaofang Zhou. Making sense of trajectory data: a partition-and-summarization approach. In *2015 IEEE 31st International Conference on Data Engineering*, pages 963–974, USA. IEEE, 2015.

[88] Ling Wang, Qi Shi, and Mohamed Abdel-Aty. Predicting crashes on expressway ramps with real-time traffic and weather data. *Transportation Research Record: Journal of the Transportation Research Board*, 37(2514):32–38, 2015.

[89] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 461–470. ACM, 2015.

[90] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[91] Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. Convolutional recurrent neural networks: learning spatial dependencies for image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 18–26, 2015.

[92] Quanjun Chen, Xuan Song, Harutoshi Yamada, and Ryosuke Shibasaki. Learning deep representation from big and heterogeneous data for traffic accident inference. In *Thirtieth AAAI Conference on Artificial Intelligence*, Palo Alto, CA, USA. AAAI, 2016.

[93] Weishan Dong, Jian Li, Renjie Yao, Changsheng Li, Ting Yuan, and Lanjun Wang. Characterizing driving styles with deep learning. *arXiv preprint arXiv:1607.03611*, 2016.

[94] Erik Erlandson. `http://erikerlandson.github.io/blog/2016/08/03/x-medoids-using-minimum-description-length-to-identify-the-k-in-k-medoids/`, 2016. Accessed: 2019-12-30.

[95] David Hallac, Abhijit Sharang, Rainer Stahlmann, Andreas Lamprecht, Markus Huber, Martin Roehder, Jure Leskovec, et al. Driver identification using automobile sensor data from a single turn. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 953–958. IEEE, 2016.

[96] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[97] Ryo Inoue, Akihisa Miyashita, and Masatoshi Sugita. Mining spatio-temporal patterns of congested traffic in urban areas from traffic sensor data. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 731–736. IEEE, 2016.

[98] Ashesh Jain, Hema S Koppula, Shane Soh, Bharad Raghavan, Avi Singh, and Ashutosh Saxena. Brain4cars: car that knows before you do via sensory-fusion deep learning architecture. *arXiv preprint arXiv:1601.00740*, 2016.

[99] Johan W Joubert, Dirk De Beer, and Nico De Koker. Combining accelerometer data and contextual variables to evaluate the risk of driver behaviour. *Transportation research part F: traffic psychology and behaviour*, 41:80–96, 2016.

[100] Sachin Kumar and Durga Toshniwal. A data mining approach to characterize road accident locations. *Journal of Modern Transportation*, 24(1):62–72, 2016.

[101] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. Unified point-of-interest recommendation with temporal interval assessment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1015–1024, USA. ACM, 2016.

[102] Fred L Mannering, Venky Shankar, and Chandra R Bhat. Unobserved heterogeneity and the statistical analysis of highway accident data. *Analytic methods in accident research*, 11:1–16, 2016.

[103] JD Tamerius, X Zhou, R Mantilla, and T Greenfield-Huitt. Precipitation effects on motor vehicle crashes vary by space, time, and environmental conditions. *Weather, Climate, and Society*, 8(4):399–407, 2016.

[104] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL: `http://arxiv.org/abs/1605.02688`.

[105] Faizan Wajid and Hanan Samet. Crimestand: spatial tracking of criminal activity. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 81. ACM, 2016.

[106] Fei Wu, Hongjian Wang, and Zhenhui Li. Interpreting traffic dynamics using ubiquitous urban data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 69, Burlingame, CA. ACM, 2016.

[107] Wenhao Yu. Spatial co-location pattern mining for location-based services in road networks. *Expert Systems with Applications*, 46:324–335, 2016.

[108] Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(6):1291–1303, 2017.

[109] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

[110] Ye Ding, Yanhua Li, Ke Deng, Haoyu Tan, Mingxuan Yuan, and Lionel M Ni. Detecting and analyzing urban regions with high impact of weather change on transport. *IEEE Transactions on Big Data*, 3(2):126–139, 2017.

[111] Weishan Dong, Ting Yuan, Kai Yang, Changsheng Li, and Shilei Zhang. Autoencoder regularized network for driving style representation learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1603–1609. AAAI Press, 2017.

[112] Sobhan Moosavi, Behrooz Omidvar-Tehrani, R Bruce Craig, Arnab Nandi, and Rajiv Ramnath. Characterizing driving context from driver behavior. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 46:1–46:4, Redondo Beach, CA, USA. ACM, 2017. DOI: 10.1145/3139958.3139992.

[113] Sobhan Moosavi, Behrooz Omidvar-Tehrani, R. Bruce Craig, and Rajiv Ramnath. Annotation of Car Trajectories based on Driving Patterns. *CoRR*, abs/1705.05219:1–10, 2017.

[114] Alameen Najjar, Shun'ichi Kaneko, and Yoshikazu Miyanaga. Combining satellite imagery and open data to map road safety. In *Thirty-First AAAI Conference on Artificial Intelligence*, Palo Alto, CA, USA. AAAI, 2017.

[115] Oluwatobi Olabiyi, Eric Martinson, Vijay Chintalapudi, and Rui Guo. Driver action prediction using deep (bidirectional) recurrent neural network. *arXiv preprint arXiv:1706.02257*, 2017.

[116] Athanasios Theofilatos. Incorporating real-time traffic and weather data to explore road accident likelihood and severity in urban arterials. *Journal of safety research*, 61:9–21, 2017.

[117] Bo Wang, Smruti Panigrahi, Mayur Narsude, and Amit Mohanty. Driver identification using vehicle telematics data. Technical report, SAE Technical Paper, 2017.

[118] Chenglong Wang, Feijun Jiang, and Hongxia Yang. A hybrid framework for text modeling with convolutional rnn. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2061–2069. ACM, 2017.

[119] Lu Wenqi, Luo Dongyu, and Yan Menghua. A model of traffic accident prediction based on convolutional neural network. In *2017 2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, pages 198–202. IEEE, 2017.

[120] Zhuoning Yuan, Xun Zhou, Tianbao Yang, James Tamerius, and Ricardo Mantilla. Predicting traffic accidents through heterogeneous urban data: a case study. In *Proceedings of the 6th International Workshop on Urban Computing (UrbComp 2017), Halifax, NS, Canada*, volume 14, New York, NY, USA. ACM, 2017.

[121] Mingming Zhang, Chao Chen, Tianyu Wo, Tao Xie, Md Zakirul Alam Bhuiyan, and Xuelian Lin. Safedrive: online driving anomaly detection from large-scale vehicle data. *IEEE Transactions on Industrial Informatics*, 13(4):2087–2096, 2017.

[122] Chao Chen, Xiaoliang Fan, Chuanpan Zheng, Lujing Xiao, Ming Cheng, and Cheng Wang. Sdcae: stack denoising convolutional autoencoder model for accident risk prediction via traffic big data. In *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, pages 328–333. IEEE, 2018.

[123] Arijit Chowdhury, Tapas Chakravarty, Avik Ghose, Tanushree Banerjee, and P Balamuralidhar. Investigations on driver unique identification from smartphone's gps data alone. *Journal of Advanced Transportation*, 2018, 2018.

[124] Saad Ezzini, Ismail Berrada, and Mounir Ghogho. Who is behind the wheel? driver identification and fingerprinting. *Journal of Big Data*, 5(1):9, 2018.

[125] Guangyuan Gao, Shengwang Meng, and Mario V Wüthrich. Claims frequency modeling using telematics car driving data. *Scandinavian Actuarial Journal*, 2018(2):143–162, 2018.

[126] Guangyuan Gao, Mario V Wuthrich, and Hanfang Yang. Driving risk evaluation based on telematics data. *Available at SSRN 3288347*, 2018.

[127] David Hallac, Suvrat Bhooshan, Michael Chen, Kacem Abida, Rok Sosic, and Jure Leskovec. Drive2vec: multiscale state-space embedding of vehicular sensor data. *arXiv preprint arXiv:1806.04795*, 2018.

[128] Bing He, Dian Zhang, Siyuan Liu, Hao Liu, Dawei Han, and Lionel M Ni. Profiling driver behavior for personalized insurance pricing and maximal profit. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1387–1396. IEEE, 2018.

[129] Xianbiao Hu, Xiaoyu Zhu, Yu-Luen Ma, Yi-Chang Chiu, and Qing Tang. Advancing usage-based insurance–a contextual driving risk modelling and analysis approach. *IET Intelligent Transport Systems*, 13(3):453–460, 2018.

[130] Chukwutoo C Ihueze and Uchendu O Onwurah. Road traffic accidents prediction modelling: an analysis of anambra state, nigeria. *Accident Analysis & Prevention*, 112:21–29, 2018.

[131] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S Jensen. Distinguishing trajectories from different drivers using incompletely labeled trajectories. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 863–872. ACM, 2018.

[132] Honglei Ren, You Song, Jingwen Wang, Yucheng Hu, and Jinzhi Lei. A deep learning approach to the citywide traffic accident risk prediction. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3346–3351. IEEE, 2018.

[133] Roel Verbelen, Katrien Antonio, and Gerda Claeskens. Unravelling the predictive power of telematics data in car insurance pricing. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(5):1275–1304, 2018.

[134] Dongning Wang, Balaji Gopalakrishnan, Sardar Afra, and Adam R Wisseman. A cnn model for measuring driver risk using synthetic images from accelerometer data. In *Proceedings of the 1st SIAM Workshop on Artificial Intelligence in Insurance*, 2018.

[135] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Pengfei Wang, Yu Zheng, and Charu Aggarwal. You are how you drive: peer and temporal-aware representation learning for driving behavior analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2457–2466. ACM, 2018.

[136] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. Hetero-convlstm: a deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 984–992, New York, NY, USA. ACM, 2018.

[137] Mercedes Ayuso, Montserrat Guillen, and Jens Perch Nielsen. Improving automobile insurance ratemaking using telematics: incorporating mileage and driver behaviour data. *Transportation*, 46(3):735–752, 2019.

[138] Bing Map Traffic API. https://www.bingmapsportal.com/, 2019. Accessed: 2019-12-30.

[139] Jie Chen, ZhongCheng Wu, and Jun Zhang. Driving safety risk prediction using cost-sensitive with nonnegativity-constrained autoencoders based on imbalanced naturalistic driving data. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[140] Graphhopper map-matching library. https://github.com/graphhopper/map-matching, 2019. Accessed: 2019-12-30.

[141] Montserrat Guillen, Jens Perch Nielsen, Mercedes Ayuso, and Ana M Pérez-Marıén. The use of telematics devices to improve automobile insurance rates. *Risk analysis*, 39(3):662–672, 2019.

[142] Xiaoyuan Liang, Guiling Wang, Martin Renqiang Min, Yi Qi, and Zhu Han. A deep spatio-temporal fuzzy neural network for passenger demand prediction. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 100–108. SIAM, 2019.

[143] Sobhan Moosavi, Mohammad Hossein Samavatian, Arnab Nandi, Srinivasan Parthasarathy, and Rajiv Ramnath. Short and long-term pattern discovery over large-scale geo-spatiotemporal data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2905–2913, Anchorage, AK, USA. ACM, 2019. DOI: 10.1145/3292500.3330755.

[144] NAIC. Usage-Based Insurance and Telematics. http://www.naic.org/cipr_topics/topic_usage_based_insurance.htm, 2019. Accessed: 2019-12-30.

[145] Nominatim Tool. https://wiki.openstreetmap.org/wiki/Nominatim, 2019. Accessed: 2019-12-30.

[146] NYC. New York taxi dataset. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml, 2019. Accessed: 2019-12-30.

[147] Open Street Map (OSM). https://www.openstreetmap.org/, 2019. Accessed: 2019-12-30.

[148] DBSCAN parameter tunning. https://github.com/alitouka/spark_dbscan/wiki/Choosing-parameters-of-DBSCAN-algorithm/, 2019. Accessed: 2019-12-30.

[149] Time And Date website. https://www.timeanddate.com/, 2019. Accessed: 2019-12-30.

[150] Traffic Message Channel (TMC) Code. https://wiki.openstreetmap.org/wiki/TMC/Event_Code_List, 2019. Accessed: 2019-12-30.

[151] MapQuest website. https://www.mapquest.com/, 2019. Accessed: 2019-12-30.

[152] Weather Underground website. https://www.wunderground.com/, 2019. Accessed: 2019-12-30.

[153] Wikipedia. Haversine formula. https://en.wikipedia.org/wiki/Haversine_formula, 2019. Accessed: 2019-12-30.

[154] Wikipedia. Usage-based insurance. https://en.wikipedia.org/wiki/Usage-based_insurance, 2019. Accessed: 2019-12-30.

[155] Mohammed J Zaki. http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software#toc19, 2019. Accessed: 2019-12-30.