Computational Method for Understanding Complex Human Routine Behaviors

CMU-HCII-18-100
June 2018

Nikola Banovic
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee

Anind K. Dey (Co-chair), University of Washington
Jennifer Mankoff (Co-chair), University of Washington
Aniket Kittur, Carnegie Mellon University
Eric Horvitz, Microsoft

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

This work was supported partly by the Natural Sciences and Engineering Research Council of Canada (NSERC) (PGSD3-438429-2013), the National Science Foundation (NSF) (CCF-1029549, IIS-1217929), the Yahoo! Fellowship, the Center for Machine Learning and Health (CMLH) at Carnegie Mellon University, and the Software Engineering Institute at Carnegie Mellon University.

ABSTRACT

The ability to collect and store large amounts of human behavior traces data collected from various sensors on people' personal, mobile, and wearable devices, as well as from smart environments, offers a new source of data to study human behavior at scale. However, existing Human-Computer Interaction (HCI) behavior sensemaking methodologies do not lend themselves to studying behaviors from such large multivariate, heterogeneous, and unlabeled datasets. On the other hand, computational modeling has been used to successfully explore and understand complex systems in other fields (*e.g.*, climate change modeling). Inspired by such prior work, we treat behaviors stored in large behavior logs as a complex system that we capture in a computational model of human behavior. In this work, we focus on behaviors in the domain of human routines that people enact as sequences of actions they perform in specific situations, which we call behavior instances. Computational models then allow us to explore different kinds of behaviors by manipulating model variables and simulating and detecting different kinds of behaviors (otherwise known as "asking what-if questions"). In this thesis, we propose a probabilistic computational model of human routine behaviors, that can describe, reason about, and act in response to people's behaviors. We ground our model in a holistic definition of human routines to constrain the patterns it extracts from the data to those that match routine behaviors. We train the model by estimating the likelihood that people will perform certain actions in different situations in a way that matches their demonstrated preference for those actions and situations in behavior logs. We leverage this computational model of routines to create various tools to aid stakeholders, such as domain experts and end users, in exploring, making sense of, and generating new insights about human behavior stored in large behavior logs in a principled way.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank everyone who made this long, yet rewarding, journey an invaluable experience.

First and foremost, I would like to thank my family for their endless support. I would like to thank my wife, Annie Malhotra, who selflessly decided to come with me to Pittsburgh and who has endured and supported me through all my research ups and downs. During our time in Pittsburgh, we welcomed our son, Kabir Mihailo Banovic, who has given me renewed energy to complete my PhD. I would like to thank my mother, Senka Ćuruvija, who has sacrificed much so that I could lead a better life and attain my education.

I would like to thank my advisors, Jennifer Mankoff and Anind Dey, who were always there for me and helped me stay on track even through the most difficult times. Their guidance, mentorship, and unconditional encouragement helped me come closer to becoming the academic I always wanted to be. I would like to thank my thesis committee members Aniket Kittur and Eric Horvitz for invaluable feedback on this dissertation. I would also like to thank Khai Truong, Tovi Grossman, and John Krumm for being my mentors and my champions, and always being available with advice when I needed it most.

I would like to thank all of my collaborators who have contributed to this work. A special thank you goes to Julian Ramos and Christine Bauer for fruitful discussions about routine behaviors, Brian Ziebart, Scott Davidoff and Jin-Hyuk Hong for their valuable input about algorithms and data sets used in this work, Fanny Chevalier and Adam Perer for their insights about the visualizations in this work, and Afsaneh Doryab for leading the data pre-processing efforts used in this work. I would also like to thank students who I have mentored over the years; in particular those whose work contributed to this dissertation: Tofi Buzali, Jae-Won Kim, Seo Hyun “Jenna” Choo, Christie Chang, Anqi “Angie” Wang, Yanfeng “Tony” Jin, Zhongmin “Angela” Xie, and Ticha Sethapakdi. This work would not have been possible without them.

I would like to thank everyone at the Human-Computer Interaction Institute, and in particular all Ubicomp Lab and Make4All (formerly Assist Lab) past and present members

and visitors. It was a great pleasure working alongside you. I would like to specially thank Queenie Kravitz, who was always there with an encouragement or an answer when I had a question regarding the PhD program. I would also like to thank my cohort, Dan Tasse, Brandon Taylor, Tatiana Vlahovic, Jenny Olsen, Anthony Chen, Chris MacLellan, and Dave Gerritsen, and my CHI travel partners Michael Nebeling and Adrian de Freitas. You inspired me to always want to do better and your friendship brought me happiness during my time at Carnegie Mellon University.

TABLE OF CONTENTS

ABSTRACT 3

ACKNOWLEDGEMENTS 4

TABLE OF CONTENTS 6

1 INTRODUCTION 10

2 BACKGROUND IN UNDERSTANDING BEHAVIORS 14

2.1 Exploratory Data Analysis 15

2.2 Data Mining 16

2.3 Information Visualization and Visual Analytics 19

2.4 Modeling Interaction with Information Technology 20

2.5 Summary 21

3 COMPUTATIONAL MODELING METHODOLOGY 23

4 OPERATIONIZABLE DEFINITION OF ROUTINES 26

4.1 Defining Routine Behaviors 27

4.2 Unified Routine Definition 29

4.3 Unified Routine Definition and Existing Models of Routines 30

5 COMPUTATIONAL MODEL OF ROUTINE BEHAVIOR 32

5.1 Model of Human Routine Behavior 33

5.2 Data Modeling and Feature Engineering 34

5.3 Learning Routine Patterns from Demonstrated Behavior 35

5.4 Validating the Model of Human Routine Behaviors 38

5.4.1 Training Models of Routine Behavior 39

5.4.1.1 Family Daily Routines Data Set 40

5.4.1.2 Aggressive Driving Behavior Data Set 42

5.4.2 Quantifying Routineness of Human Behavior 44

5.4.3 Visual Exploration of the Model 45

5.4.3.1 Study Software 45

5.4.3.2 Participants 47
5.4.3.3 Method 48
5.4.3.4 Ground Truth 48
5.4.3.5 Results 49
5.5 Summary 50
6 CAPTURING ECONOMICS OF ROUTINE BEHAVIOR 53
6.1 Extending Situation and Action Feature Engineering 54
6.1.1 Situation Features 54
6.1.2 Action Features 56
6.2 Representing Time 57
6.3 Economics of Human Routine Behavior 59
6.4 Summary 61
7 DETECTING AND GENERATING ROUTINES 62
7.1 Detecting Classes of People Based on Their Behavior 63
7.2 Detecting Behavior Instances 64
7.3 Generating Behavior Instances 67
7.4 Use Case: Predicting Cancer Patient Rehospitalization 67
7.4.1 Cancer Patient Behavior Dataset and Model 68
7.4.1.1 Participants 69
7.4.1.2 Data Collection 69
7.4.1.3 Data Pre-processing 70
7.4.1.4 Model Training 73
7.4.2 Predicting Rehospitalization 73
7.4.3 Method 74
7.4.4 Results 75
7.4.5 Discussion 78
7.5 Use Case: Detecting Aggressive Driving Behaviors 79
7.5.1 Naturalistic Driving Behavior Dataset and Model 81
7.5.2 Classifying Driving Behavior Instances 82

7.5.3 Generating Driving Behavior Instances 83
7.5.4 Preliminary Detection Validation 84
7.5.5 Domain Expert Evaluation of Driving Detection and Generation 85
7.5.5.1 Results 86
7.5.5.2 Discussion 87
7.6 Summary 88
8 ROUTINE MODELS AS SENSEMAKING TOOLS 89
8.1 Identifying Salient Patterns of Routine Behavior 90
8.2 Use Case: Differentiating Aggressive Driving Behaviors 90
8.2.1 Driving Behaviors Animation System 91
8.2.2 Driving Animation Evaluation 91
8.2.3 Acting in Response to People's Behaviors 92
8.2.3.1 Method 93
8.2.3.2 Measures 94
8.2.3.3 Results 94
8.2.3.4 Discussion 96
8.2.4 Summary 98
8.3 Generating and Validating Hypotheses about Routines 98
8.3.1 Design of Behavior Dashboard 101
8.3.1.1 Data Organization 103
8.3.1.2 Visual Representation of Behaviors 103
8.3.1.3 Interactions with Behaviors 107
8.3.2 Use Case: Understanding Behaviors that Leads to Rehospitalization 111
8.3.2.1 Behavior Analysis and Results 111
8.3.2.2 Discussion 117
8.4 Summary 118
9 CONCLUSION AND FUTURE WORK 120
9.1 Mixed-initiative Computational Modeling 123
9.2 Understanding Capabilities and Limitations of AI 125

9.3 Human-data Supported Interfaces 126
9.4 Summary 128
10 APPENDIX 129
10.1 MaxCausalEnt IRL Algorithm Implementation 129
10.2 Study Materials 139
10.2.1 Visual Model Validation Study Questionnaire 139
10.2.2 Driving Instructor Detection and Simulation Validation Questionnaire 141
10.2.3 Aggressive Driver Assessment Study Modified DBQ Questionnaire 143
10.3 Behavior Dashboard Design Materials 146
BIBLIOGRAPHY 148

1 INTRODUCTION

The amount of human behavior data collected and stored in the world every minute is staggering. Information Technology that enables us to collect behavior data touches on almost every aspect of people's lives. For example, fitness trackers count each step we take, web search engines process and store each Internet search we make, social media sites record each personal connection we establish and each message we post, map and navigation applications record each place we visit. Such unprecedented ability to collect data is enabled by both ever-increasing use of software applications that log every user interaction and the proliferation of personal, mobile, and wearable devices and smart environments outfitted with precision sensors that can track and collect data about people and their environments. Such data is already used in technological advances that improve the quality of people's lives, including advances in areas of traffic safety, healthcare diagnostics and targeted treatment, and physical assistance for elder care, to mention a few (Stone, *et al.*, 2016).

This trove of data potentially contains valuable information about people's behaviors. Study of people's behaviors in relation to Information Technology is central to the field of Human-Computer Interaction (HCI) and the knowledge we could generate from this new source of data has the potential to establish theoretical foundation for work in HCI. For example, understanding how people interact with Information Technology, the tasks that they perform, and the goals they want to accomplish helps us improve existing interactions and inform the design of future Information Technology. Understanding how people's data is used in complex Information Technology systems could also allow us to democratize technology (O'Neil, 2016; Pasquale, 2015), help address further socio-economic divisions that technology could introduce (The Economist, 2015), and safeguard us from rogue technology that does not have people's best interests in mind (Gray, Kou, Battles, Hoggatt, & Toombs, 2018). Thus, it is particularly important that HCI researchers have the right tools to explore and understand this massive amount of data about people that we generate, collect, and store.

However, the traditional HCI methodologies to study human behavior do not translate well to exploring and understanding behaviors from large behavior logs that contain multivariate, heterogeneous, unlabeled data. Qualitative methods (*e.g.*, ethnography, observations, and interviews) often have to integrate information from large amounts of data collected from many different sources (*e.g.*, video and audio recordings), but are not applicable to exploring behavior logs because they deal with the kind of data that is very different from quantitative data contained in behavior logs. Null-hypothesis testing has become a popular de-facto standard for quantitative analysis in HCI. Although such analysis applies to data collected in both lab and field studies, it is meant for testing pre-conceived hypotheses in classical experimental settings and not applicable to data exploration (Good, 1983).

In response to growing availability of behavior logs, HCI research started incorporating existing Exploratory Data Analysis (EDA) (Good, 1983; Tukey, 1977), Information Visualization (Fekete *et al.*, 2008), and Data Mining (Fayyad *et al.*, 1996) and Visual Analytics (Keim *et al*., 2008) tools and techniques to explore behaviors from behavior traces data. Each of these methods addresses some aspects of behavior data exploration. For example, EDA could identify people's general behavioral dispositions by aggregating data on isolated variables, Information Visualization could visualize data to uncover nuanced patterns made up of sequences of multivariate, heterogeneous behavior data, and Data Mining and Visual Analytics tools could automatically extract and visually present such salient patters from the data. Yet, there is still no holistic method for studying and understanding behaviors from data traces stored in large behavior logs.

Here, we address the challenges of understanding behavior data from large behavior logs by developing a computational modeling methodology to study human behaviors from data stored in large behavior logs. Computational modeling expresses processes within complex systems mathematically to enable exploration of the system by simulation and prediction. Our methodology bridges the gap between data models (that can explain behaviors, but have been constrained to very simple behavior models) and algorithmic models (that disregard underlying behaviors that generated the data as unknown, but can capture multivariate relationships in the data very well) (Breiman, 2001). The goal of our

computational modeling approach is to provide a principled way for exploring behaviors in both an aggregate and sequential manner.

We developed a probabilistic computational model of high-level behaviors, such as routines, that can be used to describe, reason about, and act in response to people's behaviors. This computational routine model describes behaviors and provides causal explanation for relationships between actions and situations in which people perform those actions. We automatically estimate such relationships from the data (*i.e.*, train the model) using an algorithm based on the principle of Maximum Causal Entropy (Brian Ziebart, 2010), which is a data mining algorithm grounded in statistical principles to ensure that the model finds explanations for behaviors that best fit the data. We use this algorithm because it trains models that can predict people's behaviors (Ziebart, Maas, Dey, & Bagnell, 2008; Ziebart, Ratliff, & Gallagher, 2009). We leverage similar properties in the model to act in response to people's behaviors (*e.g.*, automate tasks, prescribe behaviors).

Our computational model supports making sense of behavior data (*i.e.*, searching for salient representations in the data (Russell, Stefik, Pirolli, & Card, 1993)), by identifying and encoding behaviors that are characteristic of a routine. We automate multiple aspects of the sensemaking process (Pirolli & Card, 2005) by automatically searching for relationships in the data (information foraging) to model (schematize) patterns of behaviors that form routines. The ability to generate behaviors from a model grounded in our unified definition of routine behavior increases intelligibility of the model and allows stakeholders to search for evidence that the model matches empirical behavior data. We present custom visualization tools to support visual exploration of automatically detected and generated behaviors. Our visual analytics tools allow stakeholders to form hypothesis about behaviors (*e.g.*, hypothesize that aggressive drivers are more prone to speeding) and test those hypotheses by identifying evidence in the model to support the hypotheses.

In this work, we argue the following thesis statement:

> *A rich computational model of human routine behavior, that captures relationships between situations and actions, can be used to describe, reason about, and act in response to behaviors, stored as event traces in large behavior logs. We hypothesize that such a model can aid stakeholders in sensemaking about behavior of individuals and populations. We also hypothesize that such a model can automatically detect and extract salient patterns of behavior that characterize a routine, such as poor routines, and act in response to those behaviors to prescribe changes, such as simulating a better routine.*

We begin with a review of existing methods for studying, exploring, and understanding human behavior from large behavior logs, and the challenges that exist when applying the current methodologies to this problem (Chapter 2). We then introduce our computational modeling methodology, grounded in the sensemaking process (Pirolli & Card, 2005), in Chapter 3 and proceed to describe each step in detail in Chapters 4 through 8. In Chapter 4, we present our unified definition of routine behaviors, a kind of behavior that is the main focus of this work. We then show how to estimate a computational model of human routines from behavior logs in Chapter 5, and how our modeling approach builds on existing algorithms to capture the economics of routine behavior in Chapter 6. We then show how to leverage computational models of routines for behavior exploration by first presenting automated computational techniques (Chapter 7) to identify salient patterns of behaviors in the data and a set of visualization tools that allow stakeholders to leverage our computational model as a behavior sensemaking tool (Chapter 8). We conclude with a future direction for computational modeling of human behavior in Chapter 9.

2 BACKGROUND IN UNDERSTANDING BEHAVIORS

Understanding human behaviors has become an increasingly important topic in Human-Computer Interaction (HCI) as Information Technology is introduced into various aspects of people's lives. In recent years, HCI research has moved from a focused study of people's interaction with User Interfaces to a study of people's lives surrounded by Information Technology anytime and anywhere. To collect, analyze, and study such behaviors, the HCI community has developed various methodologies, with foundations in Cognitive Science, Psychology, Social Sciences, and Data Sciences. Such methods are often categorized based on two dimensions: 1) qualitative *vs.* quantitative, and 2) lab *vs.* field. Although being one of the core elements of HCI, qualitative methods (see (Beyer & Holtzblatt, 1998; Gaver, Dunne, & Pacenti, 1999; Millen & R., 2000) for examples) do not have an application in the study and analysis of behavior log data.

Thus, our focus is on quantitative methods in HCI. Quantitative lab studies offer a controlled setting to collect data about a behavior. Researchers commonly use such studies to test a hypothesis using an experimental design. Less often, they are used to explore and understand a behavior. However, such studies are often contrived and lack external validity (*i.e.*, they do not generalize to behaviors outside of the experimental setting). Also, due to short time periods of such studies, they are not appropriate for studying routine behaviors that span long periods of time and different contexts. Traditional quantitative field studies collect behavior data from natural settings and over time. However, they are resource intensive and do not scale up. Furthermore, they often produce knowledge about behaviors that is difficult to operationalize.

In this work, we further focus on quantitative analysis of data from a special kind of field studies, called log studies, that collect traces of human behaviors and store them in large behavior logs. Such behavior logs complement data from traditional lab and field observational studies by offering behavior data collected in natural settings, uninfluenced by observers, over long period of time, and at scale (Dumais, Jeffries, Russell, Tang, & Teevan, 2014). Behavior log data can be collected using various server- (Google, 2017) and client-side (Capra, 2011) software logging tools, and from sensors in the environment

(Koehler, Banovic, Oakley, Mankoff, & Dey, 2014) and on people's personal or wearable devices (Ferreira, Kostakos, & Dey, 2015). Examples of behavior logs include web use logs (Adar, Teevan, & Dumais, 2008), social network logs (Starbird & Palen, 2010), outdoor mobility logs (Davidoff *et al.*, 2011), mobile device usage logs (Banovic, Brant, Mankoff, & Dey, 2014), and even vehicle operation logs (Hong, Margines, & Dey, 2014). Stakeholders can then use the traces stored in behavior logs to understand complex behaviors. We review the most common analysis methodologies used to study and explore such behavior logs.

2.1 Exploratory Data Analysis

Explanatory methods provide support for manually searching for salient patterns in the data. This is often done using Exploratory Data Analysis (EDA) (Tukey, 1977), which offers different descriptive statistics and data visualization techniques to manually explore and understand high-level patterns in behavior logs. Note that EDA methods focus on observational data and differ from methods for analyzing data derived in classical experimental setups (Good, 1983). Finding supporting evidence that describe people's behavior from log data typically involves identifying meaningful variables to partition the data on and then comparing behaviors across the different partitions (Dumais *et al.*, 2014). Finding such partitions is at the discretion of the stakeholder, but often includes variables that describe temporal properties in the data or characteristics of people. Temporal properties can reveal periodicity and recurrence of events and user characteristics allow stakeholders to compare behaviors across different populations. For example, Hong *et al.* (2014) partitioned data on drivers' propensity towards aggressive behaviors to identify and describe the differences in high-level driving behaviors (speeding, accelerating, braking) between the two groups.

However, EDA focuses on isolated events, or temporal evolution of a particular state (*e.g.*, sleep *vs.* awake) or variable (*e.g.*, the number of steps walked per day). Such partitions describe people's general dispositions using the principle of aggregation, which often "does not explain behavioral variability across situations, nor does it permit prediction of

a specific behavior in a given situation" (Ajzen, 1991). Routines are characterized by both specific actions people perform in different situations (Hodgson, 1997) and variability of those actions across situations (Feldman & Pentland, 2003). This makes it difficult to manually find nuanced relationships in the data. Also, due to the complexity and size of behavior logs, EDA methods do not guarantee that the stakeholder will be able to find patterns of behaviors (*e.g.*, those that form routines) and not some other patterns in data that are unrelated to behaviors. Furthermore, even large behavior logs contain too few behavior instance examples to account for all possible transitions between different situations and actions. Without a statistical model that estimates the probabilities of transitions between situations and actions in a principled way, even if not all possible transitions are present in the data, each behavior instance becomes an isolated example that is difficult to generalize from.

2.2 Data Mining

Data Mining automates the process of identifying potentially useful patterns in data (Fayyad *et al.*, 1996) in a way that could automatically capture salient patterns that describe behaviors. Such methods alleviate manual exploration of data in the information foraging loop that is characteristic of explanatory approaches we discussed above (Pirolli & Card, 2005). Data Mining employs existing machine learning algorithms to detect (*e.g.*, (Bulling, Blanke, & Schiele, 2014)), classify (*e.g.*, (Hong *et al.*, 2014)), predict (*e.g.*, (Krumm & Horvitz, 2006)), and find structure of (*e.g.*, (Baratchi, Meratnia, Havinga, Skidmore, & Toxopeus, 2014; Rashidi & Cook, 2010)) patterns in the data. Such methods can automatically act in response to patterns in the data. Stakeholders can then explore the results of data mining algorithms using various visual analytics tools (Keim *et al.*, 2008).

However, concerns remain about performing data mining using black-box algorithmic-based models that often leverage patterns in the data that may not be representative of the underlying process that generated the data (Shmueli, 2010). Although this may be acceptable in certain prediction tasks (Kleinberg, Ludwig, Mullainathan, & Obermeyer, 2015), there is no easy way to inspect such models to ensure that they will not make wrong

decisions because of some fundamental misconception. For example, Google Flu Trends (Cook, Conrad, Fowlkes, & Mohebbi, 2011) used a model that was meant to predict the number of patient visits to doctors for influenza-like illness (ILI) based on patients' Google search terms. The model started significantly overestimating ILI outbreaks years after its launch (*e.g.*, in response to a media-induced epidemic panic (Lazer, Kennedy, King, & Vespignani, 2014)) because it wrongly assumed that correlation between volume of people's search terms and ILI also means causation. In the case of Google Flu Trends, wrong predictions could lead to minor annoyance or waste of resources. However, in other cases, such as when an algorithm that data mines a large data set of portrait photographs to detect criminals based on their facial expression makes a misclassification (Wu & Zhang, 2016), mistakes could have far more reaching negative consequences (*e.g.*, incarceration of an innocent person).

Also, such general-purpose data mining methods may not be well suited to extract behavior patterns. For example, such existing machine learning algorithms purposefully disregard variations in human behavior to focus on classifying and predicting only the most frequent human activity. Some variations may happen infrequently in data and are difficult to detect using those existing algorithms. Some specific infrequent variations may be detectable (*e.g.*, detecting when parents are going to be late to pickup their children (Davidoff, Ziebart, Zimmerman, & Dey, 2011)). However, this requires a case-by-case approach to address each variation, which can be difficult to apply if all possible variations are not known *a priori*.

Most such models require labeled examples about which behavior instances are characteristic of a routine and which ones are not. However, the lack of individually labeled behavior instances in large behavior logs makes it challenging to use those existing supervised machine learning algorithms to classify behavior instances into routines. For example, to train a supervised machine learning algorithm to classify behavior instances that lead parents to forget to pick up their children, Davidoff *et al.* (2010) had to manually label each behavior instance in the behavior log they collected, and confirm this information with the participants in their next study (Davidoff *et al.*, 2011). This places

significant burden on stakeholders to properly label enough data to be able to train their data mining algorithms.

Unsupervised machine learning methods cluster behaviors without prior knowledge of labels. For example, algorithms based on Topic Models (Farrahi & Gatica-Perez, 2012) allow stakeholders to generate clusters of behavior instances. However, the main limitation of general-purpose unsupervised methods is that they offer no guarantees that the resulting clusters group instances based on the routine they belong to (*i.e.*, the clusters may not represent routines). Unsupervised anomaly detection algorithms (*e.g.*, (Mcfowland, Speakman, & Neill, 2013)) could be used to find differences between behavior instances. However, they detect if a behavior instance is a deviation from a routine, but not whether it is part of the routine.

This highlights the major challenge with most traditional machine learning models: there is no easy way to inspect the model to ensure that it captures meaningful patterns of routine behavior. Like any other black-box predictive model, there is no easy way to inspect the model and ensure that it captures meaningful patterns of behavior. Stakeholders can inspect Generative Models (Salakhutdinov, 2009) by generating behaviors from the model and comparing them with empirical behavior instances in the data. However, this assumes that the stakeholder already understands behavior instances characteristic of a routine, which is what the model is supposed to automatically extract for the stakeholder in the first place.

Unsupervised methods specifically designed to model routines from behavior logs (Eagle & Pentland, 2009; Farrahi & Gatica-Perez, 2012; Li, Kambhampati, & Yoon, 2009; Magnusson, 2000) are meant to capture patterns of routine behavior. Each offers a unique approach to modeling routines. For example, Eigenbehaviors (Eagle & Pentland, 2009) map events in the data on a discrete timeline vector and use eigen decomposition to find principled components of people's behaviors (*i.e.*, most salient combinations of behavioral features). Eagle & Pentland (2009) provide custom visualization of those principled components to inspect the model. Past research has also shown that Eigenbehaviors can be used to act in response to people's behavior and predict their mobility (Sadilek & Krumm, 2012). However, such models are based on optimization methods that minimize simple

error functions between the patterns they extract and the data. Thus, it remains unclear which aspects of routines those existing data mining approaches are able to capture.

Still, the advantage of methods that specialize in extracting routines compared to general-purpose Machine Learning approaches is that they can be optimized to match some aspect of routines as identified in theory about routine behaviors. For example, T-patterns (Magnusson, 2000) search event-based behavior log data for multivariate events that reoccur at a specific time interval, which they combine to create new composite events. The algorithm recursively groups events to define structure of routine behaviors that is only described by the temporal aspects of the data. Context-Free Grammar-based models (Li *et al.*, 2009) encode the hierarchical structure of routine activities. Such models can be trained using the Expectation-Maximization algorithm (Bishop, 2006) that maximizes the likelihood of making the observations from the data using the learned hierarchical representation of routine activities. Stakeholders can explore each model using various custom visualizations to check if that the models match meaningful patterns of behavior.

2.3 Information Visualization and Visual Analytics

Visualizing data from behavior logs is another common way for stakeholders to identify salient patterns in the data. Such data is often visualized on a timeline as a sequence of events. The simplicity of this approach makes it applicable to a variety of domains, such as schedule planning to show uncertainty of duration of different events (Aigner, Miksch, Thurnher, & Biffl, 2005), visualizing family schedules (Davidoff *et al.*, 2010), and representing events related to patient treatments (Plaisant, Milash, Rose, Widoff, & Shneiderman, 1996). More advanced timelines enable the user to specify properties of the timeline (*e.g.*, periodicity) for easier viewing. For example, Spiral Graph (Weber, Alexa, & Müller, 2001) aligns sequential events on a spiral timeline using a user-defined period.

However, even advanced interactive visualizations have difficulty in visualizing behavior patterns that depend on multiple heterogeneous variables, especially as the number of variables grows. For example, parallel coordinates visualization can clearly show relationships between different multivariate aspects of behavior data (Clear *et al.*, 2009),

until the number of variables becomes too large. To address such challenges, stakeholders can manually highlight (Buono, Aris, Plaisant, Khella, & Shneiderman, 2005) and aggregate common sequences of events based on features in the data (Jin & Szekely, 2010; Monroe, Lan, Lee, Plaisant, & Shneiderman, 2013) until meaningful patterns emerge. The stakeholder is then able to judge the quality and saliency of the patterns by visual inspection. However, this can be painstakingly slow, making manual exploration challenging. The problem of underrepresented transitions present in general Exploratory Analysis translates to visualization as well. For example, CareFlow (Perer & Gotz, 2013), which visualizes paths of treatments for patients with cardio-vascular diseases, shows that sequences of at-risk patients often reduce to singular examples. This makes it difficult to estimate if such sequences should be treated as exemplars of behavior or isolated incidents and noise.

Visual Analytics tools combine data mining with information visualization. As such they often extract salient patterns from the data before visually resenting them to the user. For example, stakeholders can visually explore T-patterns using Arc Diagrams (Wattenberg, 2002) or the hierarchical structure of routine activities captured in a Context-Free Grammar-based model (Li *et al.*, 2009) using DAViewer (Zhao, Chevalier, Collins, & Balakrishnan, 2012). However, such tools often suffer from the same underlying problems as the data mining methods they use.

2.4 Modeling Interaction with Information Technology

Behavior models provide a theoretical foundation for work in HCI. Traditionally, in HCI, such models focus on capturing cognitive and motor components of interactions with User Interfaces. For example, GOMS (Card, Moran, & Newell, 1983) is a human information processing model created specifically to describe and estimate cognitive performance of people when they interact with User Interfaces (*e.g.*, a Graphical User Interface). GOMS empirically estimates times of different operators and methods that it encodes (*e.g.*, time to perceive a target in a pointing task, time to invoke the motor system, time to move the hand to press on the target), and then uses those time of simple atomic operations to

estimate the time it takes to accomplish a complex task via a User Interface. However, empirical estimates of times for each component of the model are time consuming and past research has hypothesized models that can predict behavior. For example, the Fitts' Law (MacKenzie, 1992) predicts the time it takes to press on a target can be estimated with as little as two variables: distance to and size of the target. Such models can then be used to explain people's low-level interactions with actual User Interfaces (*e.g.*, typing on a mobile keyboard by pointing at keys (Banovic *et al.*, 2017)).

Such models are driven by knowledge about behavior rather than driven by an optimization function that tries to reduce the error between model estimated patterns and data, which are characteristic of data mining. This means that they are developed based on existing hypotheses about behaviors and empirically validated, rather than simply trained on behavior data without regard for the underlying behavior processes (Breiman, 2001). However, they are mostly restricted to simple, low-level behaviors that can be described using intuitive analytical solutions. Although high-level behavior models exist, when applied to complex, heterogeneous, multivariate behavior data (*e.g.*, to explain how people adopt information technology (Venkatesh *et al.*, 2003)), such explanatory models are often unable to explain most of the variance in the observed data. Such models have less predictive power than data-mining based models (Kleinberg *et al.*, 2015), even in some cases when they closely approximate the true process that generated the data (Shmueli, 2010). Thus, they may not be able to capture the full complexity of high-level behaviors, such as routines, and be used to reason about and act in response to behaviors they capture.

2.5 Summary

In our review of existing methodologies, we identified two kinds of quantitative data analysis approaches to study and explore data in large behavior logs. The first category of approaches (EDA, Information Visualization, and Behavior Modeling) are driven by the processes that generated the data, but are resource intensive and require manual work to identify salient patterns of behavior in the data. The difficulty in using such methods in amplified by the fact that today's behavior logs contain massive amount of multivariate,

heterogeneous, unlabeled data that is not easy to conceptualize. Even once stakeholders identify salient patterns of behavior and use them to understand behaviors (*i.e.*, create a conceptual model of behaviors), it remains unclear how to make that knowledge operational so that we can create technology that can automatically reason about and act in response to people' behaviors.

The second category (Data-Mining, Visual Analytics) are algorithmic approaches that automatically extract salient patters from the data by optimizing an error function without a regard for the underlying processes that generated the data. Such powerful algorithms can quickly summarize large amount of data stored in behavior logs and can be used to automatically act in response to behaviors (*e.g.*, classify and predict future behaviors). However, such methods often extract patterns that do not correspond to actual behaviors that are of interest to the stakeholders. They could also lead to gross misclassifications and wrong predictions when the correlations in the data they leverage to extract patterns do not have any causal relationship with people's actual behaviors.

The main challenge in reconciling these two approaches is the lack of a holistic method to modeling human routine behaviors that can automatically extract patterns of behavior from large behavior logs in a way that models the processes that generated the data. For example, for Data-Mining approaches, this would mean identifying optimization functions that match important aspects of the processes that generated the data. This could improve our confidence that the extracted patterns match actual behaviors of people. Yet, the existing models of behavior based on Data Mining algorithms still largely disregard theoretical foundation about behaviors necessary for understating human behavior.

3 COMPUTATIONAL MODELING METHODOLOGY

Computational Modeling mathematically encodes processes in a complex system and enables exploration of the system through prediction and simulation (Melnik, 2015). We treat people's behaviors situated in their environments as an example of such a system. Although some aspects of computational models can be built using data mining techniques, computational modeling is different from simple data mining approaches because it insists on representing actual processes that generated the data. It also encodes patterns of behavior that allow prediction and simulation of behaviors, which is not necessarily the case with existing data mining techniques.

However, it is not immediately obvious how to compute a model from large amounts of heterogeneous and unlabeled data stored in behavior logs. For example, behavior logs contain information about what people did, but not why they did it (Dumais *et al.*, 2014). Even when data contains information about what people did, individual activities may not be clearly segmented (Hurst, Mankoff, & Hudson, 2008). Stakeholders explore and understand behavior logs through process of sensemaking, which Russell *et al.* (1993) define as "the process of searching for a representation and encoding data in that representation to answer task-specific questions."

We look at sensemaking about behaviors through the lens of Pirolli and Card's (2005) notional model of sensemaking loop for intelligence analysis. In the information foraging loop (Pirolli & Card, 2005), stakeholders first search for information and relationships in the data to find and organize evidence that supports their reasoning about behaviors patterns in the data. Stakeholders then use the evidence to schematize their current understanding of behaviors and conceptualize it in a model of behavior. Stakeholders then use this conceptual model in the sensemaking loop (Pirolli & Card, 2005) to create and support (or disconfirm) their hypotheses about behaviors and present their findings.

The behavior data information foraging loop reduces to identifying salient patterns in behavior data (representations) that describe routines. A common way to kick off this stage is when a stakeholder begins raw data exploration by searching for relevant features of situations and actions that describe behaviors of interest. For example, in case of driving

routines, situational features could include road configuration and if the driver is in a rush hour or not, while action features could include how the driver operates the steering wheel and gas and brake pedals. This allows the stakeholders to extract knowledge about different possible behavior instances (*e.g.*, how a driver operates a vehicle through a road segment, such as an intersection). The goal of the stakeholders is then to extract behavior instances that form routine variations, while at the same time rejecting deviations. It is often important to ensure that such variations are not also part of another competing routine (*e.g.*, that a behavior is characteristic of aggressive, but not non-aggressive driving routine). This process involves continued iterative search for relationship between situations and actions that form such behavior instances.

The sensemaking loop involves transitioning from evidence to conceptual model of behavior. Once there is enough evidence to support a conceptual model of a routine (*e.g.*, a conceptual model of driving routines) the stakeholder can begin to generate hypothesis about behaviors in that model. For example, the stakeholder might hypothesize that drivers drive more aggressively during rush hour than during other times of the day, or that aggressive drivers are driving faster than non-aggressive drivers. Given different competing routine models (*e.g.*, aggressive and non-aggressive driving routine) the stakeholder might hypothesize about differences between behaviors in those two routines. For example, the stakeholder might hypothesis that aggressive drivers are more likely to be speeding than non-aggressive drivers. The stakeholder would then search for support information in the conceptual model that would prove or disprove this hypothesis. Confirming or disconfirming hypotheses allows the stakeholder to form theories about behavior. In turn, any of those theories could be challenged by considering new evidence. Such new evidence informs the conceptual model, which completes the loop.

We thus propose specific set of iterative steps, grounded in the sensemaking process (Pirolli & Card, 2005), to create a computational model from large behavior data:

1. Identify a research question
2. Clearly define the type of behaviors based on existing knowledge and theory

3. Deploy a field study to collect peoples' behavior data logs from their various devices and environments that might help answer the question
4. Build a computational model that is grounded in the definition of behavior
5. Explore the model to perceive trends and create a conceptual model of behavior
6. Generate hypotheses about the behavior that help answer the research question
7. Search the model for examples that prove or disprove the hypothesis
8. Generate new insights, tune hypotheses, or adjust the research question
9. Present findings

In the remaining of this document, we describe this process in detail and present various use case to illustrate the steps. Although identifying a research question (Step 1) and collecting data (Step 3) are important steps in our methodology, they are also domain specific and will vary from use case to use case. As such we defer discussing those steps until later chapters when we present our use cases. Instead, we begin describing our process with Step 2 in our methodology in Chapter 4, by presenting our unified definition of routine behaviors, which is the main focus of this work. We then discuss Step 4 in detail, and show how to calculate a computational model of human routines in Chapter 5, and the considerations we must make to ensure that the model captures economics of routine behavior in Chapter 6. We begin our discussion about behavior exploration in Step 5 by first presenting automated computational techniques (Chapter 7) that can leverage the model to identify salient patterns of behaviors in the data. We then present tools that help stakeholders leverage our computational model as a behavior sensemaking tool in steps 5 through 9.

4 OPERATIONIZABLE DEFINITION OF ROUTINES

A definition of routines conceptualizes high-level knowledge about behaviors (*i.e.*, schematizes existing knowledge about routines). Such conceptual models encode high-level real-world processes that generate behaviors data across different routines (*e.g.*, daily routine, exercising routine, driving routine) of both individuals and populations. This allows stakeholders to compare salient patterns they identified in the data with this conceptual model of routines and ensure they found patterns that are representative of a routine and not some other patterns in the data. This is particularly important for data mining methods that automate the foraging loop to create computational models of routines from patterns of behaviors automatically extracted from the data. Such a conceptual model provides constraints on computational models of routines to favor salient patterns in the data that match properties and structure of real world behaviors. A computational model of routine behaviors grounded in a definition of routines combines the power of explanatory models to describe behaviors with the power of predictive models to automatically find salient behavior patterns in the data, and even act in response to those behaviors afterwards.

However, current definitions of routines focus on different aspect of behaviors that make up routines, which makes it difficult to reconcile them into a single unified definition that can be operationalized in a holistic computational model. We focus primarily on operationalizing routines of individuals, including routines of homogenous populations of people. Although such routines share similarities with organizational routines (see (Becker, 2004) for a review), we only broadly explore how individuals perform in the context of organizational routines (*i.e.*, the performative aspect of organizational routines (Feldman & Pentland, 2003)), and do not focus on operationalizing organizational routines *per se*. Also, the goal here is not to define the processes that people use to generate mental plans that manifest themselves as routines. Instead, we focus our analysis on physical activity in a given context (Kuutti, 1995). Our goal is to provide a definition of routine behaviors as people enact them in action.

Focusing on how people enact their mental plans allows us to broadly include habitual behaviors into an operationalizable definition of routines. Habits represent people's

tendency to act in largely subconscious ways (Hodgson, 2009) that are different from other planned behaviors that require deliberation (Ajzen, 1991). Although there exist qualitative differences between subconscious and deliberate behaviors (Hodgson, 1997), which may impact the outcome of those behaviors (Kahneman, 2011), when enacted both deliberate routines and subconscious habits form similar patterns. Note that, although we consider routines that may include bad habits and other behaviors that could negatively impact people, we exclude a discussion on pathological behaviors, such as addiction, which require special consideration.

4.1 Defining Routine Behaviors

At the highest level, we can define routines as rules: actions Y that people perform when in situation X that is the cause of action Y (Hodgson, 1997). Such a definition is too high level and is missing many aspects that define routines, and thus does not give enough information to be operationalized into a computational model of routines. For example, Hodgson (1997) does not explicitly describe what makes up situations that influence people's actions. This is likely because such features will vary across different types of routines (*e.g.*, features that describe situations and actions in a daily routine are different from those in a driving routine.

However, many existing routine definitions include essential features of routine behaviors. For example, some existing definitions consider routine only those actions that repeat on a specific time interval (Brdiczka, Su, & Begole, 2010; Casarrubea *et al.*, 2015). Other such commonly considered routine defining features include spatial context (Feldman, 2000), and social influences (Hodgson, 2009). However, it is more likely that different routines are defined by combinations of multiple different features of situations and actions.

One such influence that few existing definitions of routines explicitly consider are people's goals, which provide motivation to act. For example, Pentland & Rueter (1994) loosely define routine behaviors of individuals that are part of an organization as "means to an end;" and Hamermesh (2003) proposes that people perform routine actions to maximize utility, which implies an existence of a goal. However, this aspect of routines requires more

considerations. People act with a purpose because they want to attain a goal, and they behave in a way that they think is appropriate to reach that goal (Taylor, 1950). Goals give people an intention to act (Ajzen, 1991) and given availability of requisite opportunities and resources (Ajzen, 1985), encoded as other situational features, can such intention result in an enacted behavior that we attempt to model.

Hodgson (1997) also does not specify the granularity of situations and actions. However, behavioral theories, such as Activity Theory (Kuutti, 1995), often consider activities people perform at different levels of granularity. Picking the right granularity depends on the activity we want to study. Pentland & Rueter's (1994) definition explicitly accounts for this by breaking down routines into different hierarchical levels made up of activities at different levels of granularity. Such activities are made up of sequences of actions people perform in different situations that are characteristic of the routine. Chaining different pairs of situations and actions in Hodgson's (1997) definition can broadly encompass the sequential nature of routine actions in such activities (Pentland & Rueter, 1994).

Hodgson's (1997) definition implies that if a situation reoccurs so will the action that the person performs in that situation. This gives rise to the idea of recurrence of behaviors that characterize routines as people repeatedly perform those behaviors (Agre & Shrager, 1990). Hodgson's (1997) definition further implies a rigid, one-to-one mapping between situations and actions, which suggests that repeated behavior instances will also be characterized by same rigidity (Brdiczka *et al.*, 2010). However, routines, like most other kinds of human behaviors, have high variability (Ajzen, 1991). Thus, a unified definition must consider that routines may vary from enactment to enactment (Hamermesh, 2003). Also, people adapt their routines over time (Ronis *et al.*, 1989) based on feedback from different enactments (Feldman & Pentland, 2003).

Thus, we identify four main properties of routine behaviors in existing work: 1) structure that defines the relationships and transitions between situations and actions, 2) ordering of situations and actions within those structures, including the inherit variability of those orders, 3) granularity, and 4) the motivation for acting in a routine manner.

4.2 Unified Routine Definition

We use the four properties of routines as a starting point to define high-level structure of routines and to clarify and scope down the existing routine definitions. Put together, they form a unified definition of routine behavior that can be operationalized into a computational model of routines. We, thus, propose our own unified definition of routine behavior:

> *Routines are likely, weakly ordered, interruptible sequences of causally related situations and actions that a person will perform to create or reach opportunities that enable the person to accomplish a goal.*

Our unified definition strongly builds on Hodgson's (1997) definition to give structure and ordering to routine behavior, while at the same time allowing for variability in behavior. We do this by introducing a probability distribution over the situation and action pairs (corresponding to rules in Hodgson's (1997) definition). The probability distribution of situations and actions and the behavior structures they form are still characterized by causal relations between features of situations and actions that help describe and explain routines, that Hodgson's (1997) insists on. Similarly, the probability distribution of transitions between situations and action implies ordering of sequences.

However, our definition gives meaning to such ordering by explicitly stating that the order (and continuity) or a routine is driven by user preference given actions that are possible in the environment. Thus, we specifically require that situations and actions include information about people's goals and opportunities to accomplish those goals. Unlike some other definitions, we do not attribute recurrence and repetitiveness to routines directly, but to features of situations in the environment (*i.e.*, if situations repeat, so will corresponding actions).

We leave the features of situations and actions unspecified because finding such features in the data is dependent on the domain and research question that stakeholders want to answer. We leave the granularity of such features unspecified for the same reasons. As such, our routine definition implies that situation and actions must be represented at the

lowest level of granularity allowed by either the domain or the data used to compute a future model of routine behaviors.

4.3 Unified Routine Definition and Existing Models of Routines

The existing routine models (Eagle & Pentland, 2009; Farrahi & Gatica-Perez, 2012; Li *et al.*, 2009; Magnusson, 2000) do not capture all aspects of our unified definition of routines. As well, none of the models clearly differentiate between situations and actions. They can either consider events that describe situations only (actions are implicit) or events that are a combination the two. This limits the ability of such models to describe and explain routines because they do not explicitly model the causal relationships between situations and actions that define and describe routines. Also, by treating situations and actions separately allows the stakeholders to understand their separate effects and target interventions at people or their environments. Each of the existing approaches focuses on modeling limited aspects of routine behaviors. For example, Li's *et al.* (2009) model of routines exactly matches Pentland & Rueter's (1994) definition of routines as grammars of behavior. However, that means that their model is only able to encode routine activity hierarchy (sequences of activities at different granularities), but not the causal relationships that define routines.

Both T-patterns (Magnusson, 2000) and Eigenbehaviors (Eagle & Pentland, 2009) focus on temporal relationships between multivariate events. By considering time, they still implicitly model other aspects of the routines. For example, sequences of situations and actions can be expressed over a period of time, and recurrent situations are often periodic. However, this is a limited view of routines because there are clearly other forces that influence people's free choice to act (Hamermesh, 2003). Although time may be correlated with many aspects of routine behaviors, it may not necessarily have a causal effect on people's actions. For example, people will attend a scheduled weekly meeting because of social interactions and norms, but not simply because of a specific day of week and time of day (Weiss, 1996).

From this we conclude that while these algorithms are helpful for extracting routines from behavior logs, they are not sufficient for providing a holistic understanding of a routine behavior. Thus, we propose a new model that is grounded in our unified definition of routine behavior.

5 COMPUTATIONAL MODEL OF ROUTINE BEHAVIOR

In this section, we present an approach to automatically extract and model routines from large behavior logs. Our model captures all aspects of routines as detailed by our unified definition of routines. We model routines as likely, weakly ordered, interruptible sequences of situations and actions encoded as a Markov Decision (MDP) (Puterman, 1994). Traditionally, an MDP consists of set of states, representing situations, a set of possible actions that agents can freely chose to perform in those situations, and a set of possible transitions into new situations resulting from those actions. After performing each action, the person transitions to a new situation that reflects the effects of the action and other factors the person has no control over on their environment. We refer to such transitions as world dynamics. This allows us to encode all possible behavior instances, i.e., all possible ordered sequences of situations and actions, and their likelihood. We learn the probabilities of actions and situation transitions from the data to identify and differentiate routine variations (likely behavior instances) from instances characteristic of deviations and other uncharacteristic behaviors.

Our contribution to modeling routines is our insight that the *byproducts* of MaxCausalEnt (Ziebart, 2010), a decision-theoretic algorithm typically used to train MDP models from behavior logs and predict people's activity, capture the relationship between people' estimated reward function and the likelihood of an action in different situations in which people perform those actions. Using MaxCausalEnt (Ziebart, 2010), we can build a probabilistic model of routines that, unlike models that extract only the most frequent routines, also captures likely variations from those routines, even in infrequent situations. Our approach does this by modeling probability distributions over all possible combinations of situations and actions. Our approach supports both individual and population models of routines, providing the ability to identify the differences in routine behavior across different people and populations.

We later show that the probabilistic nature of the model allows the stakeholders to find supporting evidence for their conceptual model (their understanding of routines) by: 1) *automatically detecting* which behavior instances in the data are characteristic of a routine

(*e.g.*, aggressive driving routine) and which ones are deviations, 2) *automatically generating* example behavior instances that are characteristic of a routine, and 3) *automatically predicting* outcomes characteristic of a routine (*e.g.*, whether routines a cancer patient who has undergone a surgery will lead to rehospitalization). This also allows comparing routine variations across two competing routines (*e.g.*, aggressive driving routine *vs.* non-aggressive driving routine). In later chapters, we show how the model automates the information foraging loop (Pirolli & Card, 2005) by automatically searching for salient patterns that form routine variations. Stakeholders can inspect these patters to understand characteristic behaviors that describe a routine to develop a conceptual model of a routine. Our general model helps stakeholders make sense of routine behaviors across different domains.

5.1 Model of Human Routine Behavior

We model demonstrated routine behavior using a Markov Decision Processes (MDP) framework (Puterman, 1994). MDP is particularly well suited for modeling human routine behavior because it explicitly models people's situations, the actions that they can perform in those situations, and the preferences people have for different situations and actions (where situations with high preference imply user goals). We represent a routine model as a tuple (reminiscent of a Markov decision process):

$$\mathcal{M}_{MDP} = \big(S, A, P(s'|s,a), P(a \mid s), R(s,a)\big) \tag{1}$$

It consists of a set of situations S ($s \in S$) representing context, and actions A ($a \in A$) that a person can take. In addition, the model includes an action-dependent probability distribution for each situation transition $P(s'|s,a)$, which specifies the probability of the next situation s' when the person performs action a in situation s. This situation transition probability distribution $P(s'|s,a)$ models how the environment responds to the actions that people perform in different situations. When modeling human behavior, the transitions are often stochastic (each pair (s,a) can transition to many transition situations s' with different probabilities). However, if people have full control over the environment, they can also be deterministic (*i.e.*, for each pair (s,a) there is exactly one transition situation

s' with probability 1.0). Finally, there is a reward function $R(s, a) \rightarrow \mathbb{R}$ that the person incurs when performing action a in situation s, which represents the utility that people get from performing different actions in different contexts.

People's behavior is then defined by sequences of actions they perform as they go from situation to situation until reaching some goal situation. In an MDP framework, such behavior is defined by a deterministic policy ($\pi: S \rightarrow A$), which specifies actions people take in different situations. Traditionally, the MDP is "solved" using algorithms, such as value iteration (Bellman, 1957), to find an optimal policy (with the highest expected cumulative reward). However, our goal is to find the expected frequencies of different situations and the probability distribution of actions given situations $\big(P(a|s)\big)$ instead—information necessary to identify people's routines and variations.

5.2 Data Modeling and Feature Engineering

Data in behavior logs often consists of sequences of sensor readings, which we convert into sequences of discrete events represented by situation-action pairs. Unlike most of the existing routine models, we specifically differentiate which changes to environment we can attribute to situations (*i.e.*, context) and which ones to actions that people can perform in those situations. This explicit separation also helps us capture effects of the environment on people' behaviors. From the raw data, stakeholders can define: 1) a set of situations S defined by a set of features $\mathcal{F}_{s_t}$ which represent context, 2) a set of actions A defined by a list of binary features $\mathcal{F}_{a_t}$ which represent activities that the people can perform. At any discrete event step, the situation features contain values of all the contextual sensor readings at that event, and actions contain feature values describing the activity the people performed at that event. For example, when modeling driving behaviors, speed sensor reading could be used to express the current discretized speed of the vehicle in the current situation, while throttle sensor could describe how much the driver presses on the gas pedal to maintain or change that speed. We automatically convert raw data from behavior logs into behavior instances that we can use to train our routine model.

Our current modeling approach considers discrete categorical features that uniquely describe situations and actions that we need to study. Each feature in our model is a binary feature that can be true or false (1 or 0). Thus, both situations and actions can be represented using vectors $\mathcal{F}_{s_t}$ and $\mathcal{F}_{a_t}$ of such binary features. Consider a simple example behavior model that captures users' interaction with a mobile device screen that can be off or on. In our model this would result in a situation with two binary features which can be 1 or 0: one to indicate if the screen is on and another to indicate if the screen is off. It is important to note that even though these two features are mutually exclusive, we still need to represent both as either 1 or 0. This is because, in our MDP model, we assume a parametric reward function that is linear in $\mathcal{F}_{S,A}$, given unknown weight parameters θ:

$$R(s, a) = \theta^T \cdot \mathcal{F}_{s_t, a_t} \tag{2}$$

This associates each feature with a weight that signifies the preference for that feature (*e.g.*, how much the user prefers to have the screen on *vs.* how much the user prefers to have the screen off). Each time the person performs an action in a current situation the person incurs the reward based on Equation 2. The final reward associated with performing any behavior instance is thus equal to the sum of all rewards for each situation-action pair in the sequence. We assume that people behave in a way where preferred sequences will result in lager reward than others. When trying to recover this reward from data, we assume that frequently "visited" features are the preferred ones.

5.3 Learning Routine Patterns from Demonstrated Behavior

In this section, we explore how the MaxCausalEnt algorithm (Ziebart, 2010), an algorithm typically used to predict human behavior (Ziebart, *et al.*, 2008; Ziebart *et al.*, 2009), can be applied in a novel way to extract routine behavior patterns that match our definition of human routine behavior and relevant economic considerations from observed behavior data. MaxCausalEnt algorithm (Ziebart, 2010) makes its predictions by computing a policy (π: $S \rightarrow A$) that best predicts the action people take in different situations. Our main contribution is our insight that in the process of computing this policy, MaxCausalEnt algorithm (Ziebart, 2010) computes two other functions that express how likely it is that a

situation and action are part of a routine: 1) the expected frequency of situations (D_s), and 2) probability distribution of actions given situations $\big(P(a|s)\big)$. We now describe how we compute these two functions and how they relate to routines.

Inverse Reinforcement Learning (IRL) (Ng & Russell, 2000) approaches, which MaxCausalEnt (Ziebart, 2010) is based on, assume that people assign a utility function (modeled as the reward functions $R(s,a)$), which they use to decide which action to perform in different demonstrated situations. Each situation and action combination in our MDP model is expressed by a feature vector $\mathcal{F}_{S,A}$. For example, in an MDP that models daily commute routines, situations can have features that describe all possible locations that a person can be at, and actions can have features that describe if the person is staying at or leaving the current location. As is common for IRL algorithms (Ng & Russell, 2000; Ziebart, Bagnell, & Dey, 2013), we assume a parametric reward function we defined in Equation 2.

We begin the process of recovering the expected situation frequencies (D_s) and probability distribution of actions given situations ($P(a|s)$) by trying to learn the person's reward functions $R(s,a)$ from demonstrated behavior. This problem reduces to matching the model feature function expectations ($E_{P(S,A)}[\mathcal{F}(S,A)]$) with demonstrated feature expectations ($E_{\tilde{P}(S,A)}[\mathcal{F}(S,A)]$) (Abbeel & Ng, 2004). Intuitively, this means that the model will have the same preference for different situations and actions as the people whose behaviors we are modeling. To match the expected counts of different features, we use MaxCausalEnt IRL (Ziebart, 2010), which learns the parameters of the MDP model to match the actual behavior of the person. Unlike other routine modeling approaches described earlier, MaxCausalEnt explicitly models the causal relationships between situations and actions, and keeps track of the probability distribution of different actions that people can perform in those situations.

To compute the unknown parameters θ, MaxCausalEnt (Ziebart, 2010) considers the causal relationships between all the different features of the situations and the actions. The Markovian property of MDP, which assumes that the actions a person performs only depend on the information encoded by the previous situation, makes computing the causal

relationships between situations and actions computationally feasible. MaxCausalEnt (Ziebart, 2010) extends the Principle of Maximum Entropy (Jaynes, 1955) to cases where information about probability distribution is sequentially revealed, as is the case with behavior logs. This principle ensures that the estimated probability distribution of actions given situations $\big(P(a|s)\big)$ is the one that best fits the situation and action combinations from the sequences in the behavior logs.

MaxCausalEnt IRL maximizes the causal entropy ($H(\boldsymbol{A}^T \parallel \boldsymbol{S}^T)$) of the probability distribution of actions given situations $\big(P(A_t|S_t)\big)$:

$$\underset{P(A_t|S_t)}{\operatorname{argmax}} H(\boldsymbol{A}^T \parallel \boldsymbol{S}^T) \tag{3}$$

such that:

$$E_{P(S,A)}\left[\mathcal{F}(S,A)\right] = E_{\tilde{P}(S,A)}[\mathcal{F}(S,A)]$$

$$\forall_{S_t,A_t} P(A_t|S_t) \geq 0$$

$$\forall_{S_t,A_t} \sum P(A_t|S_t) = 1$$

The first constraint in the above equation ensures that the feature counts calculated using the estimated probability distribution of actions given situations ($P(A_t|S_t)$) matches the observed counts of features in the data, and the other two ensure that $P(A_t|S_t)$ is an actual probability distribution.

Using the action-based cost-to-go (Q), which represents the expected value of performing action a_t in situation s_t, and situation-based value (V) notation, which represents the expected value of being in situation s_t, the procedure for MDP MaxCausalEnt IRL (Ziebart, 2010) reduces to:

$$Q_\theta^{soft}(a_t, s_t) = \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \cdot V_\theta^{soft}(s_{t+1}) \tag{4}$$

$$V_\theta^{soft}(s_t) = \underset{a_t}{\operatorname{softmax}} Q_\theta^{soft}(a_t, s_t) + \theta^T \cdot \mathcal{F}_{s_t,a_t}$$

Note that this is similar, but not the same as stochastic value iteration (Bellman, 1957), which would model optimal and not observed behavior. The probability distribution of actions given situations is then given by:

$$P(a_t|s_t) = e^{Q_\theta^{soft}(a_t,s_t) - V_\theta^{soft}(s_t)} \quad (5)$$

The probability distribution of actions given situations $P(a|s)$ and the situation transition probability distribution $P(s'|s,a)$ are used in a forward pass to calculate the expected situation frequencies (D_s). This optimization problem can then be solved using a gradient ascent algorithm. Ziebart (2010) provides proofs of these claims and detailed pseudocode for the algorithm above.

5.4 Validating the Model of Human Routine Behaviors

In this section, we show how stakeholders can evaluate the quality of behavior patterns extracted using our model on two example data sets. We show how stakeholders can ensure that the routine actions we extract are predictive of most behaviors in the data; *i.e.*, that the algorithm is sufficiently predictive for modeling routines. Accuracy of this prediction task also quantifies the variability of the routines in the model, where high accuracy suggests low variability. It also shows that the extracted routines generalize to situations and actions that are not present in the training data. Then, we show how stakeholders can make sure that the routines extracted using our approach are meaningful.

During the evaluation process, stakeholders may want to answer specific questions about a routine behavior captured in the model. We express this knowledge in three research question that stakeholders may want to answer:

1. *What is the full complexity of routine behavior?* (*RQBeh):* To make sense of routines, it is critical to discern all aspects of routine behavior from the model. This includes finding relationships between different features of situations and actions, and learning which features describe opportunities people seek to accomplish their goals (both modeled as features that people have a demonstrated preference for).

2. *What are variations that are characteristic of a routine?* (*RQVar)*: Routines are characterized by routine variations—behavior instances that are characteristic of that routine. Therefore, stakeholders must be able to differentiate such variations from deviations and other uncharacteristic behaviors.
3. *How do routines compare across individuals and populations? (RQComp):* An important part of understanding a particular routine is the ability compare the routine within and between individuals and populations. For example, to understand routines of aggressive drivers, it is important to compare it against routines of non-aggressive drivers.

Part of the modeling process involves making sure that a routine model extracted meaningful routines from behavior logs. Stakeholders can do this by searching for evidence and relationships in the patters of behaviors captured in the model. Using two different existing human activity data sets, we evaluate the ability of stakeholders to make sense of different types of routines extracted using our approach from diverse types of behavior logs: people's daily schedules and commutes (Davidoff *et al.*, 2010) and activities that describe how people operate a vehicle (Hong *et al.*, 2014). We show that the extracted routine patterns are at least as predictive of behaviors in the two behavior logs as the baseline we establish with existing algorithms. Next, we recruited domain experts who work with human activity and routine data to verify that patterns extracted using our approach are meaningful and match the ground truth reported in previous work (Davidoff *et al.*, 2010; Hong *et al.*, 2014). For this task, we developed a visual analytics tool that enables domain experts to visually explore and compare the routines extracted using our approach.

5.4.1 Training Models of Routine Behavior

We illustrate our routine modeling approach on two previously collected data sets from the literature that contain logs of demonstrated human behavior. The first data set contains daily commute routines of all family members from three two-parent families with children from a mid-sized city in North America (Davidoff *et al.*, 2010). The data set was used to predict the times the parents are likely to forget to pickup their children (Davidoff *et al.*, 2011). The other data set contains driving routine behavior of aggressive and non-

aggressive drivers as they drive on their daily routes (Hong *et al.*, 2014). The data set was used to classify aggressive and non-aggressive drivers.

We picked these two data sets to show the generalizability of our approach to different types of routines. The two data sets contain routine tasks people perform daily, but that are very different in nature. The family daily routine data set incorporates the traditional spatio-temporal aspect of routines most of the existing work focuses on. The driving data set contains situational routines that are driven by other types of context (*e.g.*, the surrounding traffic, the current position of the car in the intersection).

The two data sets also differ in granularity of the tasks. The commute routines happen over a longer period of time and the granularity of the task is very coarse with few actions that people can perform in different contexts (*e.g.*, stay at the current place or leave and go to another place). The daily routines are therefore defined by the situations the people are in. The aggressive driving data set contains fine-grained actions, which often occur in parallel, that people perform to control the vehicle (*e.g.*, control the gas and brake pedals and the steering wheel). Driving routines are therefore primarily defined by the drivers' actions in different driving situations. The driving data set also showcases the ability of our approach to capture population models (*e.g.*, aggressive drivers *vs.* non-aggressive drivers) and enable comparison of routines across different populations.

5.4.1.1 Family Daily Routines Data Set

Situations when one of the parents is unable to pickup or drop-off a child create stress for both parents and children (Davidoff *et al.*, 2010). To better understand the circumstances under which these situations arise, it is important to identify when the parents are responsible for picking up and dropping off their children (*RQBeh*), when variations occur and how parents handle deviations from such routine situations (*RQVar*). This requires finding and understanding how the parents organize their daily routines around those pickups and drop-offs (*RQComp*).

This data contains location sampling (latitude and longitude) at one-minute intervals for every family member (including children) in three families from a mid-sized city in North America (Davidoff *et al.*, 2010). Location information was manually labeled based on

Table 1. Situation features capturing the different contexts of a daily commute.

Feature	Description
Day	Day of week {M, T, W, Th, F, Sa, Su}
Time	Time of day in increments of 1 hour {0-23}
Location	Current location
Activity	Activity in the past hour {STAYED AT, ARRIVED AT, TRAVELING FROM}

Table 2. Action features representing actions that people can perform when at a location.

Feature	Description
Activity	Activity people can perform in current context {STAY AT, TRAVEL TO}
Location	The current location to stay at or next location to go to

information from bi-weekly interviews with participants. Participants also provided information about their actual daily routines during those interviews.

We converted the location logs into sequences of situations and actions representing each individual's daily commute for each day in the data set. Situation features included the day of the week, hour of the day, participant's current place, and whether the participant stayed at the location from the previous hour, arrived at the location during the current hour, or left the location during the hour (Table 1). Action features included the participant's current activity that could be performed in those situations (Table 2). Participants could stay for another hour, leave the location, and once they have left a location go to another location. The data contained a total of 149 days.

We modeled the situation transition probabilities ($P(s'|s,a)$) as a stochastic MDP to model the environment's influence on arrival time to a destination. The participants could stay or leave a place with 100% probability. Once the participants leave their current location, their arrival time at their destination depends on their desired arrival time and the environment (*e.g.*, traffic, travel distance). To model the influence of the external variables, we empirically estimate the probability that participants have arrived at another place within an hour or not. The median number of situations and actions per family were 14,113 and 85 respectively, for all combinations of possible features.

5.4.1.2 Aggressive Driving Behavior Data Set

Drivers that routinely engage in aggressive driving behavior present a hazard to other people in traffic (AAA, 2009). To understand aggressive driving routines, it is important to explore the types of contexts aggressive drivers are likely to prefer (*e.g.*, turn types, car speed, acceleration) and the driving actions they apply in those contexts (*e.g.*, throttle and braking level, turning) (*RQBeh*). Aggressive drivers might also be prone to dangerous driving behavior that does not occur frequently (*e.g.*, rushing to clear intersections during rush hour (Shinar & Compton, 2004)). Such behavior might manifest itself as different routine variations (*RQVar*).

It is also important to compare the routines of aggressive drivers with non-aggressive drivers to understand how aggressive drivers can improve their routine (*RQComp*). To understand those differences, it is not enough to compare the situations both groups of drivers find themselves in, but also the actions that drivers perform in those situations. This is because both aggressive and non-aggressive drivers can attain similar driving contexts, but the quality of the execution of driving actions may differ. For example, both types of drivers might stop at a stop sign on time, but aggressive drivers might have to brake harder or make more other unsafe maneuvers than non-aggressive drivers.

This data set contains driving data from 22 licensed drivers (11 male and 11 female; ages between 21 and 34) from a mid-sized city in North America (Hong *et al.*, 2014). Participants were asked to drive their own cars on their usual daily driving routes over a period of 3 weeks. Their cars were instrumented with a sensing platform consisting of an Android-based smartphone, On-board Diagnostic tool (OBD2), and an inertial measurement unit (IMU) mounted to the steering wheel of the car. Ground truth about participants' driving styles (aggressive *vs.* non-aggressive) was established using their self-reported driving violations and responses to the driver behavior questionnaire (Hong *et al.*, 2014). The driving data collected in the study included: car location traces (latitude and longitude), speed, acceleration, engine RPM, throttle position, and steering wheel rotation. Sensor data was recorded every 500 milliseconds.

Table 3. Situation features capturing the different contexts the driver can be in.

Feature	Description
Goals	
Maneuver	The type of maneuver at the intersection {STRAIGHT, RIGHT TURN, LEFT TURN, U-TURN}
Environment	
Position	Current position of the car in the intersection {APPROACHING, ENTERING, EXITING, AFTER}
Rush hour	Whether the trip is during rush hour or not {TRUE, FALSE}
Vehicle	
Speed	Current speed of the vehicle (5-bin discretized)
Throttle	Current throttle position (5-bin discretized)
Acceleration	Current positive/negative acceleration (9-bin discretized)
Wheel Position	Current steering wheel position {STRAIGHT, TURNING, RETURNING}
Turn	Current turn vehicle is involved in {STRAIGHT, SMOOTH, ADJUSTED}

Table 4. Action features representing actions that drivers can perform between stages of the intersection.

Feature	Description
Pedal	Median throttle (gas and brake pedal) position (10-bin discretized)
Throttle Spike	Sudden increases in throttle {NONE, SUDDEN, INTERMITTENT}
Brake Spike	Sudden braking {NONE, SUDDEN, INTERMITTENT}
Turn style	Type of turn driver performed in intersection {STRAIGHT, SMOOTH, ADJUSTED}

We use a subset of this data focused on intersections (where instances of aggressive driving are likely to occur (Shinar & Compton, 2004)). We used location traces of the participants' driving routines to manually label intersections and the position of the vehicle in those intersections. One of the limitations of this data set is that there is no information about other vehicles and traffic signs and signals that represent the environment. We then split the intersection instances into sequences of sensor readings that start 2 seconds before the car enters the intersection, and end 2 seconds after the car exits the intersection. This resulted in a total of 49,690 intersections from a total of 542 hours of driving data from 1,017 trips.

To model situations we combined the driver's goals (*e.g.*, make a right turn), the environment (*e.g.*, position in intersection), and the current state of the vehicle (*e.g.*, current speed) into features of the situations (Table 3). Actions in our model represent how the driver operates the vehicle by steering the wheel, and depressing the gas (throttle) and brake pedals. We aggregate the driver's actions between different stages of the intersection and represent the median throttle and braking level, and note any spikes in both throttle and

braking. We consider the movement of the steering wheel to estimate whether the driver turned in one smooth action, or if the turn required one or more adjustments. Table 4 shows action features in our model. We identified 7,272 different situations and 446 different actions in the data set.

5.4.2 Quantifying Routineness of Human Behavior

Although we are not interested in the predictive power of the MaxCausalEnt IRL *per se*, we use the task of predicting the next action given a situation to evaluate our model's ability to extract routine. Using 10-fold cross validation for each person in each dataset, we compare the performance of this algorithm for extracting routine behavior with a Zero-R algorithm, which always predicts the overall most frequent action, and a first-order Markov Model algorithm, which always predicts the most frequent action for each situation. We chose these baselines because they explicitly establish the frequency of actions in the training set. Matching or exceeding these baselines means that the algorithm has correctly identified frequent routine actions and that the predictive power of the algorithm is sufficiently high to model routines.

The mean accuracy of the MaxCausalEnt on the family daily routines dataset was 0.81 (SD=0.09), compared to first-order Markov Model mean accuracy of 0.66 (SD=0.07) and ZeroR mean accuracy of 0.51 (SD=0.09). MaxCausalEnt algorithm likely outperformed the first-order Markov Model because of its ability to better generalize from training data. The accuracy of MaxCausalEnt algorithm also suggests low variability of routines in people's daily schedules.

The mean accuracy of the MaxCausalEnt on individual models of driving routines was 0.54 (SD=0.05) compared to first-order Markov Model mean accuracy of 0.58 (SD=0.06) and ZeroR mean accuracy of 0.33 (SD=0.06). MaxCausalEnt algorithm and the first-order Markov Model had similar accuracies likely because in each fold the training set was representative of the testing set. However, decision-theoretic guarantees of MaxCausalEnt that ensure it makes the least number of assumptions to fit the observed data make it less likely to overfit the training data in general. Relatively low accuracy of both MaxCausalEnt

and the first-order Markov Model on this data set suggests a lot of variability in the driving routines.

5.4.3 Visual Exploration of the Model

We now show that the routine patterns extracted using our approach match the actual routines of people. To do this, we recruited domain experts who work with machine learning and data mining in the domain of human behavior, and asked them to identify the routines and variations extracted using our approach. We then confirmed that those patterns matched the ground truth behaviors established in the existing work (Davidoff, 2010; Hong *et al.*, 2014). This allowed us to verify that the patterns extracted using our approach are meaningful and represent the actual routines.

5.4.3.1 Study Software

To make the routine behavior models created using our approach accessible to participants and allow them to investigate the extracted routine patterns, we developed a simple visualization tool. To maintain a level of familiarity, we base our visual encoding of routine behavior elements on a traditional visual representation of an MDP as a graph (Figure 1). Our MDP graph contains nodes representing situations (as circles) and actions (as squares), directed edges from situation nodes to action nodes (indicating possible actions people can perform in those situations), and directed edges from actions to situations (indicating situation transitions for any given situation and action combination).

To enable participants to see changes in features of situations and actions, we encode situation features and action features as a series of color-coded circular marks arranged in a spiral shape within the nodes. Each feature has a dedicated hue. Feature values that are present in the node are represented by a dark shade, and feature values not present in a light shade of that color. A dark boundary serves as a separator between features. More details, in text are always available simply by moving the cursor over a node (Figure 1.C).

To show frequent behaviors in the model, we visually represent the probability of different graph elements using line thickness. Thickness of the outside line of the situation and action nodes encodes the frequency of that situation in a behavior sequence (D_s), where thicker

lines indicate situations that are likely to be part of a routine. Similarly, the thickness of the edges encodes the probability of that edge. Thickness of edges from situations to actions is given by the probability distribution of actions given situations ($P(a|s)$), and represents the influence of each situation on the choice of actions. The thickness of edges from actions to situations is given by the probability of transition ($P(s'|s,a)$).

To layout the nodes, we sort the initial situations from the demonstrated sequences by their frequency (D_s) in descending order. We then use a version of the depth-first search algorithm, starting from the initial situation nodes, that traverses nodes by first iterating over edges in order from highest to lowest probabilities $P(a|s)$ and $P(s'|s,a)$. Situation nodes are never duplicated (*i.e.*, there is exactly one node in the layout for each situation in the model), whereas action nodes are duplicated for each situation.

When the participant selects a data set and population, the tool provides the initial layout of routines extracted using our algorithm. This shows the most important information about the extracted routines. However, to further analyze the routine behavior, the participant must be able to explore the details of routine variations filtered out as aggregate nodes (Figure 1.B.3). For example, the participant might want to find which of the parents' routines include locations where they pickup and drop off their children.

Aggregated items contain valuable information about potential routine variations. For example, a child might go to her grandparents or their friend's house on Wednesdays after school; two variations on the same routine that occur with similar probability. To show all possible variations in an aggregate, the participant can click on the aggregate to expand its content. To mark an aggregated node as a variation of interest, the researcher can *pin* that aggregated node by clicking on it, thus removing it from all of its aggregate parents. When the researcher holds Alt-key and clicks on an aggregated node, this pins all the nodes on the most likely sequence of situations and actions, determined by the probabilities of edges between the two. The pinned sequence, starting from the clicked node to the sequence end node, represents a routine variation. Pinned nodes are identified by a gray glow effect. All nodes that are part of the extracted routine are automatically pinned, and all other nodes

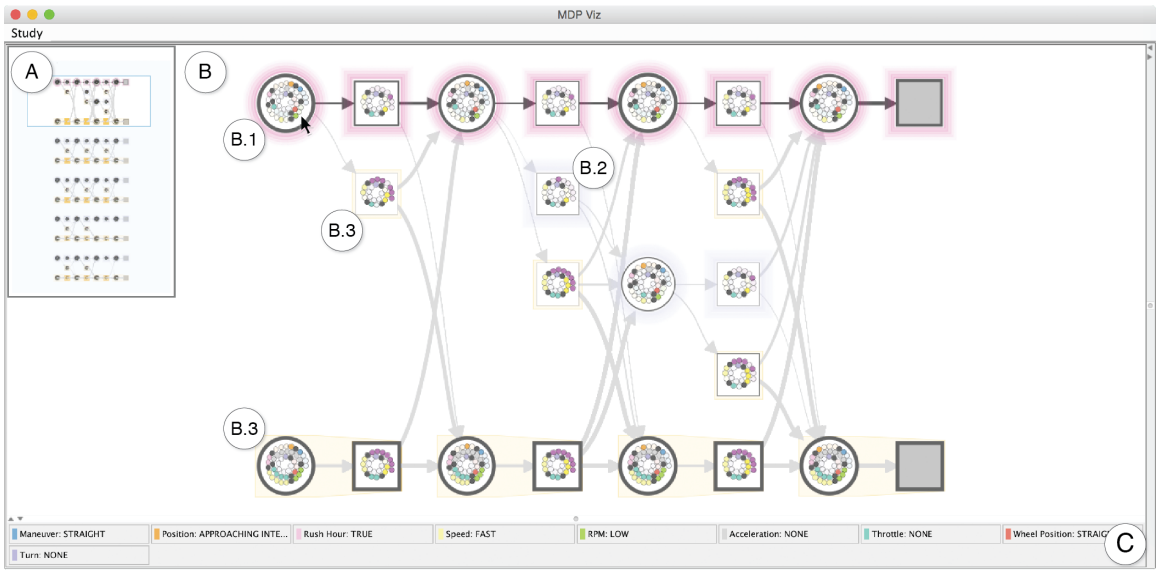

Figure 1. Study software user interface showing the main routine and one likely variation of non-aggressive drivers extracted using our approach: A) overview panel, B) the main display area containing subgraphs representing automatically extracted routine sequences of situations (circles) and actions (squares), B.1) the user is hovering over a node to highlight extracted routine (nodes highlighted in purple and dark gray edges), B.2) an action that starts a variation from the main routine B.3) aggregate items representing possible extracted variations, and C) details panel showing information about visual elements on demand.

are unpinned. Clicking on a pinned node unpins it, which returns the node into the aggregate.

To determine whether or not to pin the node, the researcher can review the features of individual and aggregate nodes by hovering over them. In addition to showing the details of individual nodes in the details panel (Figure 1.C), hovering over nodes shows relationships between different elements of the routine. Hovering over any node highlights the most likely routine path from an initial situation to the collecting node that contains the hovered over node (Figure 1.B.1). This makes it easier to understand the routine situations and actions in the area of interest.

5.4.3.2 Participants

To verify the routines, we have recruited 8 researchers (5 male and 3 female) that have had experience with machine learning and data mining, or have worked in the domain of

activity recognition and human routine modeling. The participants included Ph.D. students, Postdoctoral fellows, Research Scientists, and Professors working or visiting at our local University. All participants had experience with machine learning, 2 with data mining, 4 with activity recognition, and 1 worked specifically on modeling human routine behavior. The participants were compensated $25 for taking part in the study.

5.4.3.3 Method

When participants arrived at our lab, we briefed them on the purpose of the study and they signed a consent form. They then filled out a short questionnaire asking them about their occupation and experience with relevant research topics. We then demonstrated the visual tool to the participants and allowed them to practice using it for approximately 20 minutes. Participants had to complete two tasks: 1) identify daily routine for a randomly chosen person and weekday from the daily routines data set, and 2) identify the differences between routines of non-aggressive and aggressive drivers from the driving data set. The first task took approximately 20 minutes, and the second task about an hour. Total study time was approximately between one and a half and two hours.

For the two tasks, we asked participants to identify routines and differences between the routines without presenting them with the ground truth. We did this to avoid biasing the participants towards trying to match the routines presented in the tool with what we might have told them is the correct answer. We then compared their answers with the ground truth from the previous literature to verify that the tool extracted the right and meaningful routine. Because the main purpose of the study was to validate the extracted routines and not evaluate the usability of the tool, participants could ask clarifying questions about the tool and the user interface at any point during the tasks.

5.4.3.4 Ground Truth

In the first task, we used the family daily commute routines model to understand the patterns of pickups and drop-offs. We compare the findings of our participants with the ground truth and discussion provided by Davidoff *et al.* (2010). In this data set, the ground truth represents self-reported daily commute routines for all family members in all families that took part in the study. Family members reported their location and the time they usually

arrive and leave that location. Davidoff *et al.* (2010) then manually annotated and confirmed the routines in the raw sensor data.

In the second task, we used the driving routines of aggressive and non-aggressive drivers that Hong *et al.* (2014) identified in their data set. Hong *et al.* (2014) used their intuition and expert knowledge of driving behaviors to separately compare the distributions of each sensor stream in the raw data to gain insight about aggressive driving styles. They found that aggressive drivers drive faster than non-aggressive drivers, and that they experience higher acceleration than non-aggressive driver (*i.e.*, they are more likely to press hard on the gas and brake pedals). Additionally, they found more variability in the behavior of aggressive drivers than non-aggressive drivers.

5.4.3.5 Results

Our results show that our approach extracts meaningful patterns of routine behavior. The participants were able to point out the patterns that form the high-level routines present in the ground truth for both tasks (*RQBeh*). For the daily routines task, this means that they successfully listed the locations and times of the routines of people in the daily routine data set. However, two participants identified two separate patterns where the locations and times reported as part of the main daily routine did not correspond to the ground truth. After careful examination, we found that the participants wrongfully identified the actions as part of the routines because the events were part of infrequent routine variations that the people in the original study did not report in the ground truth. The algorithm correctly assigned low probabilities to those actions, but the participants did not notice this. This is likely an issue with the visualization rather than the algorithm itself, and is something that can be addressed with more training with the tool.

In addition to simply pointing to patterns that represented correct routines, participants also generated some insights for themselves. For example, all six participants that were presented with a parent's daily routine that contained a child pickup or drop-off specifically pointed out this activity. Also, three participants, that had a case where the parent drops off the child as part of his or her routine, but does not also pick the child up, correctly explained

that the other parent was likely responsible for the pickup, without seeing the other parent's routines.

In the driving data set, participants pointed to the patterns that form the main routines (*RQBeh*) and variations in driving behavior (*RQVar*) of both aggressive and non-aggressive drivers. All participants pointed to patterns that show that aggressive drivers are more likely to drive faster through intersections than non-aggressive drivers. Five participants showed the patterns of routine variations where aggressive drivers are likely to increase their throttle just before entering and leaving intersections. Participants pointed those out as the main differences between the two populations (*RQCom*). Two participants also pointed to the probabilities of routine variation patterns extracted using our approach that suggest that aggressive drivers are less consistent in their behavior than non-aggressive drivers. Participants likely drew their conclusions from the model, but might also have a preconceived notion that acceleration and speed are correlated with aggressive driving. However, even if our participants had preconceived notions, they could verify and document them using our model.

Although evaluation of the visualization tool was not our main goal, 2 participants mentioned that such a tool would help them explore and understand human routines. One participant, who studies routine behaviors, pointed out that the organization of routine patterns and variations helped him clarify his understanding of what constitutes routines and how they manifest themselves in people's activity.

5.5 Summary

In this section, we presented a novel approach for modeling human routine behavior from behavior logs that explicitly models all aspects of routines as defined by our unified definition of routine behavior. This model automatically conceptualizes behaviors from data in behavior logs into a computational model of routines. Stakeholders could use this model to explore and understand routine behaviors. The ability to automatically detect and generate behavior instances automates a fundamental task in the behavior sensemaking process: finding evidence to support that the model captures patterns in the data that

represent behaviors and not some other patterns in the data. In the next two sections, we show how a computational model of routine behaviors supports the sensemaking process.

We showed how our models trained using MaxCausalEnt algorithm (Brian Ziebart, 2010) can extract patterns of routine behavior from demonstrated behavior logs. This is a novel application of an algorithm that was designed to predict human behavior. In our evaluation of the algorithm we found that its ability to predict routines from the two example data sets was sufficient for modeling routines. The ability of MaxCausalEnt algorithm to generalize from small sample sizes enabled it to beat the baseline in the daily routine data set. The performance of the algorithm was comparable with the first-order Markov Model in the aggressive driving data set. This is likely because the training data happened to match the testing data well. However, this is not safe to assume in general case, and MaxCausalEnt's decision-theoretic guarantee that it will not overfit the observed data make it a better choice for modeling routines than the first-order Markov Model.

Our visualization tool helps domain experts validate the ability of our approach to extract meaningful routines. The participants were able to explore situations, actions, and the relationships between the two, to correctly identify routine behaviors (*RQBeh*) and explore routine variations in the data (*RQVar*). Their findings matched the previous research. They pointed to different relationships that establish the differences between routines of the two driver populations (*RQComp*). Although we carefully designed our tool, our goal was not to formally evaluate its usability. We did not notice any usability issues that prevented participants from learning the elements of the model. We found that the participants knew how to progress towards understanding the routines.

Our results imply that domain experts can use the patterns extracted using our approach to more quickly identify major aspects of routines by visually inspecting them, even after only short amount of training, compared to previous work. For example, Davidoff *et al.* (2011) performed tedious manual labeling of routine and routine variation patterns in the raw data based on feedback from the participants before presenting the patterns on a timeline. Hong *et al.* (2014) performed Exploratory Data Analysis in which used their intuition and expert knowledge of driving behaviors to separately compare the distributions

of each sensor stream in the raw data to gain insight about aggressive driving styles. Our participants had to only explore the patterns extracted using our approach.

The knowledge that the researchers gain about routine behaviors through exploring our models can inform the design of interventions that help people improve their routines. For example, the knowledge that aggressive drivers are likely to use higher throttles can inform the design of in-car systems that monitor the throttle and make the driver more aware of this aggressive behavior through subtle ambient notifications. Another advantage of our approach is the underlying MDP-based model, which can be used to power human-data supported interfaces that automatically classify current behaviors and prescribe new actions that improve existing routines.

6 CAPTURING ECONOMICS OF ROUTINE BEHAVIOR

In this section, we modify the basic MDP framework and its reward function representation to capture the specific economics that drive people's behavior and their choice of actions in different situations. In Chapter 5, we used simple behaviors that allowed us to encode the structure of routine behaviors by specifying strict world dynamics. This allowed us to express the ordering and continuity of possible behavior instances in our routine model as features of the environment. In our driving examples the order of situations is fully defined by the position of the vehicle in the intersection and no deviations from this pre-set order that would disrupt the continuity of behavior were present in the model. However, many existing domains that are of interest to the HCI community contain behaviors where ordering and continuity is not well understood and needs to be estimated from the data.

Here we detail our method for choosing and expressing different features of situations and actions that allow us to encode this broader category of behaviors in an MDP according to our definition of routines. Note that because we model people's enacted behaviors, our approach may differ from other domains in which the goal of using an MDP is to find the optimal behavior of an agent, such as a robot. Instead, we model underlying motivations that people have for exhibiting actions that open up opportunities that allow them to reach certain desired, goal situations. In our method, the choice of possible actions that people perform and the order of those actions is driven by a desire to accumulate reward. Thus, behavior instances that allow people to accomplish a goal (*i.e.*, reach a goal state) will accumulate more reward than other competing behavior instances that do not. We employ two basic principles in economics, opportunity cost and diminishing returns, to provide reasons behind order and alternation between competing behavior instances and routine variations they form. We assume that people choose between competing actions that they can perform, and thus competing sequences of behaviors, by considering the opportunity cost of another competing routine variation in light of diminishing returns of reaching and remaining in a goal situation.

As with other computational models that are used to study a complex system, we pick and engineer features based on hypotheses about what influences behaviors and various

outcomes. For example, clinicians have hypothesized that lack of activity is correlated with rehospitalization of patients who have undergone surgery to remove cancer (Low *et al.*, 2018) To build a computational model that helps clinicians study whether the lack of mobility leads to rehospitalization requires including both the outcome feature (rehospitalized or not) and a feature that expresses patients' activity levels. However, historically pain and nausea have been identified as one of the main predictors of rehospitalization and should therefore be included in the model to avoid confounds. Although there is no "one-size-fits-all" approach to modeling routines across domains, in the following section, we discuss what feature engineering is required to ensure that we can capture different aspects of behaviors in our definition of routines. In later chapters we will detail domain-specific considerations for various routine behaviors across different domains.

6.1 Extending Situation and Action Feature Engineering

Here we build on our basic definition of situation and action features from Chapter 5 and extend it to account for economics of human behavior. Although MaxCausalEnt IRL (Ziebart, 2010) estimates the probability of behaviors using a reward function, it is dependent on feature engineering to ensure that the reward function can express the actual economics of behavior. In our previous chapters, we showed that such considerations are not as important when the economics of behavior (*e.g.*, ordering and continuity of routine variations) are fully implied by world dynamics. However, often world dynamics do not fully capture the ordering and continuity of routine variations. We show how to engineer features in a way that allows us to apply reward function from Equation 2 to the problem of modeling broader categories of routine behaviors.

6.1.1 Situation Features

As we have seen in earlier chapters, situation features allow us to define all possible states of the environment in which behaviors are situated. As such, situation features broadly define the context, or information that is relevant to the current behavior (Dey, 2001). Whether a feature is relevant or not depends on hypotheses that the computational model

is meant to explore and is thus domain specific. We consider two main categories of situation features that generalize across domains: 1) *dynamic*, which are features that can actively be changed by people's actions or other factors in the environment, and that represent situation features that people actively seek or avoid, and 2) *static*, which represent features that could be used to identify and explore behaviors of different populations or individuals, but that people are indifferent to, unaware off, or that cannot be influenced explicitly by people's actions or the environment.

Dynamic situation features allow us to study how the environment in which behavior is situated changes throughout duration of a behavior instance. Most common dynamic situation features capture the current state of the environment. For example, in the case of mobile device usage routines, those could represent the current state of the screen, battery charge level, and foreground application. However, due to the nature of MDP, which assumes that any action and subsequent transition to a new situation only depends on the current situation, often we need to include historical information into the current situation. Sometimes this historic information is implied by the environment we are modeling. In our mobile device routine example, we only need to include information that the screen is unlocked for the user to be able to start an application because the system implies that the screen is on. However, we may want to include information about applications that the user has interacted with since unlocking the screen because this information is not implicit. As we will see later in this chapter, such historic information can help us capture other economics of behavior drive weak ordering of situations and actions in a behavior instance.

Static features, such as demographic information (*e.g.*, age, gender), help us explore potential differences in behaviors between individuals or populations. For example, to study differences in how users of different ages interact with their mobile devices would require modeling situation feature that indicates users' age group. In the course of interacting with a mobile device we expect that a user's age group will be fixed and thus their preference for their age group is irrelevant in terms of estimating their goal situations and how they generate opportunities to reach those situations. However, as we will see in the later sections, such features can effectively split the model into distinct sub-models of behaviors based on static features. This is similar to training multiple models for each value

of a static feature. However, this may be impractical as the number of features that we want to split models on grows (*e.g.*, adding a static feature that indicates if a user has a secure lockscreen or not to our previous example), in which case they are best included in a single model.

We also define a special kind of static situation features that we refer to as *behavior labels*. Such static features are derived from their dynamic counterparts to label behaviors that have a special quality. To illustrate this kind of feature, consider an example model that captures routine behaviors of patients who have undergone surgery to remove cancer. In this model, patients actively avoid being rehospitalized and thus avoid any situations with a dynamic feature that indicates that they are currently rehospitalized. To contrast the behaviors of patients who have been rehospitalized and those that have not we can add a special behavior label situation feature that splits the original model into two sub-models: 1) those who have been rehospitalized in the course of data collection, and 2) those that have not been rehospitalized in the course of data collection. Such a feature would ensure that there is no crossover between behaviors of the two populations of patients.

6.1.2 Action Features

Actions represent people's atomic behaviors (*i.e.*, behaviors that can be enacted at the same time) that they freely choose given what is possible in the current situation. Take for example plugging a mobile device to charge when a power outlet is available. This is a free choice action that users may perform in response to low battery charge level. In this simple illustration, we can model plugging device into power outlet as an action feature, whereas battery level is a situation feature. Action features could be explicit (*e.g.*, plug the device into power outlet) or implicit (*e.g.*, leave device plugged in). In our example, subsequent changes to battery charge level are a result of the device being plugged in (the effect of the environment) and do not require an explicit action from the user, but instead require an implicit action feature that says that the user is not actively trying to unplug the device.

Like situations, each action is uniquely defined by the combination of its features. To expand our previous example, we could add an action feature that indicates that the user wants to turn the screen on. Atomic actions could also be represented using combinations

of features as long as they can be performed as on atomic action. For example, a feature that indicates that the mobile user is glancing at the mobile lockscreen application could coincide with a feature that indicates that the user wants to unlock the phone using the lockscreen application. However, an action with two features that indicate that the user is interacting with both the lockscreen application and another game application at the same time may not be possible.

The reward function considers peoples' preference for different action features because people develop preference for certain repetitive, well-practiced actions over time, even if that action is not ideal for the current situation. Thus, by including action features in the reward function calculation, we express how people weigh preference for certain actions *vs.* preference for certain situations that are potential outcomes of those actions.

6.2 Representing Time

Time is often defined as progression of events in past, present, and future. As such it plays an important role when modeling human behavior as a sequence of events expressed as situation-action pairs. Properties of routines, such as event duration, ordering, repetition, and frequency are all expressed in relation to time. People also often use time to plan and schedule their behaviors. Therefore, it is no surprise that some existing models explicitly capture relationships between events in time: their duration, time between two events, and frequency of co-occurrence (Magnusson, 2000). However, existing MDP frameworks make it difficult to explicitly model time progression in the model and often treat it as an external factor.

Here, we argue that time is external to behavior and thus can still be effectively represented using an MDP. First, the definition of time assumes existence and ordering of events and not the other way around. Second, the correlation between time and events often does not explain behavior. For example, two people will meet at a certain time, not because of a particular time of day, but because they scheduled an event at that time (*i.e.*, the explanation of the behavior is existence of an agreement between the two people and their desire to reach a goal situation when they are both at the same place at the same time).

Although we have shown how our model allows us to express discrete time as a situational feature (*e.g.*, time of day when modeling daily commute routines in Chapter 5), we often model it as an external factor to the model without a need to include it as a feature of a situation or action. In our model, each situation-action pair represents contextual information at some discrete point in time and action that a person performed at that time without explicitly labeling what that time is. We refer to this atomic (smallest possible time duration of an event in the model) time as a *time tick*. Choice of duration of a time tick depends on data, domain, and intended use of the model to explain behaviors. For example, each time tick could be as small as the collected behavior data allows it: if data contains patients' daily self-reported symptoms once a day, then the model should treat each time tick as one day. In certain domains, such as modeling mobile device usage routines, the smallest duration of an event is expressed in seconds which would be an appropriate duration of a time tick. Alternatively, each time tick in a model could have a different, arbitrary duration if it is driven by another factor. For example, when modeling driving routines, a time tick could occur when a vehicle changes position in an intersection where the actual time duration between two points can vary based on the intersection size and speed of the vehicle.

We can use time as an external factor to express any relevant time-based relation between events. For example, in our model, we relate each situation-action pair in a behavior instance to a time tick which allows us to explore the order and continuity of behavior instances. The duration of any event is the number of time ticks during which a feature or a group of features that defines that has not changed multiplied by the duration of the atomic duration of the time tick (if one is available). We can use similar approach to express time duration between events. Also, two events co-occur at the same time if the features relevant to those events happen approximately on the same time tick. Similarly, we are able to evaluate behavior after any arbitrary time duration. For example, if we want to know what a mobile user would do a minute after turning on the screen, we can simply "fast forward" sixty one-second-long time ticks into the future from the current time tick and look at the resulting situation.

6.3 Economics of Human Routine Behavior

When we use dynamic situation features and action features in our model, we assume that at any time tick a person chooses an action from a subset of possible actions in the current situation. Thus, the choice of action is driven by people's preference for certain features of situations and actions. According to our definition of routine behavior, people choose actions that will allow them to create opportunities that enable them to reach their goals. However, a simple MDP with only positive rewards, as is the case in our model, implies that a person could collect infinite rewards by spending infinitely many time ticks in a cyclic routine variation if world dynamics allow it. Although the influence of world dynamics (i.*e.*, other external processes that people have no control over) may temporarily disrupt people from settling in such an equilibrium cycle, a simple MDP may allow them to choose a sequence of actions to settle back into such a cycle. Therefore, a simple MDP does not consider the opportunity cost of picking another routine variation and diminishing returns of collecting a reward from same features over and over again.

To ensure the model considers the opportunity cost (the loss of reward from alternative routine variations), we assume that behavior instances are finite and that there is a known upper bound on the number of time ticks any behavior instance can last. In MDP terminology, we refer to this type of problem as a finite horizon problem. This means that a person can only accumulate a finite amount of reward in a finite number of steps and thus has to choose actions that maximize such finite reward. Defining what constitutes a finite behavior instance is domain specific, but it is possible even in domains in which a behavior seemingly repeats forever. Our iterative learning algorithm with a finite horizon computes this opportunity cost with our feature engineering.

However, opportunity cost only allows us to capture the economics of choosing one routine variation over another, but not how people choose to interrupt an equilibrium state. For example, in a model of mobile usage routines, users may invoke their most preferred application and use only that application for the duration of the mobile session. However, this is not representative of behaviors that we can observe where users invoke an application, use it for a duration of time, and then switch to another application. The last

example illustrates the concept of diminishing returns where the preference of using an application reduces as the person uses it. Past research (Ziebart *et al.*, 2009) has explored ways to model diminishing returns within the deterministic policy MDP framework by reducing the reward of being in a situation and performing an action for each time tick the agent whose behavior the MDP models stays in the same situation. However, in our model we are interested in stochastic policy $P(a|s)$ that captures the inherent uncertainty in human behavior.

Incidentally, our dynamic situation features and action features engineering together with a stochastic policy already accounts for diminishing returns. To illustrate this, consider a simple example light-switch model that has only two situation features that represent light on and light off, and three action features that represent three possible actions to "turn light on", "keep light on", and "turn light off". In this example model world dynamics are deterministic (*e.g.*, when light is on and the user turns it off, it will result in the "light off" situation with probability 1.0). Now suppose a situation when the light is already on and the user prefers light on over light off. In this case, a deterministic policy would predict that the user will always choose "keep light on". Now suppose that the estimated reward results in a stochastic policy where when the user is in the light on situation, the possible actions "keep light on" and "turn light off" have probabilities $P(keep\ on|light\ on) = 0.9$ and $P(turn\ off|light\ on) = 0.1$. Sampling from this stochastic policy, we see that the probability that the light will be off in the future increases. For example, after one time tick the expected probability that the light is on is $P(light\ on) = 0.9$, but after two time ticks it reduces to $P(light\ on) = 0.81$ and is approximately $P(light\ on) = 0.001$ after 60 time ticks. This shows the diminishing returns of keeping the light on despite the user preferring it over light off. Our algorithm computes a stochastic policy as a function of a reward function estimated from enacted behavior instances in the data.

The last economic consideration is the order of situation-action pairs in a behavior instance, which is driven by external processes in the environment and people's preference. To model the processes in the environment, it is often enough to encode such ordering in the world dynamic (*i.e.*, transitions between situations). For example, in the case of modeling routines of drivers as they navigate an intersection, the order of situations is driven by

different positions of the vehicle in the intersection. In this example, we can assume that the vehicle will approach the intersection, enter the intersection, exit the intersection and then drive away. The order of these situations is implied by the world dynamics, which prevent any other order of vehicle positions in the intersection. However, modeling people's preference for order of action requires us to consider the economics of their behavior.

Using historical dynamic situation features allow us to model the preference that people place on reaching a specific opportunity to accomplish a goal or accomplishing a goal. At the same time, the number of time ticks that the feature is present (*i.e.*, the count of that historic feature in a behavior instance) indicates where in the sequence the event (or situation with a particular historic feature) occurred. For example, historic features with high counts indicate that people are more likely to be prefer them in the beginning of the sequence and those with lower counts towards the end of the sequence.

6.4 Summary

In this chapter, we took a closer look at the optimization function used in MaxCausalEnt IRL (Ziebart, 2010) and identified a list of considerations necessary to capture the economics of routine behavior. Without such considerations, MaxCausalEnt IRL (Ziebart, 2010) would suffer from the same challenges as most existing Data Mining algorithms because there would be no easy way to establish that the patterns of behavior extracted using the algorithms match the actual processes that generated the data. The main considerations that we addressed were ordering and continuity. We have also shown how these two important considerations can be used to introduce the concept of time as an external factor that drives the progression of events in the model.

7 DETECTING AND GENERATING ROUTINES

Studying behavior instances that are characteristic of a routine and showing how they differ from deviations establishes a body of evidence to allow stakeholders to build a conceptual understanding of behaviors. However, data traces contain both instances of routine variations and deviations. Manually exploring and differentiating all routine variations in a model to separate them from deviations is tedious. Without an ability to automatically detect and generate routine behavior instances, stakeholders face significant obstacles in making sense of people's routines.

To help stakeholders identify behaviors that are characteristic of a routine, we perform two detection tasks: 1) automatically detect classes of people based on their routine behaviors, and 2) automatically detect behavior instances are more characteristic of one routine (such as aggressive driving) than another (non-aggressive driving). When labeled data is available, both of those tasks can be performed using supervised Machine Learning methods (Bishop, 2006). However, as discussed before, such algorithmic methods may not base their classification, detection, and prediction on actual behaviors that generated the behavior traces.

Also, when labels are not available (which is often the case when classifying individual behavior instances) they would require labeling each instance individually as belonging to one routine model or another. Semi-supervised approaches (Chapelle, Scholkopf, & Zien, 2009) require labels for a portion of the training data, but labeling even a subset of data is challenging when knowledge about what constitutes a variation of a particular routine is not available beforehand. Automated anomaly detection algorithms (*e.g.*, (Buthpitiya, Dey, & Griss, 2014)), which do not require any manual labeling, focus on deviations from some normal or expected behavior; *i.e.*, they can be used to classify which behavior instances are not routine. However, they do not classify whether a behavior is a variation of one routine *vs.* another. Other unsupervised methods can cluster behavior instances (*e.g.*, (Farrahi & Gatica-Perez, 2012)), but they offer no guarantees that the behavior instance clusters they generate map onto routines.

Our routine model computes the probability of true labels of all variations and deviations in a routine model. This lets us apply probability axioms to automatically detect and generate characteristic behaviors for that model. To address manual labeling challenges, we train our routine models using weakly labeled data, which is an alternative to fully labeled datasets. Weak labels do not necessarily place every instance into the correct class (Ivanov, 2001; Mann & Mccallum, 2010). Instead of labeling each behavior instance individually, we label all instances at once based on the known routine of the person that exhibited those behaviors. For example, if a driver has had traffic violations due to aggressive driving, we would label all of the driver's behavior instances as aggressive. We train a population model using instances for each unique routine label (*e.g.*, one model for aggressive and one for non-aggressive driving). We then classify new instances into one model or the other, given knowledge about the probabilities that each instance will occur. We use the same model probabilities to sample (generate) behavior instances.

7.1 Detecting Classes of People Based on Their Behavior

Here we show how to classify people based on their behaviors when routine labels are available. For example, given a routine model of patients who have undergone a surgery to remove cancer and a label that indicates which of the patients in the model has been readmitted, we can automatically detect if future patients will also be rehospitalized based on their routine behavior over time. To do so, we add a label feature to each situation that indicates a person's membership in a sub-population of people defined by their routine or outcomes (*e.g.*, readmitted or not).

Suppose f is a label feature, and $s_{\mathcal{F}}$ is the current situation defined by the full set of features and $s_{\mathcal{F}-f}$ is the current situation defined by a set of features that does not include the label feature. Then the probability of that feature given observation about all other features that we can observe and that define the current situation the person is in $P\left(f \middle| s_{\mathcal{F}-f}\right)$ is given by:

$$P(f|s_{\mathcal{F}-f}) = \frac{P(s_{\mathcal{F}-f}|f) \cdot P(f)}{P(s_{\mathcal{F}-f})} \quad (6)$$

In Equation 6, $P(s_{\mathcal{F}}|f)$ is the probability that the current situation occurs given label *f*, $P(f)$ is the prior probability of the label, and $P(s_{\mathcal{F}})$ is the probability of the situation regardless of the label. We can then use Equation 6 to update the probability of the label feature for each situation in the behavior instance in order starting with the prior that expresses our initial belief about the probability of the feature.

To compute the probabilities in Equation 6, we use our computational model which estimates the expected counts for each situation in our model. For each situation in a behavior instance we use the total expected count to compute the expected counts for each label feature value. We then use the expected counts to compute the respective probabilities in Equation 6. At each time step in the behavior instance we compute the probability of the label feature (*e.g.*, outcome) using the previous time tick evaluation as a prior. We then use the probability of label feature value give current situation observation ($P(f|s_{\mathcal{F}-f})$) using the following classifier:

$$h(s_{\mathcal{F}-f}) = I(P(f|s_{\mathcal{F}-f}) > 1/|\boldsymbol{f}|) \quad (7)$$

The classifier above uses an indicator function *I* to classify if feature *f* is true for any observed features of situation ($s_{\mathcal{F}-f}$) in a behavior instance, if the probability of the situation in the behavior instance ($P(f|s_{\mathcal{F}-f})$) is greater than the probability of other possible label values ($1/|\boldsymbol{f}|$).

7.2 Detecting Behavior Instances

Here we show how to automatically detect behavior instances that are characteristic of a routine. This often involves considering two competing routine and finding behavior instances that are characteristic of one routine, but not the other. For example, to detect behavior instances that negatively impact people, we need to show that they are variations

of routines that negatively impact people. To show that a behavior instance could have a positive impact, we need to show that it is a variation of a good routine.

Because we weakly label behavior instances per-person based on their routines, and not per-instance, we must ensure that only variations of a given routine model are detected and not variations of another opposite routine. This is similar to existing weak labeling approaches (*e.g.*, (Mann & Mccallum, 2010)), except that it requires no prior labels for any behavior instances. We estimate the probabilities of people being in different situations and the conditional probability of actions they perform in those situations (even for situation and action pairs not present in the training data).

To classify a behavior instance, we need to calculate the probability that it belongs to a particular routine. Let b be a behavior instance and let M be a routine model. The probability that behavior instance b belongs to routine model M is given by $P(M \mid b)$. Also, we say that behavior instance b does not belong to routine M (*i.e.*, b is a deviation from the routine) if for some level $0 < \alpha < 1$:

$$P(M \mid b) < \alpha \tag{8}$$

Then, behavior instance b is more likely to belong to routine model M than some other routine model M', if:

$$P(M' \mid b) < P(M \mid b) \tag{9}$$

Given two routine models M and M' (*e.g.*, one that negatively impacts people and the other that impacts them positively), we can say that behavior instance i is in routine M, but not in routine M', if for some level $0 < \alpha < 1$:

$$P(M' \mid b) < \alpha \leq P(M \mid b) \tag{10}$$

Intuitively, Equation 10 means that, if we have evidence that b is a deviation from M', but cannot show evidence that it is also a deviation from M, then b is classified as M. Thus, α represents the probability that a behavior instance is a deviation. Increasing the value of α increases our chance of false positives (classifying b as M, when it is not a variation of M),

while decreasing α increases the chances of false negatives (not classifying b as M, when b is a variation of M). Note that Equation 10 implies that Equation 9 also holds.

To classify the behavior instances requires an indicator function that, given a behavior instance b, results in 1 when the instance is in routine M, and 0 when it is not. We define the following indicator function as our classifier:

$$h(b) = I(P(M' \mid b) < \alpha) \cdot I\big(\alpha \leq P(M \mid b)\big) \tag{11}$$

Note that existing supervised machine learning algorithms (Bishop, 2006) would require per-instance labels to calculate $P(M \mid b)$ for each routine. We instead calculate the probability that behavior instance b belongs to routine M using Bayes rule:

$$P(M \mid b) = \frac{P(b \mid M) \cdot P(M)}{P(b)} \tag{12}$$

where $P(b \mid M)$ is the probability of the instance given that it belongs to the routine M, $P(M)$ is the probability that the routine of the person whose behavior we are classifying is M, and $P(b)$ is the probability that people, regardless of their routine, would perform behavior instance b.

Assuming two models of opposing routines M and M' with probabilities of all possible behavior instances in the model, by law of total probability, Equation 12 becomes:

$$P(M \mid b) = \frac{P(b \mid M) \cdot P(M)}{P(b \mid M) \cdot P(M) + P(b \mid M') \cdot P(M')} \tag{13}$$

We define behavior instance b as a finite, ordered sequence of situations and actions $\{s_1, a_1, s_2, a_2, \ldots, s_n, a_n\}$, where in each situation s_i in the sequence, the person performs action a_i which results in a new situation s_{i+1}. Then, assuming that each situation depends only on the previous situation and action, we calculate the probability of the behavior instance using:

$$P(b|M) = p(s_0) \cdot \prod_i p(a_i|s_i) \cdot p(s_{i+1}|s_i, a_i) \tag{14}$$

where the probability of the initial situation s_0 ($p(s_0)$) and the conditional probability of actions given situations ($p(a_i|s_i)$) are specific to routine model M.

7.3 Generating Behavior Instances

We enable the model to automatically generate behavior instances that are variations of a routine. Traditionally, having an MDP model allows us to find the sequence of situations and actions that maximizes expected reward based on a reward function (Bellman, 1957). In our case, this is the most probable behavior instance starting from a given situation. However, generating only the most probable instances hides the inherent uncertainty and variance in human behavior.

Instead, we sample behavior instances using the probability distributions in a routine model. We start by sampling an initial situation s_0 from the probability distribution of situations ($P(s)$). We then sample the next action from the probability distribution of actions given situation s_0 ($P(a|s_0)$), which gives us an action a_0. We then sample the next situation in the sequence using transition probabilities $P(s|s_0, a_0)$ and get a situation s_1. We repeat this process for situation s_1 and so on until we encounter a stopping situation or reach a maximum behavior instance length.

This procedure allows us to sample from a subset of initial situations constrained on the values of the features of those situations ($P(s|\boldsymbol{f}_S)$). For example, for driving, we could sample situations when a driver is approaching a four-stop intersection. This conditional probability distribution can easily be computed from situation frequencies (D_s) from a routine model. This allows us to generate behavior instances characteristic of a routine for specific situations. We show how to simulate behaviors from a model in one of the next two use cases.

7.4 Use Case: Predicting Cancer Patient Rehospitalization

Currently, after patients are discharged from the hospital following a surgery to remove gastrointestinal cancer, they see clinicians only for infrequent follow ups or when their

symptoms worsen significantly, following which they are rehospitalized. This leaves clinicians with little insight about patients' behaviors between when they are discharged from the hospital after surgery and when they are readmitted to the hospital. Here, we illustrate behavior classification and prediction on a real-world example in the domain of health informatics, and how it could help clinicians predict behaviors of patients with gastrointestinal cancer that lead to rehospitalization after surgery. Identifying differences in routines of cancer patients who have been and have not been rehospitalized could inform interventions that help improve patients' health and wellbeing.

Current commodity mobile and wearable devices, such as smartphones and fitness trackers, offer continuous sensing of patients' behaviors even when they are not in the clinicians' office. Such devices are more affordable than medical grade sensing technologies and are already available to a large number of adults in the United States. This kind of tracking would allow clinicians to study routine behaviors of patients that could help identify new automated interventions and treatments. However, enrolling this population of patients in such data collection studies is challenging because of their general poor health and because they are few compared to the general population. Also, collecting objective and subjective behavior trace data from this population is challenging both due to recruiting and compliance issues. As such, behavior instances resulting from such data collection field studies often have limited number of examples. We address this challenge by modeling routines of cancer patients using our probabilistic model of routine behavior that can generalize from training examples to estimate probabilities of related, but unseen behavior transitions.

7.4.1 Cancer Patient Behavior Dataset and Model

Here we describe our steps to build a computational model of cancer patient routines to answer the following research question: can mobility among cancer patients who have undergone a surgery to remove gastrointestinal cancer reduce the risk of rehospitalization? This research question is driven by recent findings that lack of mobility is correlated with rehospitalization (Low *et al.*, 2018). Our goal was to collect data and build a model to explore this correlation in more detail by looking at behavior instances of patients and routine variations that are characteristic of patients who have been rehospitalized and try

to find evidence that mobility reduces risk of rehospitalization—one of the viable interventions for this patient population.

7.4.1.1 Participants

We recruited 60 patients (28 female and 32 male; median age 66, min age 39, max age 81), who were scheduled for a surgery to remove gastrointestinal cancer, in our data collection study. Fifty-one participants completed the study. Four participants were removed from the study because they were ineligible due to their condition (*e.g.*, did not end up in surgery), four participants withdrew from the study (*e.g.*, because of worsening health conditions, time commitment), and one participant passed away before completing the study. Of the participants who have completed the study, 22 were female and 29 male, all having ages between 39 and 81 (median=64).

All participants started data collection no more than a month and no less than a week before their scheduled surgery. All participants that have completed the study stayed in it for at least 30 days from the day they were discharged from the hospital following their surgery. At the end of the study, 14 participants were rehospitalized (27.45%) within 30 days of being discharged from hospital following their surgery.

7.4.1.2 Data Collection

We gave each participant an Android smartphone device and a Fitbit activity tracker. Participants who have had a compatible Android smartphone device were allowed to use their own device. We collected data using the Aware framework (Ferreira *et al.*, 2015). Collected data included GPS location traces, step counts, sleep, and subjective symptoms that participants entered once a day using an Experience Sampling Method (ESM). Participants received an ESM survey every day that asked them to rate their symptoms (pain, fatigue, concentration, sadness, anxiety, shortness of breath, numbness, nausea, and diarrhea) on a scale from 0 to 10 (where 0 is no symptoms and 10 is the worse symptom they have ever experienced). We have excluded all other sensors from the phones (*e.g.*, screen usage, application usage) because most patients did not use the smartphone as their primary mobile device. We have manually recorded the dates for each state in the surgery process (pre-surgery, surgery, recovery, discharge, and rehospitalized).

7.4.1.3 Data Pre-processing

We extracted features from a subset of raw sensor data streams and aggregated them for each day. Table 5 and Table 6 summarize situation and action features we used to train our model.

Our decision to perform our analysis on a day-by-day basis was guided by having ESM data collected daily. We thus picked a time tick to be a day. We then modeled our situations and actions for each day the participant spent in the study. We first differentiated the current stage of the study the patient is in. We identify 6 distinct stages that drive the world dynamics: pre-surgery, in surgery, recovery, discharged, not admitted following the surgery, and admitted following the surgery. Although our focus was on patients' behaviors following a discharge from the hospital, we included earlier stages as well to account for any symptoms or behaviors that lead up to later discharge stage. Our goal was to identify behaviors that lead to patients transitioning from not admitted to admitted stage.

Our primary focus was on patients' mobility, which was inspired by prior work (Low *et al.*, 2018) that established a correlation between patient activity (or lack thereof) and rehospitalization. We hypothesized that patients who stay at home most of the time will also remain stationary or in bed most of the time and lack physical activity. Following Low *et al.* (2018), we explore physical activity (step counts) as a predictor rehospitalization, but also consider mobility outside of patients' homes and the time they spend in bed.

We extracted significant locations (locations at which people spend time) from GPS position traces. When then identified the patient's home and hospital locations based on time and date and their stay patterns. We calculated the percentage of time they spend away from home, they spend performing physical activity, and they spend in bed for each day. We extracted physical activity and sleep features from a Fitbit fitness tracker. We used minute-by-minute tracking to identify active bouts, defined as at least 10 steps in a given minute) and differentiate them from stationary behavior (less than 10 steps in a given minute). We then use the active bouts total time to calculate the percentage of time spent performing physical activity. Similarly, we calculated the time patients spent in bed.

We then discretized the percentages in three bins ranging from "low" to "high". We also included a special bin for days when no data was collected (*e.g.*, due to data collection failure, due to empty smartphone battery). The bin boundaries for time spent away from home were no time away from home for "low", up to and including 10% for "medium", and more than 10% of the time for "high". The physical activity bin boundaries approximately corresponded to 10 to 720 steps for Low, 721 to 1,440 for Medium, and 1,441 and more for High. Note that the average American takes between 5,000 and 7,000 steps a day with 10,000 steps a day being the recommended target for individuals to gain health benefits from their activity. However, we picked these bins from the data because our population of patients are unable to perform physical activity at the level of healthy individuals. For sleep, we set the boundary to up to 5 hours in bed per day for "low", up to 10 hours for "medium", and more than 10 hours for "high" percentage of time spent in bed.

We also added additional features to account for any confounding variables (*i.e.*, variables that could also have an effect on rehospitalization). We included patient age and gender to account for effects of demographics on rehospitalization. For example, we hypothesized that older patients may be more likely to be rehospitalized due to their age. We discretized patient age into 3 bins (younger than 60, 60 to 69, and 70 and older). We chose the bin boundaries by selecting 5 years around the median age on both sides. We used binary gender at birth to account for any medical differences between the two sexes. We also included two main symptoms historically associated with rehospitalization ((Low *et al*., 2018)): pain and nausea. We left all the other self-reported symptoms out of the model on this iteration for simplicity.

We also captured different stages the patients were in. Those included pre-operation days, day in the operating room (OR), post operation recovery days in hospital, discharge day, and post-discharge stage split into not-admitted and readmitted stages. In addition to these stage features, we also included features that indicate if the patient is currently in hospital, whether they requested home health care assistance, and if they had contact with a clinician (*e.g.*, form personal care physician, surgical oncology clinicians, or have visited the emergency department (ED)).

Table 5. Situation features capturing the different contexts the patients can be in.

Feature	Description
Outcome label	
Readmitted	Whether this patient was readmitted or not. {TRUE, FALSE}
Demographics	
Gender	Gender of patient at birth {FEMALE, MALE}
Age	Age of the patient at surgery {younger than 60, 60-69, 70 and older}
Stages	
Stage	Current stage in the study the patient is in {PRE-SURGERY, SURGERY, RECOVERY, DISCHARGE, NOT AMIDTTED, READMITTED}
In Hospital	Current hospitalization state {TRUE, FALSE}
Home Care Requested	If home care was requested at discharge {TRUE, FALSE}
Clinician Visit	If patient had a clinician visit or not {TRUE, FALSE}
Subjective Symptoms	
Pain Level	Dominant pain level for the day {NOT RECORDED, VERY BAD, BAD, GOOD}
Nausea Level	Dominant nausea level for the day {NOT RECORDED, VERY BAD, BAD, GOOD}
Mobility	
Yesterday's Away From Home	Patient's macro-level mobility the previous day {NOT RECORDED, LOW, MEDIUM, HIGH}
Yesterday's Activity Bouts	Patient's micro-level mobility (physical activity) the previous day {NOT RECORDED, LOW, MEDIUM, HIGH }
Yesterday's In Bed	Patient's time spent in bed the previous day {NOT RECORDED, LOW, MEDIUM, HIGH }

Table 6. Action features representing patient's mobility.

Feature	Description
Survey	Patient's response to the symptoms survey {NOT RECORDED, RECORDED}
Away From Home	Patient's current macro-level mobility {NOT RECORDED, LOW, MEDIUM, HIGH}
Activity Bouts	Patient's current micro-level mobility (physical activity) {NOT RECORDED, LOW, MEDIUM, HIGH}
In Bed	Patient's time spent in bed {NOT RECORDED, LOW, MEDIUM, HIGH}

We converted each patient's data into one training behavior instance per participant, starting from the day they joined the study to 30 days after they have been discharged from the hospital. We labeled each behavior instance based on if the patient has been rehospitalized or not. We then use training behavior instances to train a model. We identified 1,280 different situations and 117 different actions in the behavior log. The final model resulted in 69,120 possible situations and 128 possible actions, with 84,123,648 possible world dynamic transitions.

7.4.1.4 Model Training

We begin our model training by estimating the world dynamics. Unlike our previous use cases where people's actions influence world dynamics almost exclusively, in case of modeling behaviors of cancer patients, it is the medical procedures and processes that drive the transitions between different stages as the patient moves from surgery to recovery, to being discharged from hospital after surgery. For example, patients who participated in our data collection study joined the study at different times in their pre-surgery stage. Thus, we use the behavior instances data and the frequency of transition between stages to estimate the probability that a transition to the next stage will occur.

The other aspect of transitions that we need to model are patients' symptoms. Such symptoms may be driven by other external factors in the environment in addition to patients' behavior. For example, very bad pain and nausea could be a result of surgery complications that we have no data about. Continued bad symptoms could be the result of patients' current state. Thus, we build our model of world dynamics by estimating the probability of transitions for each symptom in our model based on situation and action data from the previous day (including information about patients' demographics). We estimate the probability distribution of each symptom using the maximum entropy classifier (Bishop, 2006). Similarly, patients' symptoms are the main predictors of rehospitalization. We use the same approach to estimate the probability that a patient will be rehospitalized on the next time tick. We then combine the probability estimations for each individual symptom and rehospitalization predictions to estimate the probability of transitions into new situations.

Once we estimate the world dynamics, we can apply the MaxCausalEnt IRL algorithm (Ziebart, 2010) to estimate the probability distribution of peoples' actions given those world dynamics. Our hypothesis is that participants will choose behaviors, given their abilities, that will allow them to avoid being rehospitalized.

7.4.2 Predicting Rehospitalization

One of the use cases is leveraging the computational model of routines to predict patients that are at risk of rehospitalization before they are rehospitalized when exploring their

behavior instances. Clinicians could do this by loading patients' behavior instances into the data track and using the model to predict rehospitalization. This prediction could also be used to automate interventions and treatments that prevent rehospitalization (*e.g.*, detecting patients at risk and then contacting them to assess their health). Being able to predict rehospitalization before it happens also shows the ability of the model to properly capture world dynamics and how people behave given those dynamics.

To predict rehospitalization, we use a modified Equation 6 to predict the probability of rehospitalization feature *R* is true:

$$P(R = True|s_{\mathcal{F}-R}) = \frac{P(s_{\mathcal{F}-R}|R = True) \cdot P(R = True)}{P(s_{\mathcal{F}-R}|R = True) \cdot P(R = True) + P(s_{\mathcal{F}-R}|R = False) \cdot P(R = False)} \quad (15)$$

We then use probability computed in Equation 15 in a classifier (based on the classifier from Equation 7) that automatically predicts is a patient will be rehospitalized each day:

$$h(s_{\mathcal{F}-R}) = I(P(R = True|s_{\mathcal{F}-R}) > 0.5) \quad (16)$$

7.4.3 Method

We evaluate the ability of the model to predict if a patient will be rehospitalized for each day after they have been discharged. Although our algorithm computes probability of risk of rehospitalization for each day since patients enter the study, we decided to only make predictions using our classifier in Equation 16 for each day after discharge because automated risk assessment is valuable when clinicians cannot closely monitor patients (unlike when they are in hospital) and to compare more directly to clinical readmission predictors that predict at time of discharge.

To estimate the accuracy of the model, we performed 10-fold cross validation. We split patients into 10 folds and then for each fold, trained the model on the remaining 9 folds (*i.e.*, estimated both world dynamics, probability distribution of actions given situations on the training data, and the likelihood of each situation), and predicted rehospitalization of the patients in the remaining fold for each day after they have been discharged until: 1) they have been classified as at risk, 2) they have been rehospitalized (at which point we

already know the label and it is pointless to do further prediction), or 3) they end their participation in the study.

We compare our algorithms with three baselines: the majority rule-based algorithm, and two clinical readmission predictors (HOSPITAL score and LACE index (Robinson & Hudali, 2017)). The majority rule predicts no patient will ever be rehospitalized until they show up at the hospital and it is too late to prevent rehospitalization (current *status quo*). However, this is not a desired approach because it does not allow clinicians to administer treatments that could prevent rehospitalization. Clinical research has developed the HOSPITAL score and LACE index (Robinson & Hudali, 2017) to predict patients at risk of readmission based on patients' medical tests (*e.g.*, hemoglobin levels) and the length of stay at the hospital during recovery. Because gastrointestinal cancer is often aggressive, we hypothesize that the two clinical baselines will estimate that the majority of patients will be rehospitalized (including a lot of false positives). However, this is not a feasible approach because it requires that clinicians check on almost every patient every day, which would waste resources on patients that will not be rehospitalized in the future.

7.4.4 Results

In this section, we compare the results of our algorithm compared to the baselines. Figure 2 shows the accuracy of our model prediction compared to the majority rule-based algorithm and two predictors based on HOSPITAL score and LACE index (Robinson & Hudali, 2017). Our results show that after 30 days, our algorithm was able to correctly perform early detection of rehospitalization for 12 out of the 14 rehospitalized patients, across different folds.

Our algorithm had a lower accuracy than the majority rule-based baseline, but also had a higher accuracy than the clinical baselines (Figure 2). The accuracy of the clinical baselines is low because they overestimate the risk of rehospitalization more than our behavior-based algorithm. Unlike the baselines that only make one prediction at the date of the discharge, our algorithm tracks behavior changes over time. This allows it to detect patients after they have been discharged, but before they have been readmitted, which increases the accuracy

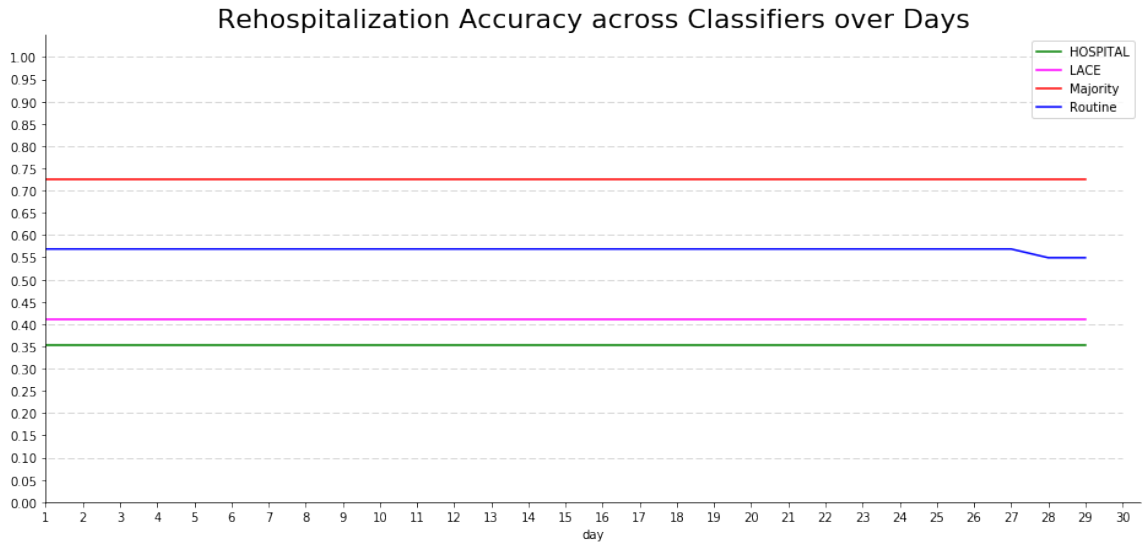

Figure 2. Accuracy of our algorithm compared to two rule-based predictors.

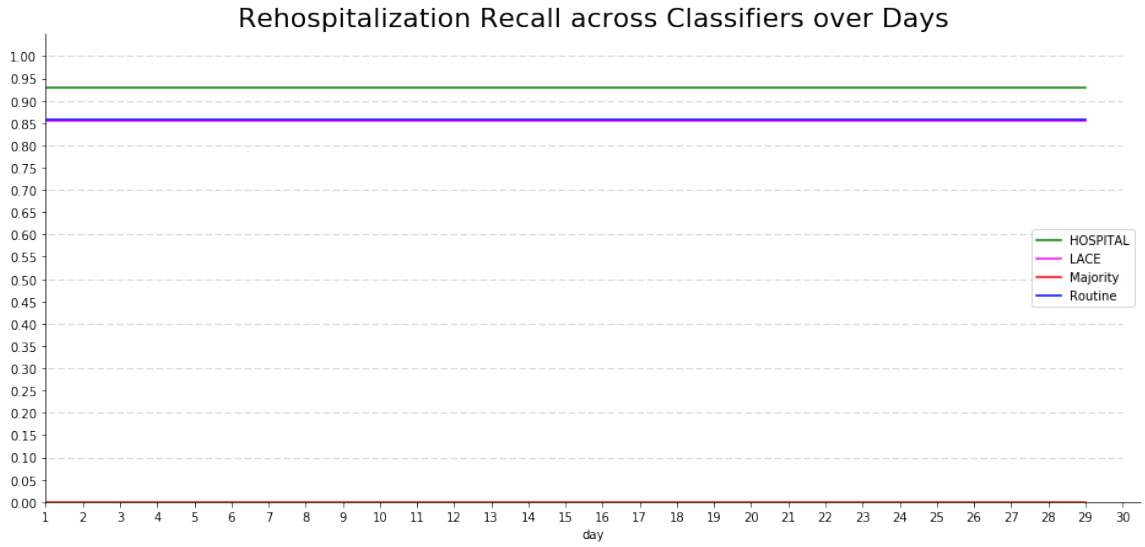

Figure 3. Recall of our algorithm compared to two rule-based predictors.

as more time passes. However, this ability also leads to a decrease in accuracy towards the end of the study once our algorithm starts to predict more false positives.

Despite the high accuracy of the majority rule-based algorithm, it also never predicts that patients will be rehospitalized before they are because it always predicts that no patient will ever be readmitted. This means that it is useless for early rehospitalization detection. Figure 3 illustrates this using the recall measure, or the probability that we can detect a

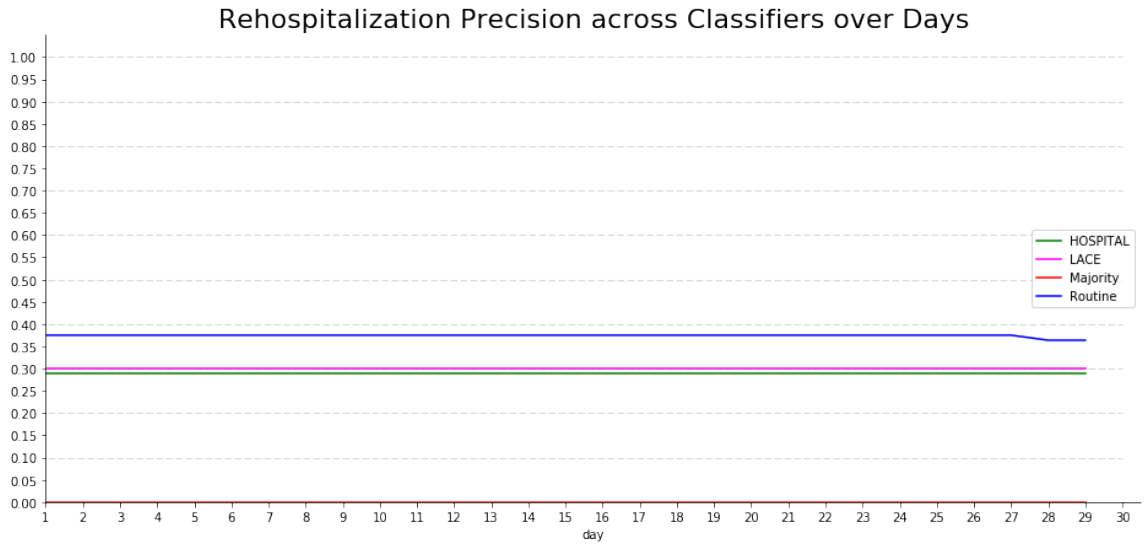

Figure 4. Precision of our algorithm compared to rule-based predictors.

patient that will be rehospitalized, and shows that the majority rule-based algorithm can never perform early detection. It also shows that, on average, our algorithm has the same recall as the LACE-based predictor, but a slightly lower recall than the HOSPITAL-based predictor.

Although the HOSPITAL score and LACE index (Robinson & Hudali, 2017) identified a large number of patients that will be rehospitalized (Figure 3), they will also produce a lot of false positives (*i.e.*, patients that will be detected, but that will not actually be rehospitalized), which would waste hospital resources. Figure 4 shows that our algorithm has higher precision than any other predictor in our experiment. Precision is the probability that a patient we detect will be rehospitalized, and thus higher precision means our algorithm identifies fewer false positives (identifying patients that will not be rehospitalized) than the baselines, which could save clinician resources. The precision of our algorithm dips slightly after day 27, when it adds a false positive to prediction due to the patient's worsening symptoms (Figure 4). However, our error analysis shows that the patient was later rehospitalized after the 30 day milestone, which suggests that the algorithm still correctly detected a patient at risk.

7.4.5 Discussion

Our results show that patients' behaviors prior to being discharged from the hospital are a good predictor of rehospitalization. Our algorithm identified each of the 12 true positives before they were even discharged from the hospital after surgery. This is similar to how existing clinical predictors assess patients at the time of discharge (Robinson & Hudali, 2017). Our algorithm identified as many true positives as the LACE index and one less than the HOSPITAL score, but without any medical test data. Although the HOSPITAL score and LACE index (Robinson & Hudali, 2017) are readily available to patients, our model and prediction algorithm is complementary to the two and we expect that the recall of our approach could be improved by including the HOSPITAL and LACE scores into the model.

The main advantage of our algorithm is that it predicted less false positive than the two clinical baselines (Robinson & Hudali, 2017). Since it predicted less false positives, our algorithm had a higher accuracy than the two clinical baselines. Unlike the majority rule-based algorithm (that predicts no rehospitalizations), which had a higher accuracy than our algorithm, our algorithm is able to predict rehospitalization before it happens. The ability to predict with less false positives reduces the number of potential clinician follow ups with patients who are not actually at risk.

Unlike the existing clinical predictors that only assess risk at the time of discharge, our algorithm assesses patients' behavior over time. For example, our algorithm can start assessing patient risk even before their surgery and help them prepare and improve their behavior prior to their surgery. Similarly, our algorithm offers an opportunity to introduce interventions for at-risk patients even before they are discharged from the hospital; which is earlier than the HOSPITAL score and LACE index are computed (Robinson & Hudali, 2017). Although not as valuable to clinicians as a risk assessment tool at that time because they have medical equipment at their disposal while patients are in the hospital, our algorithm enables automated interventions that can coach patients to improve their behaviors.

Our algorithm also detected an increase in the risk of patients within the first 30 days since they have been discharged. Thus, our approach allows clinicians to monitor patients only when their condition deteriorates according to the algorithm and stop using clinical resources to closely monitor patients when their conditions improve according to the algorithm. Although our data set did not contain any true positives that our classifiers can detect after discharge, the algorithm detected more false positives after day 27 (decrease in precision in Figure 4). Our further error analysis showed that the false positive was a patient who was later rehospitalized after the 30 days mark. Such information would still be valuable to clinicians who continue to treat patients even after the 30 day milestone.

It is difficult to explore which features contribute to prediction in timeseries event data using traditional feature exploration approaches (*e.g.*, feature selection, model feature weight comparison). Instead, we leverage our model as a computational modeling tool that enables tracking changes in probability of feature values and probability of rehospitalization over time. Our model analysis suggested that the probability of readmission significantly increases each day patients spend in the hospital recovering after surgery (features also used to compute HOSPITAL score and LACE index (Robinson & Hudali, 2017)). We also found that high pain and nausea were predictive of rehospitalization. This is likely because such symptoms usually prompt patients to seek clinical help. Later, in Chapter 8, Section 3, we present a detailed discussion about how different features contribute and which ones are predictive of rehospitalization.

7.5 Use Case: Detecting Aggressive Driving Behaviors

The previous section introduced a use case study that showed how to classify behaviors using a supervised method when behavior labels are present (*e.g.*, rehospitalized *vs*. not). However, such labels are not always available, especially when classifying individual behavior instances. Here we show illustrate how to classify behavior instances when no individual behavior instance labels are present.

Helping drivers who routinely engage in aggressive driving behaviors (and thus present a hazard to other people in traffic (AAA, 2009)) understand differences between aggressive

and non-aggressive driving behaviors could improve traffic safety. Technology that can automatically detect aggressive driving instances and coach drivers by simulating what non-aggressive driver would do in the same situation could help drivers improve. For example, a system could try to calm the driver to avoid immediate future aggressive behaviors. It could wait until after the trip is over and show the driver a better, non-aggressive way to drive on the same portion of the trip where the driver drove aggressively. If the driver continues to drive aggressively, the system could suggest that the driver take corrective driving classes with a list of particular instances to work on. However, such technology requires labeling driving behavior instances, which is difficult because drivers may be prone to aggressive driving behavior in some situations, but not in others (*e.g.*, rushing yellow lights during rush hour (Shinar & Compton, 2004)). High-level characteristics of aggressive routines (*e.g.*, speeding) can be detected with lengthy manual observations (Shinar & Compton, 2004)). However, high-level observations may not capture nuances of driving behaviors required to label driving instances to train classification algorithms. Thus, we detect and generate aggressive and non-aggressive driving behavior using weakly labeled data.

We test external validity and show a real-life application of our work in a series of user studies. We show how our work supports technology that helps people identify instances of aggressive and non-aggressive driving through use of animation (Figure 5). We train our driving routine models on a naturalistic driving dataset of 13 aggressive and 13 non-aggressive drivers, labeled based on their driving history (Hong *et al*., 2014). Through our own analysis and with help from driving instructors, we show that our algorithm accurately detects aggressive behaviors and generates meaningful non-aggressive alternatives. In another user study, we show that our approach helps raise drivers' awareness of their own aggressive behaviors. Our algorithm identifies problematic behaviors and suggests ways to improve those behaviors, and thus informs computer-supported coaching technology.

Here we describe how to apply our behavior detection and generation algorithms in the domain of driving routines. We show how stakeholders can leverage the two models of driving routines, and the ability of the model to automatically detect and generate characteristic behaviors. To assist different stakeholders in this task, we built a domain-

specific driving animation tool. This tool is different from the earlier visual analytics tool in that it supports stakeholders who may not understand concepts associated with that tool, such as driving instructors (domain experts) and drivers (end users).

7.5.1 Naturalistic Driving Behavior Dataset and Model

We use the same dataset used in the previous section and originally collected by Hong *et al.* (2014). Earlier we showed that this data can be used to train meaningful driving routine models of how non-aggressive and aggressive drivers drive through intersections. Here, we extend the original data set and the way we modeled driving behaviors earlier. We use the Open Street Map API to mark each intersection in the data with speed limits, intersection types (*e.g.*, t-intersection), and traffic signs and signals. This allows us to detect more nuanced behaviors than in our domain experts study. For example, with this addition we can detect if a driver has properly stopped at a stop sign or not, whereas in the old dataset this was not possible. Lack of information about other vehicles that may impact the driver's behavior remains a limitation.

We divide intersections into four stages (approaching, entering, exiting, and leaving the intersection). Sequences of these stages are the behavior instances in our model. Position information, along with the type of maneuver and details of the vehicle, such as its current speed, make up the situation features (Table 7). Actions represent how the driver operates the vehicle by depressing the gas (throttle) and brake pedals. Because we model driving through an intersection in stages, we aggregate the driver's actions between different intersection stages to represent the changes in throttle and braking levels (Table 8).

We then weakly label instances into aggressive and non-aggressive routines using the same per-person labels from Hong *et al.* (2014), which they assigned based on drivers' self-reported driving violations and their answers to the driver behavior questionnaire from (Reason *et al.*, 1990). We build two models, one for each label, and estimate the probabilities of all possible behavior instances in each model. To model how a vehicle moves in response to driver actions, we empirically estimated the situation-action transitions ($P(s'|s,a)$) from the training data by counting the frequency of transitions between features that describe the vehicle situation. We identified 20,312 different

Table 7. Situation features capturing the different contexts the driver can be in.

Feature	Description
Goals	
Maneuver	The type of maneuver at the intersection {STRAIGHT, RIGHT TURN, LEFT TURN}
Environment	
Position	Current position of the vehicle in the intersection {APPROACHING, ENTERING, EXITING, AFTER}
Rush hour	Whether the trip is during rush hour or not {TRUE, FALSE}
Intersection	Intersection layout including road types in each direction (40 discrete values)
Traffic signs	Traffic signal layout {STOP, STOP OPPOSITE, ALL STOP, LIGHT SIGNAL}
Maximum Speed	The maximum speed in each position of the intersection. {25, 35, 45}
Vehicle	
Speed	Current vehicle speed (5-bin discretized + stopped)
Throttle	Current throttle position (3-bin discretized)
Acceleration	Current positive/negative acceleration (5-bin discretized)

Table 8. Action features representing actions that drivers can perform between stages of the intersection.

Feature	Description
Pedal	Aggregated gas and brake pedal operation between intersection positions (47 discrete values)

situations and 43 different actions in the dataset. The final model had 234,967 different situations, 47 different actions, and 5,371,338 possible transitions.

7.5.2 Classifying Driving Behavior Instances

The model trained on behavior instances of all aggressive drivers in our training data allows us to compute the probability of situations ($P(s \mid Agg)$) and probability of actions given situations ($P(a \mid s, Agg)$). Similarly, the other model, trained on all non-aggressive drivers, allows us to compute $P(s \mid NonAgg)$ and $P(a \mid s, NonAgg)$.

To classify a new behavior instance b as either aggressive or not, we use an indicator function which is 1 when b is a variation of the aggressive routine, and 0 otherwise:

$$h(b) = I(P(NonAgg \mid b) < \alpha) \cdot I\big(\alpha \leq P(Agg \mid b)\big) \tag{17}$$

Similarly, we classify b as either non-aggressive or not using an indicator function which is 1 when the behavior instance b is in the non-aggressive routine, and 0 otherwise:

$$h(b) = I(P(Agg \mid b) < \alpha) \cdot I\big(\alpha \leq P(NonAgg \mid b)\big) \tag{18}$$

Table 9. The mean percentage of behavior instances classified as aggressive, non-aggressive, or neither across different α levels.

	α =0.1			α =0.25			α =0.45			α =0.5		
	Agg	*Neither*	*Nonagg*	*Agg*	*Neither*	*Nonagg*	*Agg*	*Neither*	*Nonagg*	*Agg*	*Neither*	*Nonagg*
aggressive	0%	99.99%	0.01%	2.02%	92.34%	5.64%	41.82%	26.25%	31.92%	56.23%	0%	43.77%
non-aggressive	0%	99.77%	0.23%	1.08%	91.91%	7.01%	38.45%	23.78%	37.76%	50.35%	0%	49.65%

Given the two classifiers and two different α (one for each classifier), we can classify behavior instances as strictly aggressive, strictly non-aggressive, or neither. Later in our validation section, we use different values for α to test the impact of this parameter on our classification. We estimate the prior probability of an aggressive driver $P(Agg) = 0.5$ because the number of behavior instances in the training set is balanced between people with the two driving routines.

7.5.3 Generating Driving Behavior Instances

We start sampling driving behavior instances from our driving routine models by conditioning initial situations (a driver approaching an intersection) on features that describe the environment and driver goals (see Table 7). We sample the initial situation from the conditional probability distribution $P(s|\boldsymbol{f}_S)$, where $\boldsymbol{f}_S$ is a set of situation features values. Note that conditioning the probability of the initial situation on features that include the state of the vehicle (*e.g.*, speed) also allows us to explore how a non-aggressive driver would recover from a particular aggressive situation.

Generating behavior instances for specific driving situations allows us to explore "what-if" scenarios for the two driver populations, even if our training data does not contain those exact scenarios. For example, suppose we detect that a driver aggressively approached a t-intersection with traffic lights. To learn how a non-aggressive driver would behave in that scenario, we sample behaviors from our non-aggressive model starting with an initial situation in the same intersection. We can then use the generated non-aggressive instance to show the driver how to improve.

7.5.4 Preliminary Detection Validation

Validating our algorithms is hard because we train them with no ground truth about which instances are aggressive and which are non-aggressive. We could not compare our algorithms with supervised machine learning algorithms or compute metrics that are helpful for evaluating supervised machine learning algorithms (*e.g.*, accuracy and ROC curves) because both require per-instance labels. Instead, we used leave-one-out validation to avoid training and testing on the same data, and manually checked if a subset of detected/generated instances were variations of the two models.

We used our driving animation tool to manually inspect behavior instances that we detected using our algorithm. For each driver in the dataset, we withheld the data from that driver, and trained the models on the remaining drivers. We then used the driver data we withheld to classify all driving behavior instances from that driver using the two indicator functions (Equations 17 & 18). We sorted classified behavior instances by their frequency in the data and the probability that they belong to one of the routine models, and inspected the most frequent behavior instances.

Our results show that our algorithm on average found more aggressive driving instances among the aggressive drivers and more non-aggressive instances for the non-aggressive drivers across α levels. Table 9 shows mean percentages of classified behavior instances for different α levels.

In a majority of detected aggressive instances, the drivers drove over the speed limit. In the most aggressive instances, drivers were exceeding the speed limit by 20 MPH. In most of these situations, the drivers were going straight through intersections with stops signs for traffic coming from the left or right. The drivers were likely expecting other drivers to respect their signs. However, at high speeds they may not be able to react in time if a vehicle turns in front of them.

The majority of actions in the detected aggressive behavior instances involved drivers pressing hard on gas and brakes, matching summary statistics from Hong *et al.* (2014). Aggressive driving instances also included pressing the pedals softly. Further analysis

showed that the drivers were already in situations that were indicative of aggressive driving when they performed those actions. Although the drivers had made an attempt to correct their behavior, it was already too late.

Drivers in automatically detected non-aggressive behaviors observed the traffic law (*e.g.*, maintained the speed limit). The most likely non-aggressive instances included an easily identifiable pattern where the driver would brake softly when approaching an intersection and then applying gas softly to increase the speed to clear the intersection. Non-aggressive driving instances were equally likely to occur in and out of rush hour. This is in contrast with about 70% of aggressive driving instances that occurred during rush hour.

Our algorithm also detected nuanced differences between the aggressive and non-aggressive instances. For example, it detected a common driving behavior instance where the driver goes through an intersection with traffic lights at a constant speed matching the speed limit of 35 MPH as aggressive with high probability (p_{Agg}=0.6892). A generated non-aggressive behavior for the same scenario shows that non-aggressive drivers are likely to slow down slightly as they approach the intersection and then increase their speed after clearing the intersection (p_{NonAgg}=0.7458). This shows that our algorithm can detect behaviors that may not be obvious, but that are characteristic of a particular routine.

7.5.5 Domain Expert Evaluation of Driving Detection and Generation

Two licensed driving instructors volunteered to evaluate our algorithms and ensure that they accurately detect and generate meaningful driving behaviors. The two instructors (1 male and 1 female) arrived at our lab and signed a consent form. One of them had over 10 years of experience as a driving instructor, and the other had over 30 years of experience as a driver safety instructor and driver license examiner. They were compensated with $35.

The evaluation consisted of two tasks. The instructors first each rated 55 different randomly selected driving behavior instances from our training data set. Instructors rated each instance as: 1) *aggressive*, 2) *non-aggressive*, or 3) *neither*. The instructors then rated another 30 random automatically detected aggressive driving instances and 30 corresponding generated non-aggressive instances. They also rated if each generated instance was a good non-aggressive alternative to the aggressive behavior or not. In both

tasks, the first 5 behavior instances were used as warm up and to explain the tasks to the instructors. We asked the instructors to think aloud in both tasks as they were rating the instances.

We compared probabilities that behavior instances belong to the aggressive routine model (P_{Agg}) between different ratings using the Kruskal-Wallis test. We did a pairwise comparison using Mann-Whitney's U test with Bonferroni correction. We hypothesize that the aggressive probabilities will be highest for behavior instances that the instructors rated as aggressive, followed by those rated as neither and non-aggressive. We use our results to identify a reasonable α for our classifiers to be used in future driving systems.

7.5.5.1 Results

As expected, we found differences in P_{Agg} between different instructor ratings ($\chi^2(2)=6.73$, $p=.0346$). The median P_{Agg}=62.40% of instances rated as *aggressive* was higher than P_{Agg} of behavior instances rated as *non-aggressive* (median=48.14%; $p=.0360$, r=.31). The difference between P_{Agg} of instances rated *aggressive* and instances rated as *neither* (median=50.69%) was only marginally statistically significant ($p=.0940$, r=.34). Our tests did not find a statistically significant difference between P_{Agg} of instances rated as *non-aggressive* and *neither* ($p>.9999$, r=.02).

Over 75% of instances rated as *aggressive* had P_{Agg} greater than 50.32%. Over 75% of instances rated as *neither* and *non-aggressive* had P_{Agg} lower than 59.75% and 57.28% respectively. To detect as many aggressive instances, while limiting false positives, we set aggressive classifier α_{Agg} to 1-0.55=0.45. Because more than 75% of instances rated as *neither* had P_{Agg} greater than 38.98%, we set a more cautious non-aggressive instance classifier α_{NonAgg} to 0.35.

The instructors rated 72.5% of automatically detected aggressive instances as *aggressive*, 5% as *non-aggressive*, and 22.5% as *neither* in the second task. They rated 85% of the corresponding automatically generated non-aggressive instances as *non-aggressive*, 5% as *aggressive*, and 10% as *neither*. They confirmed that all generated instances that they rated

as *non-aggressive* were appropriate alternatives for the automatically detected aggressive instances.

7.5.5.2 Discussion

Our classifier detected aggressive driving instances that match known characteristics of aggressive driving behavior (*e.g.*, speeding (Shinar & Compton, 2004)), hard acceleration and braking (Hong *et al.*, 2014)). The two driving instructors also confirmed that in most cases our algorithms detected and generated non-aggressive driving behavior instances that are safe enough to suggest to drivers as alternatives to aggressive driving behaviors. However, in some cases, the instructors could not properly rate the instances due to the lack of information about the environment (*e.g.*, other vehicles in traffic, pedestrians).

The lack of such information made it difficult for the instructors to rate if a behavior was *aggressive* or *non-aggressive*, and they had to mark such instances as *neither*. Lacking this information, they had to assume that the driver is responding to the environment in a reasonable way. For example, when the animation shows the vehicle going straight through an intersection with traffic lights, they assumed the light was green; when the animation showed the vehicle stop in the middle of the intersection before proceeding they had to assume that this was because another vehicle was blocking the driver's way.

Our validation also yielded some surprising findings. For example, we found that drivers labeled as non-aggressive may still frequently engage in aggressive behaviors. Thus, high-level classification techniques that target aggressive drivers (*e.g.*, driving assessment questionnaires (Reason *et al.*, 1990)) may miss important behaviors that could help other people improve their driving, too. This also means that sampling from the non-aggressive driving model may in some cases result in an aggressive behavior instance. To ensure that our system never suggests aggressive driving behavior to the user, we only present generated instances that we classify as characteristic of a non-aggressive driving routine.

We reviewed instances where the instructors disagreed with our automatic classification, and found that our classifiers are more sensitive to individual driver actions. For example, the instructors did not rate instances in which the driver drives straight through an

intersection at the speed limit as aggressive. However, our preliminary evaluation showed that such behavior has a better non-aggressive alternative.

This shows that our detection algorithm is able to point to new information about what is characteristic of aggressive driving. This does not mean that all driving instances we detect as characteristic of aggressive routine are necessarily endangering the driver and others in traffic. It means that drivers that do exhibit such behaviors may be at higher risk of causing intentional traffic violations because they routinely engage in aggressive driving behaviors.

7.6 Summary

In this chapter, we showed that our model can be used to automatically detect and generate salient patterns of behavior from large behavior logs. This is an extension to the existing MaxCausalEnt IRL (Ziebart, 2010) algorithm that has been used to predict behaviors. The results of our evaluation show that our algorithm can be used to classify quality of routines at the behavior instance level as opposed to coarse classification at the level of the person (*e.g.*, classifying that a driver is aggressive vs. that an instance of driving through an intersection is aggressive). This is important because it gives us an ability to identify poor behaviors even among population of people whose overall routines may hide such behaviors (*e.g.*, drivers who only sometimes engage in aggressive behaviors). In later chapters, we will illustrate the value of automatically detecting and generating behaviors as evidence to aid stakeholders in building a conceptual model of differences between routines in the sensemaking process.

8 ROUTINE MODELS AS SENSEMAKING TOOLS

Once stakeholders identify salient patters of behaviors that make up routine variations and deviations in a computational model of routine behaviors, they can move on to exploring and understanding the routine captured in the model. We show how such routine models help in two important stages of the sensemaking loop: 1) to identify salient behavior instances characteristic of a routine in the foraging loop, and 2) to generate and test hypotheses about routines in the sensemaking loop. We illustrate this on two different use cases: 1) in a driving domain in which we developed a specialized animation tool that helps

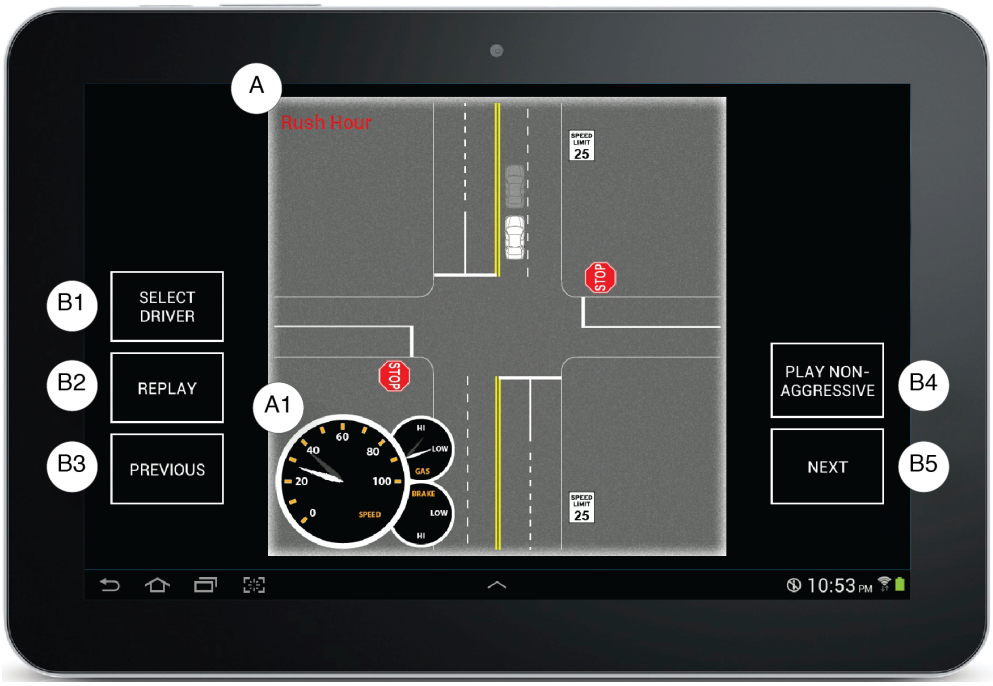

Figure 5. Driving behavior detection and generation tool user interface. The main animation region (A) shows a scenario in which a vehicle travels straight through an intersection. The simplified intersection consists of a main road with 25 miles per hour speed limit and a residential street with stop signs for opposing traffic. The current scenario depicts an automatically *detected* aggressive driving behavior (speeding transparent vehicle) and an automatically *generated* non-aggressive behavior (opaque vehicle, which drives within the posted speed limit). Dials (A1) show the vehicles' speed, and gas and brake pedal positions. The user can: B1) select a driver and a trip to review, B2) replay current driver behavior, B3 & B5) load previous and next driving behavior in the current trip, and B4) play or replay an automatically generated non-aggressive behavior for the given driving scenario.

driving instructors and drivers to automatically identify and contracts behaviors that are characteristic of aggressive and non-aggressive driving routine, and 2) in a health informatics domain in which we developed a specialized visual analytics tool that utilizes a model of patient routines to allow clinicians to verify their hypotheses about which behaviors lead to rehospitalization of patients who have undergone a surgery to remove cancer.

8.1 Identifying Salient Patterns of Routine Behavior

One of the main steps in building a conceptual model of a routine is to identify behavior instances that are characteristic of that particular routine and to contrast it against another competing routine. For example, to understand what makes a driver aggressive requires identifying different variations on the driving routine that are more likely to be exhibited by aggressive drivers than non-aggressive drivers. Automatically detecting salient patterns of behavior helps identify reduce the cost of searching for such behavior instances in the behavior logs. However, automating this step is often challenging because behavior instances are not labeled based on the routine they are characteristic off. In this section, we show the ability of our model, trained on weakly labeled behavior instances, to automatically characterize routine behaviors and help various stakeholders to conceptualize the differences between competing routines.

8.2 Use Case: Differentiating Aggressive Driving Behaviors

Helping drivers understand the difference between aggressive and non-aggressive driving behaviors and identifying specific instances when they are behaving aggressively and showing them how they could improve, could help them become safer in traffic. However, it is often difficult for drivers to identify these kinds of behaviors on their own. Here we show how to leverage our computational model of driving routines to help drivers automatically detect when they are driving aggressively and coaching them by simulating what non-aggressive drivers would do in the same situation.

8.2.1 Driving Behaviors Animation System

We implemented our driving behavior animation system for Android touchscreen tablets. The client mobile application is powered by a server-side routine modeling service. The client downloads driving behavior instances from the server and plays them to the user.

The client user interface (Figure 5) features an animation area (Figure 5.A), which depicts a vehicle in an intersection reminiscent of illustrations found in driving study books. Each intersection depicts a situation with an intersection type (four-way intersection or t-intersection), intersecting road types (main roads or residential roads), and traffic signs (speed limits, stop signs, and traffic lights). Our data does not contain information about traffic light's current light color and instead only shows that the intersection is controlled by traffic lights. The roads and the vehicle depict average road and sedan vehicle sizes in North America.

Vehicle animation shows how an actual vehicle may have moved through an intersection. We implemented a simple 2D physics engine to compute the speed and acceleration of the vehicle as it drives through different intersection positions. The maneuver feature guides the trajectory of the vehicle. The action pedal feature modifies speed in between two consecutive positions in the intersection to illustrate how drivers' actions affect the movement of the vehicle.

The user can review trips from a particular driver (Figure 5.B1), and load all intersections from a trip or a subset of intersections where the driver drove either aggressively or non-aggressively. The user can then replay the current driver's behavior instance (Figure 5.B2), or switch between previous and next behavior instances in the subset (Figure 5.B3 and B5). For any instances, the user can generate and animate how a non-aggressive driver would drive through the same intersection (Figure 5.B4).

8.2.2 Driving Animation Evaluation

Our tool shows simplified driving maneuvers in abstracted intersections. To ensure that general users can interpret the animations correctly, we conducted a pilot study with 12 participants (6 male and 6 female), who were ages between 19 and 35 (median=21), and

had between 0 and 13 years of driving experience (median=3). They were compensated $10. We lost two participants' data due to technical issues.

Participants arrived at our lab and signed a consent form before we briefed them about the study. They reviewed 25 randomly selected driving behaviors from our training set, and compared them with randomly generated behaviors from our model. For each actual and generated instance, we asked them to write a paragraph describing what the driver did in the intersection, and another paragraph about what the driver did differently between the two. The first five scenarios were warm ups to ensure the task was understood.

Two researchers independently coded participant responses and rated them as: 1) incorrect, if the answer did not match the driving behavior in the scenario; 2) partially correct, if the answer had most, but not all relevant information about the scenario; and 3) correct; if the answer had relevant information about the scenario and the driving behavior without mistakes. Each researcher rated 600 descriptions and they perfectly agreed on 83.86% of them (Cohen's kappa=0.54). For each rating, we computed the average score rounded down towards the lower of the two scores.

The participants accurately described the scenarios. They correctly described 85% of driving instances, partially correctly described 13%, and incorrectly described 2% of the instances. They correctly compared 79% of instances, partially compared 15.5%, and incorrectly compared 5.5% driving instances. To increase the users' accuracy and to reduce the time and effort to compare behaviors, we modified our application interface to show a ghost vehicle (Figure 5) when the user (re)played generated behaviors.

8.2.3 Acting in Response to People's Behaviors

We now illustrate how we leverage driving routine models to raise people's awareness and help them understand their driving behaviors. We could generate behaviors from both aggressive and non-aggressive models to help drivers understand how these routines differ in frequent driving situations. However, our approach is uniquely positioned to detect actual drivers' aggressive behaviors to help them reflect on differences between their behavior and generated non-aggressive driving behavior in the same situations.

We conducted a user study in which participants drove on a predefined route and reviewed their driving behaviors using our tool. We hypothesize that showing participants their aggressive behaviors and non-aggressive alternatives will change their rating of their driving expertise and quality. Out tool is not meant to motivate or lead to behavior change—we assume that the drivers are already motivated to improve (*e.g.*, to pass a court ordered driving test).

8.2.3.1 Method

We recruited 20 participants (12 male, 8 female), ages between 19 and 72 (median=25). Participants had valid driver's licenses, and had between 1 and 55 years of driving experience (median=7). They arrived at our lab and signed a consent form. We briefed them on the study, and then installed an OBD2 sensor and a smartphone (to collect data from OBD2) into their vehicles. Participants then drove on a predefined route in a mid-sized city in North America. The route reflected different kinds of driving situations in 24 intersections. Participants drove five laps (total 120 intersections) to ensure that they had the time to fall into their driving routine. They drove alone to minimize the effect of a user study on their normal driving behaviors.

After returning to our lab, we uploaded their driving data into our modeling system. The upload procedure converted OBD2 data into behavior instances, and automatically detected aggressive driving instances using our model now trained on all aggressive drivers from our extended data set.

Participants then completed a pre-test questionnaire, in which they rated their driving expertise and quality on 6-point Likert scales, where expertise ranged from very inexperienced to very experienced, and quality ranged from very aggressive to very non-aggressive. We also modified the standard driver behavior questionnaire (DBQ) (Reason *et al.*, 1990) to include only 15 violation and accidental violation questions relevant to our study. Participants answered how frequently they engaged in driving behaviors in each of 15 questions, and how that makes them feel about their driving expertise and quality on the same 6-point Likert scales.

We then randomly split participants into two groups: 1) *baseline*, in which participants used our modified driving animation tool to review *all* of their driving instances, and 2) *tool*, in which participants reviewed their automatically detected aggressive driving instances, and compared them with corresponding automatically generated non-aggressive driving instances. We ensured that both conditions had the same proportion of males (n=6) and females (n=4). We explained to participants how to use the tool and told them that they can review the behaviors for as long as they want.

After they finished using the tool, participants answered the same questionnaire again. We compared their pre- and post-test answers to understand the effects of our tool on their awareness and understanding of their aggressive driving behaviors. Afterwards, participants commented on the tool. The study lasted about two hours and participants got $35.

8.2.3.2 Measures

We measured the change between participants' answers in pre- and post-tests. We computed the change in overall driving expertise (Δ_E) and quality (Δ_Q), median change in frequency of behaviors in our modified DBQ (Δ_{DBQ}), and median change in expertise (Δ_e) and quality (Δ_q) over all DBQ questions. We compared the differences between the changes in the two groups using Mann-Whitney's U test.

8.2.3.3 Results

Most participants in the two conditions did not change their answers about overall driving expertise (both conditions median Δ_E=0) and quality (both conditions median Δ_Q=0). Our tests could not find a significant difference between the two conditions for Δ_E (U=47, Z=0.26, p=.9288, r=.06) and Δ_Q (U=46, Z=0.34, p=.8727, r=.08). Thus, participants may be unlikely to change their overall driving expertise and quality rating after reviewing only a few of their behaviors.

However, they did change their perception of specific driving behaviors in our modified DBQ. Participants in the *baseline* condition reduced their frequency of intentional and accidental aggressive violations (median Δ_{DBQ}=-1.0) compared to participants in the *tool*

condition who slightly increased the frequency (median Δ_{DBQ}=0, 75th percentile Δ_{DBQ}=1; U=23, Z=2.1808, p=.0259, r=.49).

Also, as we hypothesized, participants in the *tool* condition changed their answers about their driving quality towards aggressive behaviors (median Δ_q=-1.0), while participants in the *baseline* condition changed towards non-aggressive (median Δ_q=1.0; U=82, Z=2.65, p=.0128, r=.59). Participants in the *baseline* condition changed their answers about their expertise towards the inexperienced end of the scale (median Δ_e=1.0), but the *tool* condition participants mostly did not (median Δ_e=0). Our test could not find a significant difference (U=49, Z=0.08, p=.9682, r=.02).

The only difference between the two conditions was the participants' exposure to automatically detected aggressive and generated non-aggressive instances. The percentage of detected aggressive instances was similar across the two conditions. Thus, we conclude that exposure to the detected and generated behaviors raised participants' awareness about their specific driving behaviors.

Lack of information about other vehicles and pedestrians affected participants' ability to understand aggressive instances. For example, in response to slowing down for a vehicle in front, *P11 (tool)* said: *"I did not agree that the scenarios flagged as aggressive were aggressive, especially the ones where I was moving significantly more slowly than the 'non-aggressive' scenario."* However, slowly tailing a vehicle in an intersection could result in the driver's vehicle blocking the intersection if the lane in front fills up. The generated behavior may account for the gap that the non-aggressive drivers keep between vehicles in such situations.

Participants also at times disagreed when our detected instance challenged their notions of aggressive driving. For example, *P6 (tool)* commented: *"Most of the time the tool suggested I drive slower when I was already driving under the actual legal speed limit, which was weird."* However, our non-aggressive driving routine model favors softly pressing on the gas and lower acceleration, which resulted in a slower speed than the participant's in those situations.

In other cases, generated non-aggressive instances helped participants understand why the system made its decisions. For example, *P16 (tool)* said: *"I didn't think that going through an intersection at the speed limit is aggressive, but then I saw the non-aggressive option that showed I should slow down a bit when approaching the intersection and then speed up after I passed it. And I thought: 'Oh, that does make sense!'"* Although participants may disagree that a particular instance of their behavior is aggressive, they learned of a better way to drive through the intersection.

Drivers are not fully aggressive or fully non-aggressive, but show aggressive and non-aggressive behaviors in different situations. Drivers using the tool had both an increase in their self-assessed aggressive driving quality and frequency. Our tool raised awareness of more aggressive driving behaviors than status quo (no coaching in pre-test) and a baseline condition (unguided review). Our tool can coach drivers to identify aggressive driving behaviors and contrast them with non-aggressive behaviors.

8.2.3.4 Discussion

We automatically detect and generate behaviors using probabilities that people with a certain routine will perform sequences of actions in a given situation. We calculate the probabilities in a principled way that ensures that the model best fit training data from large behavior logs. Without our contribution, we would have to perform tedious manual exploration of the existing models to understand specific differences between variations of different routines.

Our work is based on a real-life example where the lack of labeled behavior instances impedes technology that could help people improve their lives. The challenge of labeling individual behavior instances is not limited to driving and spans number of other domains. For example, our approach can simplify labeling daily routines of parents who are likely to forget to pick up their children. Here, the weak, per-parent labels indicate if they have forgotten their children in the past or not. Our approach offers a generalizable solution to classify and generate behavior instances in different domains, because it is based on probability axioms and a proven model of routine behavior.

We showed the ability of our algorithms to automatically detect and generate driving behavior instances using weakly labeled data. Our approach can detect behavior instances that can negatively impact people and those that can have a positive impact on them. An important by-product of our approach is that it can also be used to detect behavior instances that are not characteristic of any particular routine (*e.g.*, behavior instances that are frequently exhibited by both aggressive and non-aggressive drivers). We refer to such behavior instances as *aproblematic*, and hypothesize that behavior change technologies could use such instances as a stepping stone towards better behavior.

We found that people who have demonstrated a certain routine do not always behave in ways that are characteristic of that routine. We found that aggressive drivers often drive in a way that is not necessarily aggressive. Also, some drivers who generally drive non-aggressively will at times exhibit aggressive behaviors. The existing screening techniques (*e.g.*, driving assessment questionnaires (Reason, Manstead, Stradling, Baxter, & Campbell, 2011)) would miss an opportunity to help such drivers improve.

Our use case illustrates how automatically detecting and generating driving behaviors can help driving instructors evaluate other peoples' driving behaviors and help drivers reflect on and understand their own driving behaviors. However, we have also identified a common theme during our validation that both kinds of users are quick to trust obvious behaviors that may match their preconceived notions about aggressive driving. On the other hand, they were slow to accept other more nuanced behaviors that may be characteristic of a particular routine. Although validating their preconceived notions confirms our algorithms are correct in most cases, we would ideally like to provide a tool that helps generate new knowledge.

The users need to be able to trust the tool before they can accept new knowledge that it suggests. To increase user trust, our tool should include complete information about the environment and show that it observes the environment in a correct way. Showing that the algorithm detects behaviors that matter could also increase trust. For example, the tool could animate what could go wrong when the driver engages in less obvious aggressive behaviors (*e.g.*, show that drivers that slowly tail other vehicles can get stuck in

intersections). Building trust in new knowledge that the tool generates is fundamental to behavior change, which is hard to do if we cannot persuade people that their behavior could negatively impact them.

8.2.4 Summary

In this section, we showed that visualizing automatically detected classes of behaviors and contrasting them with automatically simulated behaviors from another competing routine through animation can help stakeholders understand the differences between routines. Our user study showed an example of how such exploration of data helps stakeholders to reflect on and build better conceptual models of those behaviors. This kind of exploration through classification and simulation is the key contribution of our computational modeling methodology. Our study has shown that without automated detection it is difficult for stakeholders to identify patterns that are characteristic of a behavior. However, evaluation of our early prototypes showed that even the ability to automatically detect behaviors may not be enough to explain the behaviors. It is only after we implemented a feature that contrasted the aggressive and non-aggressive alternatives that participants managed to clearly ascertain what makes a certain behavior characteristic of one routine and not the other.

8.3 Generating and Validating Hypotheses about Routines

Computational models enable stakeholders to explore a complex system by simulating what would happen in the system in response to changing conditions. In the domain of behavior modeling, simulation allows stakeholders to "play out" hypothetical behavior instances based on their conceptual model of behavior and verify that their hypothesis matches the simulation. This in turn enriches stakeholders' conceptual model of a routine and allows them to reason about and act in response to behaviors they have studied. For example, a clinician studying behaviors of patients who have undergone a surgery to remove a cancer may hypothesize that lack of mobility is correlated with high rates of rehospitalization of patients within the first 30 days from their surgery. Being able to explore what happens to various groups of patients after they have been discharged from

the hospital could help clinicians confirm their hypothesis and in turn inform the design of information technology-based interventions that try to motivate sedentary patients to move more and avoid negative outcomes of their treatment.

However, it is difficult to explore large computational models of routines, which often contain millions of possible transitions between situations and trillions of possible behavior instances, without a visualization tool that allows the stakeholders to explore the model. Although existing MDP visualization tools can visualize behavior instances resulting from an MDP model, they focus on visualizing deterministic policies and not stochastic policies, which are prevalent in behavior modeling. One way to visualize and explore such stochastic policies is through simulation. For example, in the previous chapter, we illustrated the value of behavior simulation by automatically generating non-aggressive driving behaviors to contrast them with aggressive behaviors and help driving instructors and drivers explore and understand the differences between the two. However, our illustration is limited in that it does not allow the user to specify which aspects of behavior to explore. Instead, exploration is driven by automatically detected poor routines. However, stakeholders may want to freely explore other aspects of behavior. Another challenge of our previous illustration is that it is domain specific. This means that for each domain we study, we would have to create a specialized simulation environment, which is resource and time consuming.

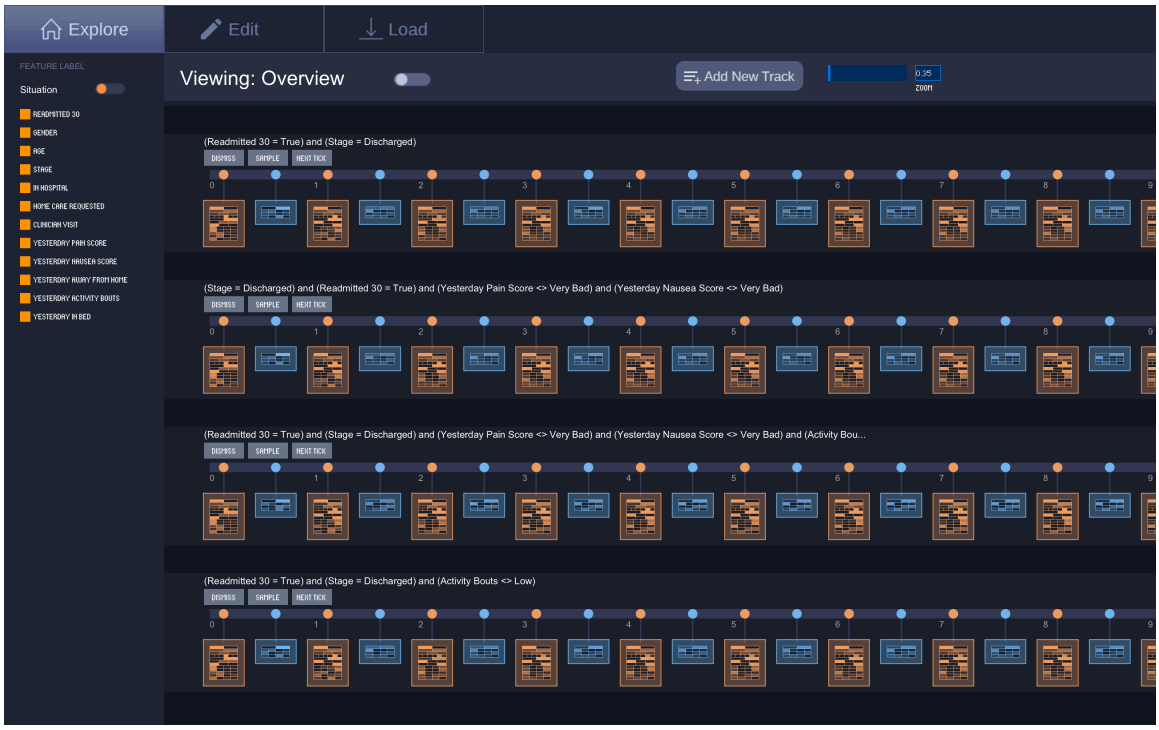

Figure 6. Behavior Dashboard is a computational modeling visual analytics tool that leverages our model of human routine behavior to compute routine variations and deviations from the data, and then present those results in an interactive visualization.

To support stakeholders in their sensemaking process (Pirolli & Card, 2005), we present a visual analytics tool, called Behavior Dashboard (Figure 6), which allows stakeholders to validate their hypothesis about sequences of behaviors that are characteristic of a routine and present their findings visually. Behavior Dashboard is a direct extension of the visualization tool in Figure 1, which we used to validate that our computational model extracts meaningful patterns of behavior by showing overall most salient behavior instances that define routine variations in the data. We changed the visual representation of behavior instances and added an ability to query different aspects of the model and to simulate behaviors in specific situations, an ability important for exploring computational models.

We designed and implemented a preliminary prototype of Behavior Dashboard to help stakeholders enrich their conceptual model of a routine behavior by visually inspecting overview of behaviors captured in the model, followed by querying and simulating routine variations from specific parts of the model, and exploring details of actual behavior

instances. Because it is based on a probabilistic computational model of routines, our tool allows the stakeholders to explore nuanced aspects of routine variations and behavior instances than what is possible with simple aggregation statistics.

8.3.1 Design of Behavior Dashboard

Our main design goal for Behavior Dashboard was to enable stakeholders to explore and understand routine behaviors across domains. In our design process, we focused on domain experts who wish to study human behaviors collected in large behavior logs, although some of our design decisions may generalize to end-users who wish to explore their own behaviors (*e.g.*, individuals in the quantify-self movement). We focus on domain experts with limited data analytics knowledge who seek novel tools to address the challenges of understanding data from large behavior logs. Our design process started with a combination of informal and semi-structured interviews with our primary stakeholders from various domains (*e.g.*, medical, information technology security and privacy) about their current processes to explore data from large behavior logs.

Based on our findings, we generated two design artifacts that helped us ground our design of Behavior Dashboard (see Appendix for materials): 1) a Concept Map, a diagram representing relationship between different concepts and artifacts that users generate in their sensemaking process; and 2) a Primary Persona, or an imaginary domain expert who encompasses actual users we have interviewed and their requirements and needs. Our Primary Persona is a clinician interested in studying patient behaviors from large behavior logs collected using commodity hardware (*e.g.*, smartphones, fitness trackers). Thus, the rest of this chapter uses examples from the medical domain.

Using these design tools, we confirmed that domain experts generally follow an iterative pipeline matching the sensemaking process for data analytics (Pirolli & Card, 2005):

1. Identify a research question
2. Deploy a field study to collect subjects' behavior data logs from their various devices and environments that they hypothesize will help answer the question
3. Explore data to perceive trends and create a conceptual model of behavior
4. Generate hypotheses about the behavior that help answer the research question

5. Search data for examples that prove or disprove the hypothesis
6. Generate new insights, tune hypotheses, or adjust the research question
7. Present findings

The primary means for analyzing data in steps 3 through 7 are various Exploratory Data Analysis techniques that aggregate data across group variables (*e.g.*, grouping clinical outcomes, such as rehospitalized or not, based on age, gender, *etc.*). However, as we have confirmed in our findings, the main challenge domain experts face is exploring *sequences* of behaviors. Some reasons include lack of knowledge about methods and lack of specialized tools that would help them explore sequences of behaviors from data stored in large behavior logs.

In our design of Behavior Dashboard, we also focus on steps 3 through 7, and assume that the domain expert users have already identified a research question and collected relevant behavior log data. Prior to step 3, we assume the domain experts have selected features of interest and converted raw data from a behavior log into behavior instances which we use to train a computational model to generate data structures required for the later visual exploration of behaviors. We also assume the routine model training followed the guidelines from Chapter 3. Given these requirements, we present a design and implementation of specialized visual representations and interactions that allow the users to explore complex routine behaviors.

When designing our visual analytics tool, we ensured the tool follows the Visual-Information Seeking mantra (Shneiderman, 2003): 1) Overview, 2) Zoom and Filter, 3) Details on Demand. Past research (Fekete *et al.*, 2008) has identified this approach as one that offers a fast path to discovery of knowledge in Information Visualization tools. Throughout our design process, we continued to demo our prototypes (ranging from simple low fidelity sketches to fully functional high-fidelity prototype) in a form of a Cognitive Walkthrough (Rieman, Franzke, & Redmiles, 1995) to a stakeholder and an Information Visualization expert. We adjusted our prototypes based on their feedback to ensure usability and usefulness of Behavior Dashboard.

8.3.1.1 Data Organization

Behavior Dashboard organizes behavior data and the corresponding models into projects. Each project comes from a distinct data collection study and is associated with: 1) a training set of behavior instances, a repository of behavior instances from behavior logs used to train the computational model, 2) an optional testing sets of behavior instances, which are behavior instances not used to train the model, and are used to either validate the model or to explore new, unlabeled behaviors not available during model training, 3) and a computational model, which uses an MDP to capture both world dynamics and people's behaviors as enacted given those world dynamics. Each model contains a set of situations. At the start of a session, the user selects a project from a list of available projects to begin exploring corresponding behaviors (Figure 7).

8.3.1.2 Visual Representation of Behaviors

Behavior Dashboard uses a visual language to representation different elements of our computational model of routines. Here, we list all elements of this language and how they relate to the model in Equation 1.

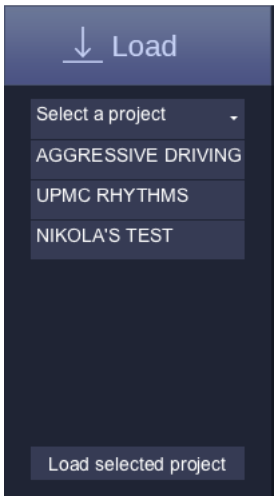

Figure 7. List of available projects the user can choose from to start exploring behaviors using Behavior Dashboard.

Figure 8. Behavior Dashboard represents situations (orange) and actions (blue) as a grid of feature values (first two from the left). Situations and action can be distinguished from other situations and actions by visual pattern of the feature values. The opacity of the borders around the visual items corresponds to probabilities of transitions from visual item to visual item (in order from left to right). The user can find details about situations or actions by hover over a feature value cell (far right), which shows details about the corresponding feature.

8.3.1.2.1 Situations and Actions

In Behavior Dashboard, situations and actions are represented as rectangles that encode three distinct values (Figure 8): 1) color of the rectangle, 2) features and their values organized in a grid; and 3) border. Color distinguishes situations (orange) from actions (blue). Feature grid is organized in rows, where each row represents a feature and all or its possible values. For a distinct situation or action, there is always one active feature value, which is represented as a filled, opaque rectangle; all other inactive features use a hollow rectangle to represent they are inactive. Each situation and action have a border of fixed width, where transparency of the border indicates the probability of that situation ($P(s)$) or action ($P(a \mid s)$) in the model, ranging from fully transparent (0) to fully opaque (1.0).

8.3.1.2.2 Behavior Instances

Behavior instances are represented as sequences of situation-action pairs. The smallest unit of a behavior sequence represents a current situation that a person might be in, the action the person performs in that situation, and the resulting situation (Figure 8). Behavior Dashboard sets the border opacity of each element to the corresponding probability from the model. For example, in Figure 8, the probability of the leftmost situation (s) is based on probability of that situation in the model ($P(s)$), the probability of the action (a) is equal to the probability of that action given the previous situation ($P(s) \cdot P(a|s)$), and the

probability of the resulting situation (s') is based on the world dynamics ($P(s) \cdot P(a|s) \cdot P(s'|s,a)$).

Each situation-action pair in a behavior instance take place over a single time tick. Transitions occurs over two time ticks (the first situation-action pair occurs at the first and the resulting situation in the next). Behavior Dashboard, thus, orders situation-action pairs according to their time tick in a behavior instance (Figure 9). To mark time ticks, Behavior Dashboard draws a circle above each situation and action and marks the position of situation-action pairs with a number corresponding to the index of the time tick in the sequence.

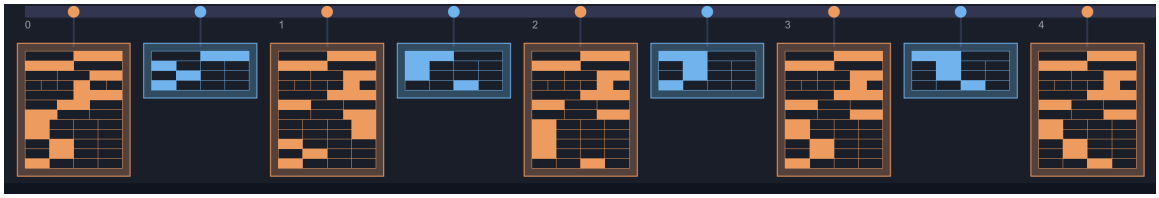

Figure 9. An example behavior instance taking place over four time ticks. Each time tick (*t*) is represented by a circle above a situation and marked with the index of the time tick.

8.3.1.2.3 Routine Variations

Behavior Dashboard represents routine variations (or different possible behavior instances characteristic of a routine) by aligning behavior instances on time ticks and aggregating situations and actions across all behavior instances at each time tick (Figure 10).

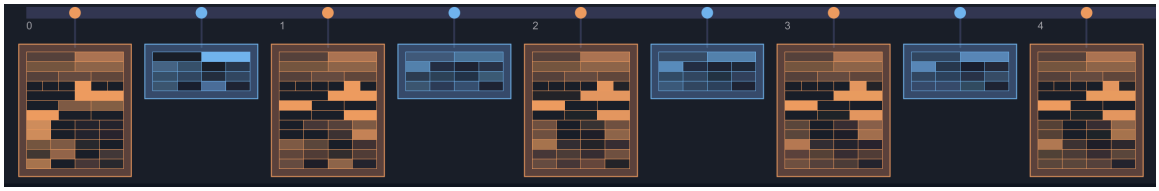

Figure 10. A routine variation, which combines related behavior instances by first aligning them based on time ticks and then aggregates situations and actions for each time tick.

At each time tick situations-action pairs are added together so that the resulting aggregated visual item has (Figure 11): 1) boarder with opacity representing the sum of all probabilities of visual items in the aggregate, and 2) for each feature, the feature values show the

probability distribution over all feature values of individual items. In addition to providing information about how values change within a feature, the aggregate situations and actions also show relationships between different features and their values over time.

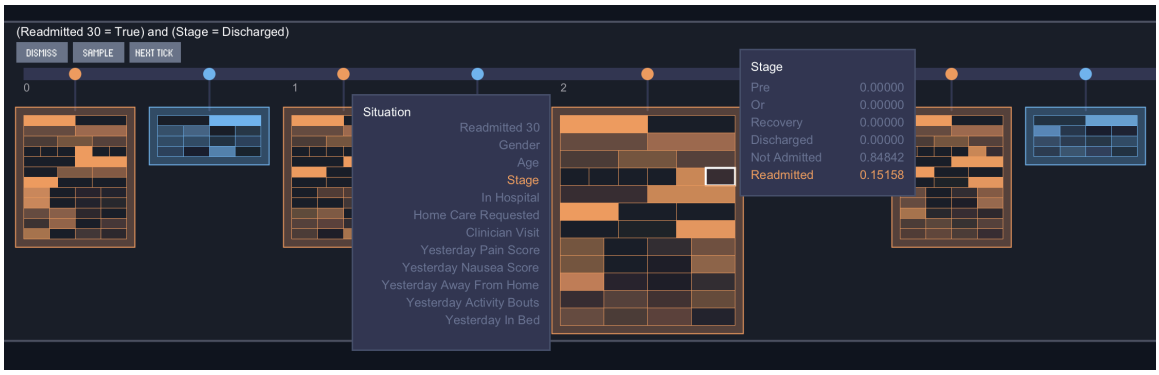

Figure 11. Aggregate situation in a routine variation. Situation detail view shows the probability distribution of values of a feature.

Each routine variation is organized in its own behavior track (Figure 12), which combines related routine variations with a common start time tick. Behavior tracks allow for branching within a routine variation to explore different behavior alternatives within a routine. In the next section, we detail how the user interacts with these visual items to explore behaviors from large behavior logs.

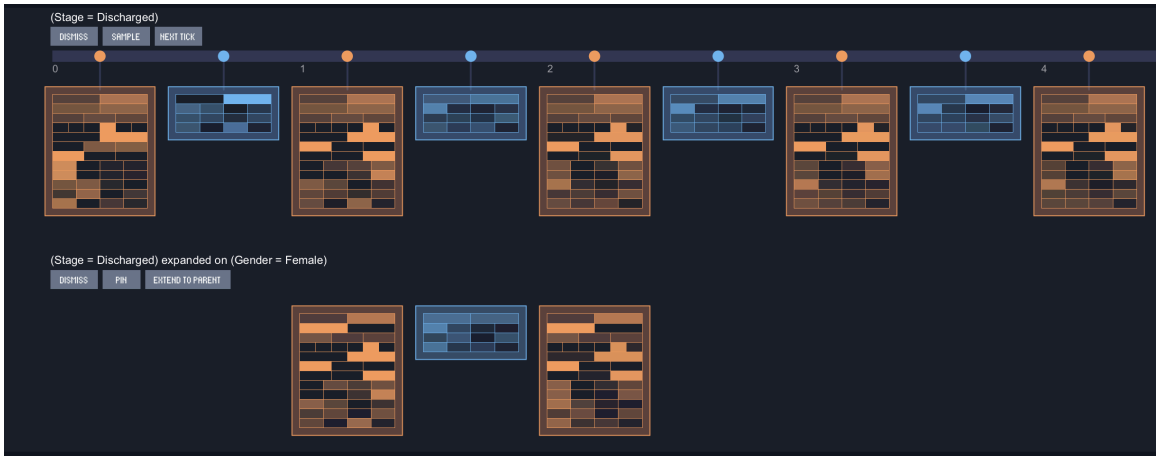

Figure 12. A behavior track containing a routine variation starting at initial time tick t_0 and an alternative routine variation branch starting at time tick t_1.

8.3.1.3 Interactions with Behaviors

Behavior Dashboard enables a set of interactions with the visual items we described in the previous section that help the user to explore and understand behaviors captured in a computational model of routines.

8.3.1.3.1 Behavior Overview

The user begins behavior exploration with an overview of all behavior instances across all time ticks aggregated into a single aggregate situation (Figure 13).

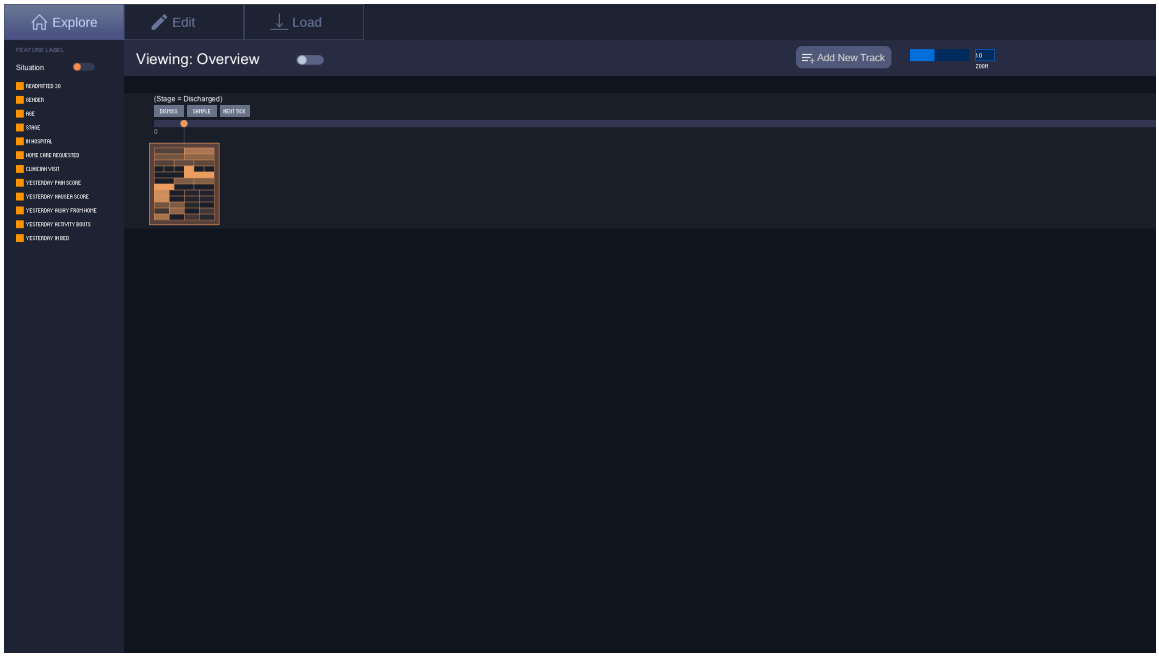

Figure 13. Starting screen of Behavior Dashboard after loading a project. The initial behavior track shows an overview of behaviors captured in the model as a single overview aggregate situation.

Detail view of the aggregate (Figure 14) shows probability distribution of feature values for each feature for the entire model and all of the routine variations and deviations it identified in the data. This view allows the user to identify features of interest based on their likelihood in the model.

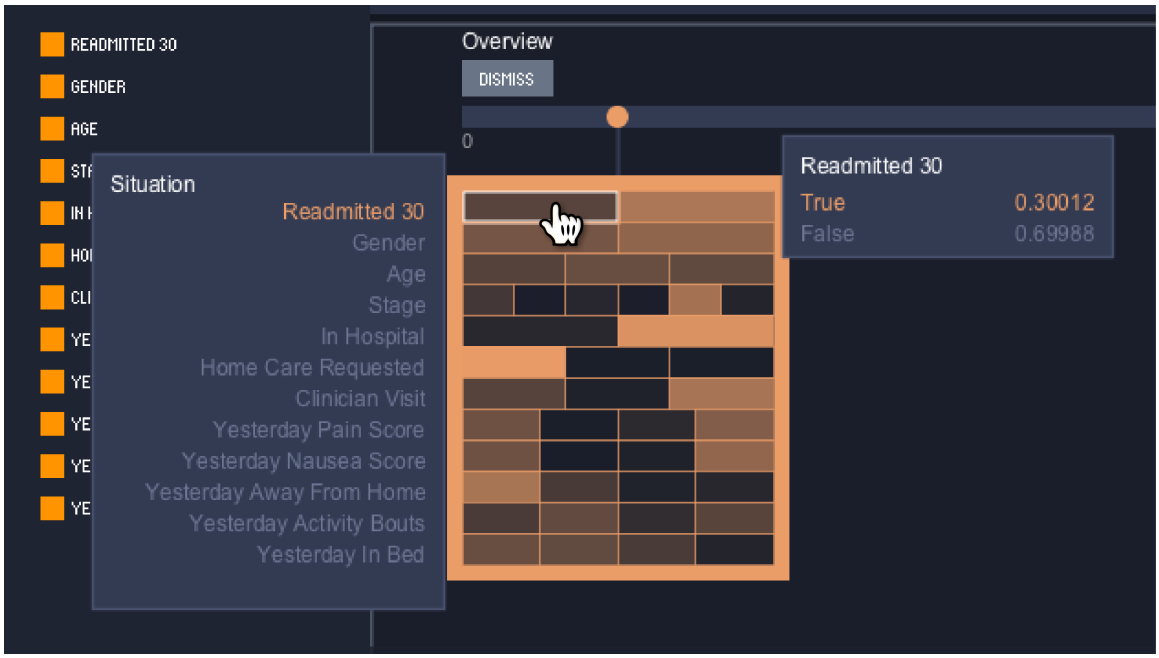

Figure 14. Starting overview aggregate situation. Overview combines all routine variations and deviations from a model across all possible time ticks into a single visual item detailing the probability distribution of all situation feature values in the model.

8.3.1.3.2 Filtering Situations and Actions

Behavior Dashboard enables the user to filter situations and actions based on feature value by clicking on a feature value of a situation or an action. This expands the node and creates a new sub-track within the original track (Figure 15). The user can reorder behavior tracks and sub-tracks by dragging them and placing them to create a new vertical order. Behavior Dashboard computes probability distribution of feature values of the new, expanded aggregate situation or action by conditioning the probabilities on the value of expanded feature.

Figure 15. The original overview aggregate situation expanded on a feature value. The resulting expanded situation is placed in the same behavior track for easy visual comparison.

8.3.1.3.3 Exploring Details of Data Behavior Instances

The users can load and explore actual behavior instances from the data. These behavior instances could either be part of the training data set or validation instances that were not used to train the model and instead are used to validate the model. To load a behavior instance, the user specifies a data set, selects a particular participant from the data collection study, and then selects one or more behavior instances from that participant. The user can then select features that Behavior Dashboard will use the model to predict based on the value of other features in the behavior instance. This allows the user to validate how well the model predicts people's behaviors. For any behavior track, the user can turn the time forward any number of ticks by pressing on the "Next Time Tick" button and selecting the number of ticks in the resulting pop-up dialog.

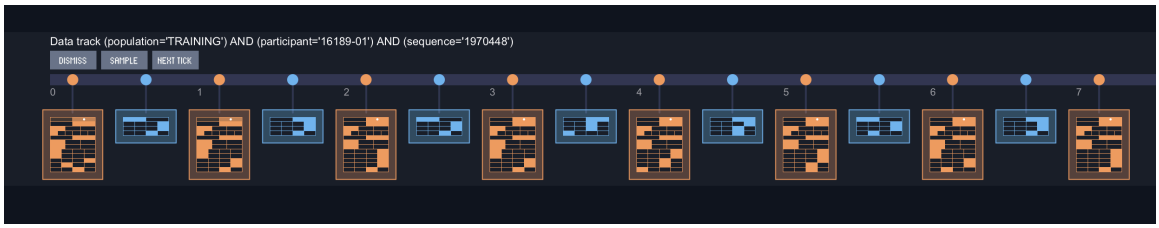

Figure 16. A behavior instance loaded from the data. The behavior track label (top) shows the source of behavior instance. The buttons allow the user to dismiss this behavior track, sample from the behavior track, or fast forward in time by selecting a number of next time ticks. The behavior instance in this example shows seven time ticks. The user selected to have the probability distribution of feature value for the feature in the first row of each situation computed from the model. White circle on top of the feature value cell indicates actual value in the data.

8.3.1.3.4 Exploring Hypothetical Routine Variations

Users can "zoom" into parts of the model based on situations or actions of interest by querying the model and creating special query behavior tracks. The user can specify queries using the dialog in Figure 17. The dialog enables the user to specify situations and actions that will or will not be part of the variations using simple equals or not equals operators. Each query part can apply to only the first situation that matches the query ("not fixed") or be applied to all subsequent time ticks ("fixed").

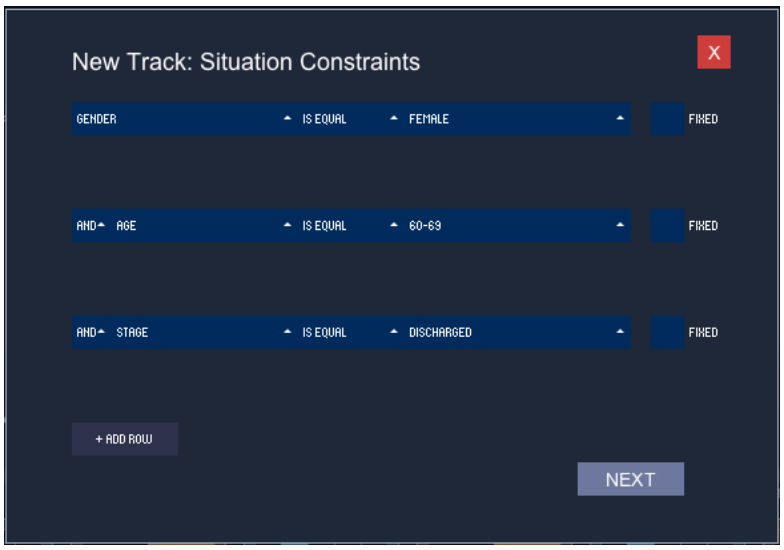

Figure 17. Model query dialog. The user can specify a query that selects only specific situations and actions based on their feature values.

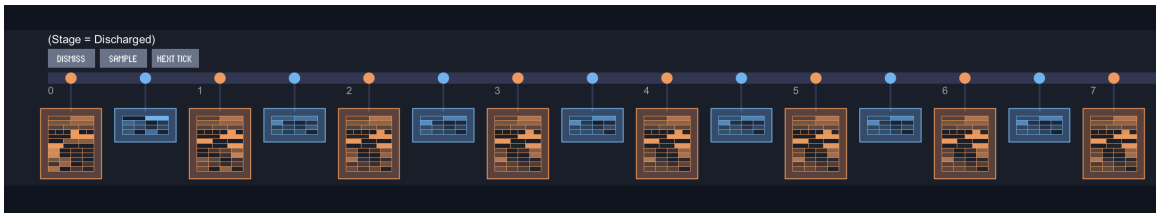

Figure 18. A behavior track matching a query. Such tracks can be used to ask "what-if" questions from the model.

This creates a new behavior track containing all the routine variations that match the specified query (Figure 18). The user can then filter routine variations by feature values using the expand feature, “zoom into” specific variations in the expanded tracks, and request details about both behavior instance that are part of the variation by using the sample feature and details of situations and actions by hovering over their respective feature values.

8.3.2 Use Case: Understanding Behaviors that Leads to Rehospitalization

Here we show how clinicians can use Behavior Dashboard to understand behaviors of their patients that lead to rehospitalization. In our scenario, we assume expert use of the system. Here, we illustrate various features of Behavior Dashboard and their use in the sensemaking process.

8.3.2.1 Behavior Analysis and Results

We started out analysis by visually exploring the overview visual item (Figure 15). The overview showed that model estimated that each participant has approximately 30% chance to be rehospitalized. This is slightly higher than 27.45% of patients in our study. The model uses statistical principle of Maximum Entropy (Jaynes, 1955) to generalize to unseen data and estimate these probabilities based on the information present in the data and the size of the data set. For example, the estimated percentage is accurate if we consider participants who have dropped out of the study due to worsening health. Thus, we conclude that our model approximates the rate of rehospitalization in our population well.

8.3.2.1.1 Behavior Overview

We confirmed that other demographic data follows the distributions from the data. However, we also noticed that there is a high probability of patients not reporting their symptoms (approximately 40% for pain and nausea). Similarly, the model estimated that 67.58% days do not have recorded information about patient location, but fitness tracker data was missing for only 24% of days for step tracking and 44% of days for sleep tracking. A large portion of missing subjective symptom ratings were likely due to an inability to comply during hospital stays. Filtering by stage allowed us to confirm this hypothesis during the time patients spent in the hospital or on the day they were discharged. However, the missing objective location data was due to tracker failures, which prevented us from a more detailed analysis of macro-level mobility.

Behavior Dashboard also estimated that patients are unlikely to rate their symptoms very bad (worst pain and nausea they have ever experienced in their life), which were less than 1% for both. These are encouraging results because it means that patients did not suffer such severe problems very often. However, worsening symptoms are of interest to us because they could mean that the patient is likely to be rehospitalized. This highlights the importance of having our computational model that can predict symptoms even when symptoms data is missing (*e.g.*, predicting missing self-reported symptoms in future User Interface that provide automated medical interventions). Unlike simple visualizations of raw behavior instances, Behavior Dashboard still allows us to explore behaviors even in infrequent situations because it uses the underlying computational model to generalize to unseen situations and estimate probability of routine behaviors in those situations, too.

Overview of the model also gives us an opportunity to reproduce analysis similar to existing Exploratory Data analysis approaches. To illustrate this, take for example past research (Low *et al.*, 2018) that could not find a significant effect of demographics on rehospitalization rates. The overview showed an estimated distribution of genders with 43.81% female and 56.19% male. We then conditioned (expanded) the overview visual item on both genders and explored the readmission rates for the two genders. Our model computed that probability of readmission for female patients $P(Readmitted = True \mid Gender = Female) = 25.89\%$ and probability of readmission for male patients

$P(Readmitted = True \mid Gender = Female) = 33.22\%$. Visually, this difference is small (Figure 15), which is supported by the low Bayes Factor ($K = 1.28$) giving more support for previous findings (Low *et al.*, 2018).

8.3.2.1.2 Effects of Self-reported Symptoms on Rehospitalization

We then proceeded to explore the effects of symptoms and patient mobility on rehospitalization. Unlike the previous analysis (Low *et al.*, 2018), here, we focus on behaviors over time to understand the nuances of patients' behaviors. Because the goal of our analysis is to understand the behaviors of patients that lead to rehospitalization after they have been discharged, we used Behavior Dashboard query functionality to focus our further analysis only on behaviors after patients have been discharged. We queried the model and zoomed into the situation that represents the day the patients are discharged from the hospital after surgery. We then simulated behavior over the next 30 days (30 time ticks) using model predicted behaviors in our select situations.

Behavior Dashboard showed that patients across genders and age groups are unlikely to provide symptom ratings or may experience some pain and nausea in the first week after being discharged. For example, the model estimates that approximately 86% of patients will not enter a symptom on the day they are discharged. This is expected because patients may still be feeling ill shortly after surgery or may not have the time to enter symptoms due to moving back home. Although, the model continues to predict low compliance for symptom ratings (down to approximately 30% after first 7 days), it also shows that patients' symptoms improve. For example, the model shows that pain stabilizes by day 7, going down from approximately 21.62% chance of experiencing the worst pain a day after being discharged down to less than 0.1% chance on day 7. This decrease in symptoms is expected if the patients' recovery after being discharged is going well.

The ability of our algorithm to predict rehospitalization in early stages is probably due to the poor symptoms many patients experience and report soon after being discharged. This is evident in both the empirical data and in our evaluation of our readmission predictor from Chapter 7. Behavior Dashboard shows that there is about 17% chance of being rehospitalized within the first 7 days which then tempers off and slowly increases by day

30, at times by approximately only 0.1% per day. This means that some early symptoms are indicative of patients' rehospitalization soon after being discharged.

We then used Behavior Dashboard to simulate the behaviors of patients who never experience very bad pain and nausea (Figure 19 and Figure 20, second routine variation from the top), and we found that the model estimates that their rehospitalization rates decrease significantly, down to less than 0.01% on day thirty. Such patients are able to perform more physical activity and more likely to spend a reasonable amount of time in bed. Thus, catching these symptoms early could help clinicians prescribe interventions that could improve poor symptoms and prevent rehospitalization.

8.3.2.1.3 Effects of Mobility on Rehospitalization

Previous work has hypothesized (Low *et al.*, 2018) that increased physical activity could improve patients' symptoms and thus improve rehospitalization outcome. However, the most likely routine variation for rehospitalized patients was staying at home with low activity bouts, and low to medium time spent in bed; although many spent more than 10 hours per day in bed. Although we expect such low activity in the first few days after being discharged, Behavior Dashboard showed that there is a large probability that patients will continue to remain sedentary each day, even as their symptoms improve. This leaves an opportunity to motivate patients to move more.

Thus, we used the model to simulate changes to the readmission rates across patients, to motivate patients to track their steps and take approximately 720 or more steps per day; a small but realistic increase in physical activity level for this population (Figure 19 and Figure 20, third routine variation from the top). We also simulated motivating patients to stay in bed for no longer than 10 hours per day. Behavior Dashboard estimated that patients are likely to make slow progress increasing their activity in the beginning, but the probability of them being able to make at least 1,420 steps per day on day 30 increases from approximately 28% to approximately 43% if successfully motivated to move more. Behavior Dashboard estimates that this small change in physical behavior might not be predictive of readmission rates in the first 7 days (the readmission rate remains at approximately 17%), but that continued activity predicts reduction in probability of

readmission by approximately 4% by day 30. Similarly, Behavior Dashboard estimated that motivating patients to spend less than 10 hours in bed predicts reduction in readmission rates by approximately 3% by day 30.

Behavior Dashboard estimates that such an increase in physical activity may not be associated with improved pain and nausea symptoms, which the model predicts with or without increased physical activity. This indicates that it is more likely that symptoms affect patients' ability to perform physical activity than other way around. We then queried model for behaviors of patients without very bad pain and nausea, and with medium or high percentage of activity bouts, and the model predicted only marginal improvement of the rehospitalization outcome (approximately less than 0.1% decrease compared to unconstrained activity bouts).

This means that the model predicts that there will be more opportunity to provide patients with interventions as their symptoms improve. For example, we simulated a holistic approach to motivating patients to have healthier routines, where we hypothesize that we are able to motivate them to walk more and to spend only between 5 and 10 hours in bed. Behavior Dashboard estimated that, when able to do it given their symptoms, there is approximately 5% probability that patients with this kind of routine would be readmitted in the first 7 days (compared to approximately 17% probability without any intervention); this probability then remains constant all the way up to day 30 (Figure 19 and Figure 20, last routine variation).

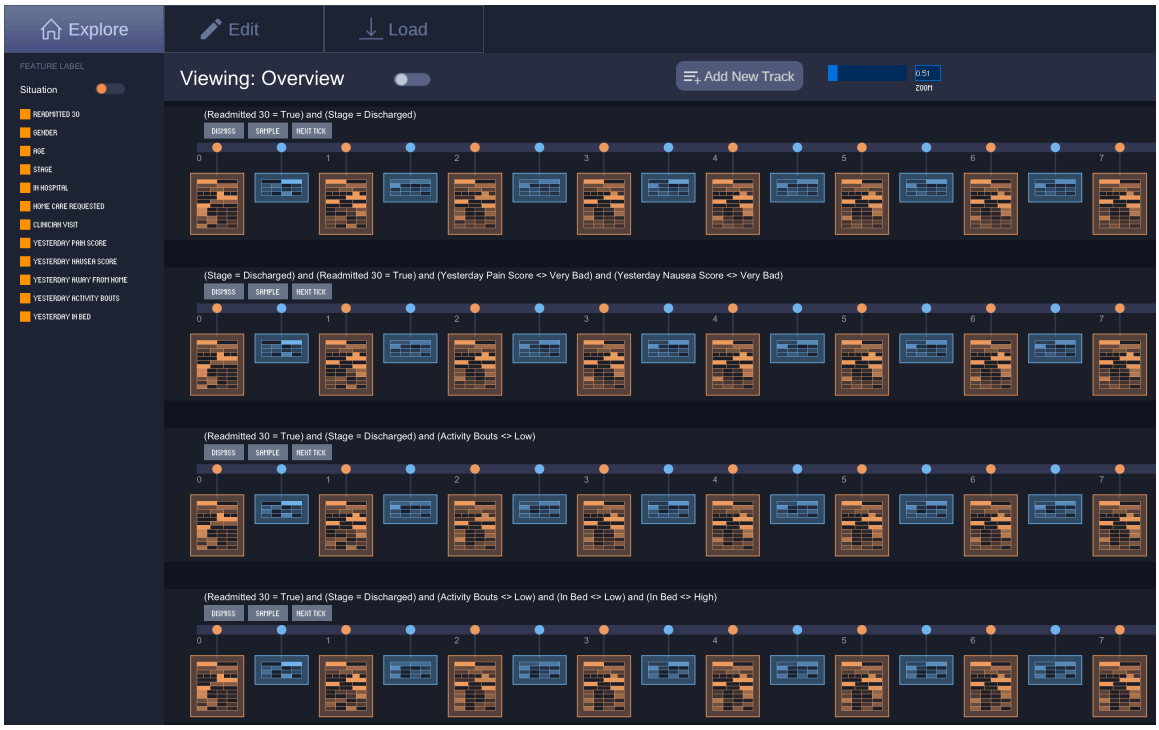

Figure 19. Our analysis of four behavior tracks showing model estimated patient current routine variation (top) and various hypothetical interventions in the first week after they have been discharged.

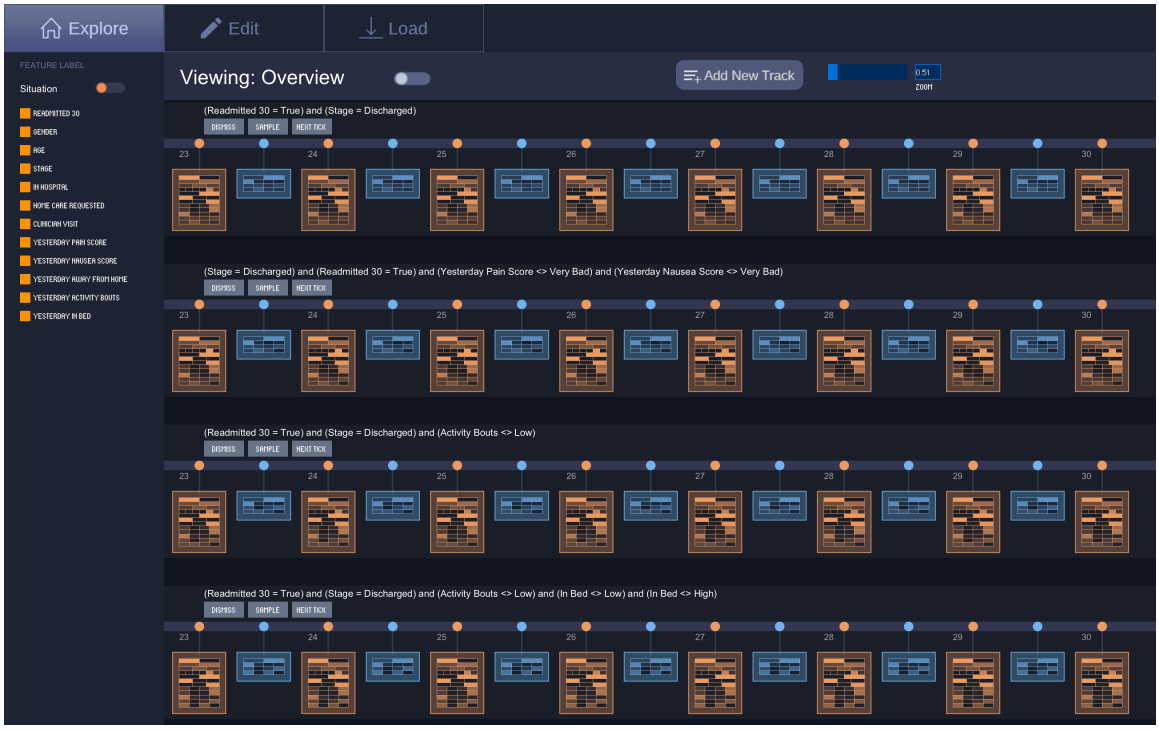

Figure 20. Our analysis of four behavior tracks showing model estimated patient current routine variation (top) and various hypothetical interventions in the last week of the study.

8.3.2.2 Discussion

In this use case, we showed how a computational model of routines of patients who have undergone a surgery to remove cancer could help clinicians generate and test hypotheses about patients' health. We started with a research question that seeks to answer if increased patient mobility could lead to more positive rehospitalization outcomes. With this research question in mind, we used our methodology to train a model on data collected from real patients. We picked the features in collaboration with their actual clinicians based on previous knowledge in the domain and their desire to explore how data collected from commodity hardware could capture people's behaviors. We engineered the features according to our methodology to capture the economics of patients' behaviors as they go through the surgery and subsequent recovery process.

Earlier in Chapter 7, we showed that the model is able to predict at-risk patients before they are rehospitalized. We have shown that our detection algorithm can in some cases detect at-risk patients within the first week of them being discharged. This is in line with our analysis using Behavior Dashboard that has shown an increase in readmission probability in the first 7 days after being discharged. Such automated prediction enables future automated interventions that can detect at-risk patient and propose a treatment.

One such intervention is to motivate patients to avoid remaining sedentary for long and to perform light physical activity, such as walking (Low *et al.*, 2018). We showed how Behavior Dashboard can help clinicians validate this hypothesis that physical activity (when possible) could lead to better rehospitalization outcomes. Although motivating patients to perform light physical activity might not directly impact their symptoms (*e.g.*, pain and nausea), our model estimates that motivating patients to walk and spend less time being sedentary is a potentially viable intervention that could lead to small improvements in rehospitalization outcomes. Although for an average healthy person, that kind of activity might not be enough to show health benefits, out computational model shows that it could have a positive effect on the rehospitalization outcome of cancer patients.

Behavior Dashboard supported this by enabling us to ask the following question: what if we were able to motivate patients to wear their fitness tracker and walk more. We used Behavior Dashboard to narrow down our search to a specific part of the model (after the patients are discharged) and then simulated behaviors in those situations to simulate both current patients' behavior and a possible intervention and estimate corresponding rehospitalization rates. This allowed us to explore potential interventions without actually running empirical studies in this early formative stage. Using knowledge generated from Behavior Dashboard allows clinicians to pick the most promising interventions and to only test those using empirical studies, which saves time and resources.

8.4 Summary

In this chapter, we showed how our computational modeling methodology can help stakeholders in the behavior sensemaking process. We showed how our methodology helps stakeholders explore and understand behaviors from large behavior logs in all stages of the sensemaking process (Pirolli & Card, 2005). Our computational modeling methodology automates aspects of the information foraging loop part of the sensemaking process by extracting salient patterns and searching for relationships in the data that describe routine behaviors. We have shown how to schematize this collection of evidence into an automatically trained computational model of routines that can later be used to generate and test hypotheses about behaviors in the model.

We presented two tools that support exploration of these salient patterns and the transition between the foraging loop and the sensemaking loop. The driving simulation tool focused on helping drivers schematize information about aggressive and non-aggressive driving routine variations. By automatically detecting aggressive driving instances and contrasting them with simulated non-aggressive behaviors and visually presenting them using animation we helped drivers identify examples of behaviors that support their conceptual model of driving behavior. However, our driving simulation tool is a domain dependent sensemaking tool designed specifically for end-users in that one particular domain.

We addressed this issue by building Behavior Dashboard, a general-purpose human behavior data computational modeling and visual analytics tool that support the routine sensemaking process more broadly. We illustrated how Behavior Dashboard can be used to visually explore, filter and search for routine variations identified using our computational model, generate and validate hypotheses about classes of routine behavior, and visual present the data in a way that tells a story about the behavior. We have illustrated how this process allows the stakeholders to describe, reason about, and act in response to people's behaviors.

9 CONCLUSION AND FUTURE WORK

The goal of this work was to create a method for exploring and understanding complex human routine behaviors from large behavior logs. We focused this work on routine behaviors because they describe the structure of and touch on almost every aspect of people's lives. As such, studying this type of purposeful behavior will be paramount for understanding tasks that people perform and the goals they will want to accomplish when interacting with future Information Technology. We illustrated our approach through use cases in four distinct domains: 1) daily commutes, 2) driving safety, 3) mobile device usage, and 4) patient treatment and care. In each of these domains, we discussed the potential for future, personalized Information Technology that could help improve the quality of people's lives.

Our focus on understanding behaviors differentiated our method from traditional Machine Learning approaches that seek to simply predict or act in response to people' behaviors. Such existing methods minimize the prediction error by optimizing some loss function. Although such approaches are very good at modeling the data, they offer no guarantees that they are modeling the processes that generated the data. This is because the loss functions they use are not derived from processes that guide people's behavior.

Even existing Data Mining approaches seek to recover patterns of behavior based on optimization functions that might not be representative of those behaviors. As such, we argued that they are not well suited for our main goal—to understand the underlying processes that form the behaviors we want to study. Unlike the existing data mining-based methodologies that use algorithmic approaches to extract valuable features and salient patters from the data, our approach is almost exclusively hypotheses driven. We made this choice to ensure that every step of our methodology can be explained with theory of behavior we are modeling.

Thus, in our methodology, stakeholders begin their exploration by defining the type of behavior they are interested in modeling (*e.g.*, routines) and identifying features of interest that they believe influence behavior. We illustrated how to define behaviors in Chapter 4 where we presented our unified definition of routines. Our definition combined properties

of routine behavior we identified from existing work in a way that allowed us to operationalize it in a computational model of routines. This definition provided a grounding for our choice of algorithm and feature engineering considerations to train our computational model from data stored in large behavior logs.

Our choice of algorithm in Chapter 5 was influenced by our routine definition and the need to capture goal-oriented aspects of such purposeful, yet inherently uncertain and variable human behavior. We chose Markov Decision Processes (MDP) (Bellman, 1957) as a data structure because it allowed us to encode relationships between situations and actions that people perform in those situations probabilistically. We leveraged Inverse Reinforcement Learning (Ng & Russell, 2000), which has been traditionally used to recover a policy of an agent from demonstrated behaviors, to estimate the probabilities of behaviors from the data. We specifically use MaxCausalEnt IRL algorithm (Ziebart, 2010) because it tries to establish a causal relationship between situations and actions as represented in the data. In Chapter 6, we showed how to engineer features we use to train our model to ensure that the policy we recover considers economics of routine behaviors.

The rest of this work focused on leveraging the computational model to describe, reason about, and act in response to behaviors, stored as event traces in large behavior logs. We showed that such a model can aid stakeholders in sensemaking about behavior of individuals and populations through understanding how behavioral features interact in the processes that describe people's enacted behaviors. We grounded our methodology in the sensemaking process (Pirolli & Card, 2005), which splits exploration and understanding of such data into two loops: 1) information foraging loop, which helps stakeholders develop a conceptual model of behaviors, and 2) the sensemaking loop, which helps stakeholders explore the conceptual model to gain better understanding of routines. Together the two loops allow stakeholders to generate knowledge about routines.

One of the goals of our methodology was to automate aspects of the foraging loop and automatically detect and extract salient patterns of behavior that characterize a routine. Using our computational model as a data mining tool, we reduced the cost of manually searching for evidence of patters of behaviors in the data (Russell *et al.*, 1993). We

validated this by first showing that our computational modeling approach can extract and identify salient and meaningful patterns of behavior characteristic of a routine (*i.e.*, routine variations) in Chapter 5. Later in Chapter 7, we presented and validated an automated method for detecting and simulating classes of routine variations and differentiating them from deviations and other uncharacteristic behaviors.

We then showed how such information can help stakeholders organize information to enable them to enrich their conceptual model of routine behaviors in Chapter 8. We designed and implemented two different routine visualization tools and showed how they can help both end users and domain experts to search for relationships in the routine models to generate and test their hypotheses about behaviors. We showed this in both a driving domain (where we automatically detected and simulated behavior instances characteristic of a driving routine) and a healthcare domain (where we automatically detected at-risk patients and predicted outcomes and viability of a potential treatment).

Our interactive visual representations of routine data demonstrated the ability of our approach to present findings about human behaviors and give stakeholders a holistic picture of this type of human behavior. We have illustrated the ability of the model to generate behaviors also allows the stakeholders to generate hypothesis in different "what-if" scenarios. We also showed that the model can support interfaces that can detect and extract salient patterns of behavior that characterize a routine, and act in response to those behaviors to prescribe behavior change.

This work was in part influenced by the growing movement to make Machine Learning (ML) and Artificial Intelligence (AI) more usable, explainable, and interpretable. As such, it has broader applications in understanding capabilities and limitations of complex AI systems (an important aspect of usable AI), and in the field of mixed-initiative computational modeling, which helps domain experts build more accurate representations of their complex systems they want to model. Our routine behavior model's ability to automatically reason about and act in response to routine behaviors also opens up opportunities for creating a new class of human-data supported interfaces that can

automatically learn about people's behaviors and use this knowledge to act in response to those behaviors.

9.1 Mixed-initiative Computational Modeling

Human experts can improve the accuracy of Machine Learning models by manually producing sequences of examples that explain a concept and interactively training the model (Cakmak & Thomaz, 2011). We have shown in our studies that different stakeholders often have at least a high-level conceptual model of the behaviors they study and the world dynamics in which the behaviors take place. We can leverage this knowledge to help the model learn a more accurate representation of behaviors and world dynamics faster. We have already shown that manually specifying situation transitions that are not possible or transitions that people have no control over could significantly reduce the training time for our computational models. For example, we have used the knowledge that it is not possible for a vehicle to exit an intersection and then suddenly appear before the same intersection again no matter what action the driver performs. It is also not possible for the driver to press on the brake pedal in a stopped vehicle and have the vehicle accelerate. Stakeholders may also know (*e.g.*, the weather), which could help estimate the effects of such factors faster.

However, the current model requires the stakeholders to manually specify what actions are possible in an environment and how the environment responds to people's actions and other external factors that operate in and influence the environment. We can manually specify such world dynamics when they are known ahead of time. This is often the case when people's actions fully describe situation transitions (*e.g.*, when the model considers only factors that people have full control over in the environment). For example, it is easy to specify world dynamics in a routine model of people's daily commutes between different places that are all known ahead of time because the person always ends up at the place they indented to go to or stay at with 100% probably. However, if we introduce more external factors into the model, we must also estimate the effect of those factors on the environment. For example, suppose the stakeholder adds information about the weather to the model to

understand how it impacts people's commute. It is possible for the weather to change from sunny to cloudy no matter what the person does (*i.e.*, stays at the same location or leaves to go elsewhere). In this case, we must model both situation transition probabilities when the weather stays the same and when the weather changes over time. Such world dynamics are often not known ahead of time, and even if they were, it may be tedious to encode such dynamics manually when they are driven by multiple variables.

Automatically learning possible world dynamics from the data is challenging because it requires a large number of training examples to accurately model its complexity. For example, in a model where there are $|S|$ number of situations and $|A|$ number of actions, we need to estimate situation transition probability distribution ($P(s'|s,a)$) for $|S| \times |A| \times |S|$ number of transitions. This problem is compounded when modeling human behavior from behavior logs. In this case, transitions involving actions that represent deviations from a routine will be infrequent in the data (by definition). Some possible, but infrequent transitions will also not be well represented in the data. However, the nature of log studies prevents the stakeholders from asking people to go into their environment and perform such actions and hope they end up in situations that we have not observed. Even in situations when the stakeholders could contact people, asking them to perform specific actions might be cumbersome (*e.g.*, if it requires time and resources), inappropriate (*e.g.*, if they are unable to perform such actions), or unethical (*e.g.*, if those actions could negatively impact them).

Future work should explore different strategies to guide stakeholders to apply their knowledge about the world dynamics using a mixed-initiative learning approach (Suh & Com, 2016) to estimate situation transition probabilities for a routine model. Future researchers in this area should work closely with stakeholders to study their process for understanding behaviors from empirical data. This will inform design changes to our visual analytics tool. Our goal is to modify Behavior Dashboard and add an interactive component to it, which will allow the stakeholders to specify domain knowledge that will aid in model training. We will explore how to teach stakeholders to apply model training strategies in the information foraging loop to improve their ability to conceptualize routines and different intervention outcomes in the sensemaking loop. This will extend our routine

models to estimate a more accurate representation of the world dynamics from the data. Such an accurate representation also helps better estimate the probability distribution of actions in that environment. The proposed changes to the training algorithm will result in models that will allow the stakeholders to generate and test more realistic hypotheses about the situations that people find themselves in and the actions they perform in those situations. This could also improve the ability of the model to detect and generate more realistic routine variations.

9.2 Understanding Capabilities and Limitations of AI

Current advances in Artificial Intelligence (AI), including reasoning, knowledge representation, planning, learning, and perception, are already changing many aspects of our lives (Stone *et al.*, 2016). This rapid influx of Information Technology and AI into people's lives is in part enabled by computational advances. For example, advances in computational power has brought together various Large-scale Machine Learning and Deep Learning methods (Jordan & Mitchell, 2015) that have revolutionize fields of healthcare (Shin *et al.*, 2016), autonomous transportation (González *et al.*, 2016), and even gaming (Silver *et al.*, 2016), to name a few.

However, rapid advances in AI have also spawned concerns that such technology could have a negative impact on people's lives, for example by increasing inequality and threatening democracy (O'Neil, 2016) and even presenting an existential risk (The Economist, 2015). Despite dismissing some of these concerns as fictional, the first report from the "One Hundred Year Study on Artificial Intelligence" (Stone *et al.*, 2016) concludes that "it remains a deep technical challenge to ensure that the data that inform AI-based decisions can be kept free from biases that could lead to discrimination."

One of the main challenges is that most of the existing successful applications use black-box technologies, whose inner workings cannot be examined to determine that they are not negatively impacting people for whom they make decisions (Pasquale, 2015). Although much existing work has tried to address issues of interpretability and explainability of algorithms for ML experts (Abdul *et al.*, 2018), little work has been done for other

stakeholders. For example, it is easy to imagine a future in which User Experience (UX) designers will be able to pick existing ML models "off the shelf" and used them as a design material to imbed them into their future user interfaces. However, currently there is lack of ability for this important group of stakeholders to explore and understand capabilities and limitations of existing technological advances and algorithms in Machine Learning and AI (Yang, Banovic, & Zimmerman, 2018).

Future work in HCI should therefore explore ways and create methodologies to bridge this gap between AI and UX design. Inspired by how designers communicate with other materials (*e.g.*, bending and cutting out carboard to create new shapes), we propose a framework which allows exploration through interaction with existing AI black-box algorithms to enable stakeholders to learn about the capabilities and limitations of those algorithms. This will enable a future in which UX designers will be able to seamlessly integrate AI and ML advances into their future Information Technology designs and products.

9.3 Human-data Supported Interfaces

User interfaces that learn about people's behaviors by observing them and interacting with them enable a future in which technology helps people to be productive, comfortable, healthy, and safe. In this future, such *human-data supported interfaces* will automatically reason about and describe common user behaviors, infer their goals, predict future user actions, and even coach users to improve their behaviors. In this future, a mobile phone interface learns from a user's behavior to proactively clear out the user's email inbox and schedule meetings. A smart home interface learns about the residents' daily routines to control heating in a way that reduces their energy bill while making sure they are comfortable in their own home. A medical informatics user interface aids a clinician in understanding patient data, diagnosing a chronic condition, and finding the best treatment that is personalized for the patient based on the patient's behavior. A car interface that detects when a user is driving aggressively coaches the driver to drive less aggressively by showing what a non-aggressive driver would do in the same situation.

Human-data supported interfaces offer personalized experiences based on users' behaviors by establishing a common ground with the user through observing and interacting with them. Such interfaces will be able to learn virtually any user behavior, thus opening up possibilities for user experiences that touch on every aspect of people's lives. However, in the absence of technology that can automatically learn and encode knowledge about people's behaviors, interface designers opt to hardcode limited knowledge, beliefs, and assumption about behaviors into their interfaces. User Interfaces that are not supported by a computational model cannot act fully autonomously and can only respond to a subset of predefined commands in well-defined environments to accomplish specialized tasks.

This work illustrated capabilities of my routine model to automatically detect and describe classes of behavior, and act on the behaviors it detects to prescribe changes using a human-data supported interface. We have already shown the applicability of such models to people's mobility, driving routines, and behaviors of patients with chronic conditions. Our computational model of routines is particularly suited for such interfaces because of its ability to probabilistically reason about behaviors, explain reasoning decisions to the user, and act without making decisions with irreversible negative consequences under uncertainty.

Future work should therefore study how such interfaces can leverage computational models to improve people's wellbeing and help them be productive, healthy, and safe. Future work should further explore the capabilities of human-data supported interfaces to automatically detect suboptimal behaviors and generate coaching instructions, and study how such interfaces can help people learn and apply the guidance in other domains. For example, this work offers a direction for human-data supported intelligent tutoring systems that will help students anywhere, at any time, and at a fraction of cost of a human coach. Our computational models can also detect and reason about optimality of a behavior even for infrequent behaviors or behaviors that it has not trained on. For example, our routine model could be applied to detect abnormal user behaviors that may indicate a compromised information system. A similar approach has applications in detecting when people with disabilities face emergency situations in public transit.

9.4 Summary

In summary, this work sets foundation for modeling the human accurately across domains to support design, optimization, and evaluation of user interfaces to solve a variety of human-centered problems. It is a step towards addressing the grand challenge of establishing theoretical foundation for work in HCI in which computational models provide a quantitative method to explore and understand complex human behaviors. The ability to model how people interact with information technology is essential to offer people services in an intelligible and autonomous way. The work on exploring and understanding complex computational models of human behavior has direct implications on study of intelligibility of complex computational systems to provide tools that ensure correctness of such systems. This work enables a future in which User Interfaces powered by Artificial Intelligence have a positive impact on society through improving the quality of people's lives.

10 APPENDIX

10.1MaxCausalEnt IRL Algorithm Implementation

```
import argparse
import MySQLdb
import numpy as np
import tensorflow as tf
import random
import itertools

PARSER = argparse.ArgumentParser(description=None)
PARSER.add_argument('-s', '--study_id', default=0, type=int, help='study id')
PARSER.add_argument('-p', '--training_population_id', default=0, type=int, help='study id')
PARSER.add_argument('-r', '--run_id', default=0, type=int, help='run id. if -1 then new run
will be created, otherwise existing run will be used.')
PARSER.add_argument('-f', '--fold_id', default=0, type=int, help='fold id. if -1 then new fold
will be created, otherwise existing fold will be used. also if not -1 then fold transitions
used.')
PARSER.add_argument('-t', '--compute_transitions', default=0, type=int, help='load transitions
from database or calculate on the fly (0:database only, 1:recalculate)')
PARSER.add_argument('-l', '--learning_rate', default=0.1, type=float, help='learning rate')
PARSER.add_argument('-n', '--n_iters', default=200, type=int, help='number of iterations')
PARSER.add_argument('-m', '--max_sequence', default=100, type=int, help='largest sequence
length')
PARSER.add_argument('-b', '--batch_size', default=100, type=int, help='number of training
examples in each gradient batch')
ARGS = PARSER.parse_args()
print ARGS

STUDY_ID = ARGS.study_id
TRAINING_POPULATION_ID = ARGS.training_population_id
RUN_ID = ARGS.run_id
FOLD_ID = ARGS.fold_id
COMPUTE_TRANSITIONS = ARGS.compute_transitions
LEARNING_RATE = ARGS.learning_rate
N_ITERS = ARGS.n_iters
MAX_SEQUENCE_LENGTH = ARGS.max_sequence
BATCH_SIZE = ARGS.batch_size

N_STATES = 0
N_STATE_FEATURES = 0
N_ACTIONS = 0
N_ACTION_FEATURES = 0

TRAINING_DATA_SIZE = 0

ERROR = None

states = None
actions = None
state_transitions = None
start_states_p = None
end_state_indices = None


def reduce_soft_max_condition(v_soft_prime, q_soft, rows):
  return tf.greater(rows, tf.constant(0))

def reduce_soft_max_body(v_soft_prime, q_soft, rows):
  global N_STATES

  v_s_inf = tf.constant(np.repeat(-np.inf, N_STATES), shape=[1,N_STATES], dtype=tf.float32)
  #v_s_zero = tf.constant(np.zeros(N_STATES), shape=[1,N_STATES], dtype=tf.float32)

  q_soft_slice, q_soft_rest = tf.cond(tf.greater(rows,tf.constant(1)), lambda: tf.split(q_soft,
[tf.constant(1), tf.constant(-1)], 0), lambda: [q_soft, tf.constant([], shape=[0,N_STATES],
dtype=tf.float32)])
  q_soft_slice.set_shape([1, q_soft.get_shape()[1]])
```

```
  current_max = tf.maximum(v_soft_prime, q_soft_slice)
  current_min = tf.minimum(v_soft_prime, q_soft_slice)

  diff = current_min - current_max
  diff_fix = tf.where(tf.is_nan(diff), v_s_inf, diff)

  soft_max_update = current_max + tf.log(1+tf.exp(diff_fix))
  soft_max_update_fixed = tf.where(tf.is_nan(soft_max_update), v_s_inf, soft_max_update)

  return [soft_max_update_fixed, q_soft_rest, rows-1]

#==============================================================================
  # Algorithm 9.1: state log partition function calculation.
  # Require: MDP, MMDP, and terminal state reward/potential function, f(s) -> R.
  # Ensure: state log partition functions, V_soft(s_x).
  # Changes: phi is now an indicator function with values of 1 for final states and 0 for all
  # other states.
  #          This is to improve the performance for the algorithm using sparse matrices.
  # Notes: For acyclic graphs lambda should be set to 1 and T to the length of the longest
  # possible sequence.
  #
  # Input:
  #   states - states features Tensor
  #   actions - action features Tensor
  #   state_transitions - |A|x|S|x|S| probability Tensor.
  #   end_state_incides - 1x|S| sparse end state indicator.
  #   theta - 1x(|Fs|+|Fa|) Tensor.
  #==============================================================================
def v_soft_condition(v_soft, q_soft, v_soft_error, v_soft_error_delta, theta, iter_n):
  global MAX_SEQUENCE_LENGTH
  #return tf.logical_and(iter_n < MAX_SEQUENCE_LENGTH,
tf.greater(tf.reduce_max(v_soft_error_delta), ERROR))
  return iter_n < MAX_SEQUENCE_LENGTH

def v_soft_body(v_soft, q_soft, v_soft_error, v_soft_error_delta, theta, iter_n):
  global end_state_indices

  # Mask for v_soft calculations.
  v_s_zero = tf.constant(np.zeros(N_STATES), shape=[1,N_STATES], dtype=tf.float32)
  v_s_inf = tf.constant(np.repeat(-np.inf, N_STATES), shape=[1,N_STATES], dtype=tf.float32)

  idx = tf.where(tf.greater_equal(v_soft, 0))

  # Create a Tensor for each action slice.
  q_a_soft_tensors = []
  for i in range(0,N_ACTIONS):
    action = tf.sparse_slice(actions, [i,0], [1,N_ACTION_FEATURES])
    action_state_transitions = tf.sparse_reduce_sum_sparse(tf.sparse_slice(state_transitions,
[i,0,0], [1,N_STATES,N_STATES]), axis=0)

    tile_action = tf.sparse_concat(0,[action] * N_STATES)
    states_action_features = tf.sparse_concat(1, [states, tile_action])

    # Set all features to 0 for state,actions pairs without a transition. This ensures reward
is 0 for those.
    mask = tf.sparse_reduce_sum_sparse(action_state_transitions, axis=1)
    tile_mask = tf.sparse_transpose(tf.sparse_reshape(tf.sparse_concat(0,[mask] *
(N_STATE_FEATURES+N_ACTION_FEATURES)), shape=[N_STATE_FEATURES+N_ACTION_FEATURES, N_STATES]))

    masked_states_action_features =
tf.multiply(tf.sparse_tensor_to_dense(states_action_features),
tf.sparse_tensor_to_dense(tile_mask))

    rewards = tf.matmul(theta, tf.transpose(masked_states_action_features), b_is_sparse=True)

    from_mask = tf.sparse_reduce_sum(action_state_transitions, axis=1)

    v_soft_no_inf = tf.where(tf.greater_equal(v_soft, v_s_zero), v_soft, v_s_zero)

    v_s_prime = tf.transpose(tf.sparse_tensor_dense_matmul(action_state_transitions,
tf.transpose(v_soft_no_inf)))

    v_s_prime_fix = tf.where(tf.greater(v_s_prime, v_s_zero), v_s_prime, v_s_inf)

    q_a_soft = rewards + v_s_prime_fix
```

```
    q_a_soft_tensors.append(q_a_soft)

  q_soft_prime = tf.reshape(tf.concat(tf.tuple(q_a_soft_tensors), axis=0), shape=q_soft.shape)

  n_actions = tf.constant(N_ACTIONS)
  reduce_soft_max = tf.while_loop(reduce_soft_max_condition, reduce_soft_max_body, [v_soft,
q_soft_prime, n_actions], shape_invariants=[v_soft.get_shape(), tf.TensorShape([None,
N_STATES]), n_actions.get_shape()])
  v_soft_prime = reduce_soft_max[0]

  error = tf.abs(v_soft_prime - v_soft)
  error_fixed = tf.where(tf.is_nan(error), v_s_zero, error)

  error_delta = tf.abs(error_fixed - v_soft_error)

  error_delta_fixed = tf.where(tf.is_nan(error_delta), v_s_zero, error_delta)

  return [v_soft_prime, q_soft_prime, error_fixed, error_delta_fixed, theta, tf.add(iter_n,
tf.constant(1, dtype=tf.int32))]

#=======
# Algorithm 9.3 Expected state frequency calculation
# Require: MDP, M_mdp, stochastic policy, p(a_x,y|s_x)), and initial state distribution
# P_0(s_x).
# Ensure: state visitation frequencies, Dsx under policy p(a_x,y|s_x).
def d_condition(d_sum, d_s, d_s_error, d_s_error_delta, state_action_policy, iter_n):
  global MAX_SEQUENCE_LENGTH
  return iter_n < MAX_SEQUENCE_LENGTH

def d_body(d_sum, d_s, d_s_error, d_s_error_delta, state_action_policy, iter_n):
  d_s_a_tensors = []

  for i in range(0,N_ACTIONS):
    action_policy = tf.slice(state_action_policy, [i,0], [1,N_STATES])
    action_state_transitions = tf.sparse_reduce_sum_sparse(tf.sparse_slice(state_transitions,
[i,0,0], [1,N_STATES,N_STATES]), axis=0)

    d_s_a =
tf.transpose(tf.sparse_tensor_dense_matmul(tf.sparse_transpose(action_state_transitions),
tf.transpose(tf.multiply(d_s, action_policy))))

    d_s_a_tensors.append(d_s_a)

  d_s_prime = tf.reshape(tf.reduce_sum(tf.concat(tf.tuple(d_s_a_tensors), axis=0), axis=0),
shape=[1,N_STATES])
  d_sum_prime = tf.add(d_sum, d_s_prime)

  error = tf.abs(d_sum_prime - d_sum)
  error_delta = tf.abs(error - d_s_error)

  return [d_sum_prime, d_s_prime, error, error_delta, state_action_policy, tf.add(iter_n,
tf.constant(1, dtype=tf.int32))]

#=======
# Algorithm 9.3 Expected state frequency calculation
# Require: MDP, M_mdp, stochastic policy, p(a_x,y|s_x)), and initial state distribution
P_0(s_x).
# Ensure: state visitation frequencies, Dsx under policy p(a_x,y|s_x).
def gradient_descent_condition(Ef_hat, theta, theta_error, theta_error_delta,
state_action_policy, state_transitions, d_s, iter_n):
  global N_ITERS

  return tf.logical_and(iter_n < N_ITERS, tf.greater(tf.reduce_max(theta_error_delta), ERROR))

def gradient_descent_body(Ef_hat, theta, theta_error, theta_error_delta, state_action_policy,
state_transitions, d_s, iter_n):
  global TRAINING_DATA_SIZE
  global start_states_p
  global end_state_indice
  global N_STATES


  empty_action_constant = tf.constant(np.repeat(0, N_ACTION_FEATURES),
shape=[1,N_ACTION_FEATURES], dtype=tf.float32)
  idx = tf.where(tf.not_equal(empty_action_constant, 0))
```

```
  empty_action = tf.SparseTensor(idx, tf.gather_nd(empty_action_constant, idx),
empty_action_constant.get_shape())

  tile_action = tf.sparse_concat(0,[empty_action] * N_STATES)
  states_action_features = tf.sparse_concat(1, [states, tile_action])

  rewards = tf.matmul(theta, tf.transpose(tf.sparse_tensor_to_dense(states_action_features)),
b_is_sparse=True)

  v_soft_initial = np.repeat(-np.inf, N_STATES)
  v_soft_initial[end_state_indices] = 0.0
  v_soft_const = tf.constant(v_soft_initial, shape=[1,N_STATES], dtype=tf.float32)

  v_soft = tf.where(tf.not_equal(v_soft_const, 0), v_soft_const, rewards)

  q_soft = tf.zeros(shape=[N_ACTIONS,N_STATES], dtype=tf.float32)
  q_soft_inf = tf.constant(np.repeat(-np.inf, N_ACTIONS*N_STATES), shape=[N_ACTIONS,N_STATES],
dtype=tf.float32)

  v_soft_error = tf.ones_like(v_soft)

  d_s_initial = tf.sparse_tensor_to_dense(start_states_p)
  d_s_error = tf.ones_like(d_s_initial)

  state_log_partition = tf.while_loop(v_soft_condition, v_soft_body, loop_vars=[v_soft, q_soft,
v_soft_error, v_soft_error, theta, tf.constant(0, dtype=tf.int32)])

  v_soft_prime = state_log_partition[0]
  q_soft_prime = state_log_partition[1]

  diff = q_soft_prime - v_soft_prime
  diff_fix = tf.where(tf.is_nan(diff), q_soft_inf, diff)
  state_action_policy_prime = tf.exp(diff_fix)

  state_visitation_frequencies = tf.while_loop(d_condition, d_body, [d_s_initial, d_s_initial,
d_s_error, d_s_error, state_action_policy_prime, tf.constant(0, dtype=tf.int32)])

  d_s = state_visitation_frequencies[0]
  d_s.set_shape([1,N_STATES])

  f_s = tf.transpose(tf.sparse_tensor_dense_matmul(tf.sparse_transpose(states),
tf.transpose(d_s)))
  f_a = tf.transpose(tf.sparse_tensor_dense_matmul(tf.sparse_transpose(actions),
tf.matmul(state_action_policy_prime, tf.transpose(d_s))))

  f_sa = tf.concat([f_s, f_a], axis=1)

  slice_n = tf.mod(iter_n, TRAINING_DATA_SIZE)
  Ef_hat_batch = tf.reshape(tf.slice(Ef_hat, [slice_n,0],
[1,N_STATE_FEATURES+N_ACTION_FEATURES]), shape=[1,N_STATE_FEATURES+N_ACTION_FEATURES])

  l = tf.divide(tf.cast(tf.constant(LEARNING_RATE), tf.float32),tf.cast(iter_n + 1,
tf.float32))
  theta_prime = tf.reshape(tf.multiply(theta, tf.exp(tf.multiply(l, (Ef_hat_batch - f_sa)))),
[1, N_STATE_FEATURES+N_ACTION_FEATURES])

  theta_prime_error = tf.abs(theta_prime - theta)

  theta_prime_error_delta = tf.abs(theta_prime_error - theta_error)

  return [Ef_hat, theta_prime, theta_prime_error, theta_prime_error_delta,
state_action_policy_prime, state_transitions, d_s, tf.add(iter_n, tf.constant(1,
dtype=tf.int32))]

def main():
  global N_STATES
  global N_STATE_FEATURES
  global N_ACTIONS
  global N_ACTION_FEATURES

  global STUDY_ID
  global TRAINING_DATA_SIZE
  global TRAINING_POPULATION_ID

  global ERROR
```

```
  global states
  global actions
  global state_transitions
  global start_states_p
  global end_state_indices

  db = MySQLdb.connect(host="archer.assist.cs.cmu.edu", user="human_routines",
passwd="human_routines2468!", db="human_routines")

  cursor = db.cursor()

  population_ids = load_populations(cursor, STUDY_ID, TRAINING_POPULATION_ID)

  #============================================================================
  # Create a run in the database.
  #============================================================================
  if RUN_ID == -1:
    cursor.execute("INSERT INTO `Run` (population_id, run_type_id) VALUES (%s, %s)",
[population_ids[0], 1])
    run_id = db.insert_id()
  else:
    run_id = RUN_ID

  if FOLD_ID == -1:
    cursor.execute("INSERT INTO `Fold` (run_id, fold_number) VALUES (%s,1);", [run_id])
    fold_id = db.insert_id()
  else:
    fold_id = FOLD_ID

  db.commit()

  #============================================================================
  # Load data.
  #============================================================================
  state_ids, states_idx, state_feature_ids, state_features_idx, states = load_states(cursor,
STUDY_ID)
  action_ids, actions_idx, action_feature_ids, action_features_idx, actions =
load_actions(cursor, STUDY_ID)

  N_STATES = len(state_ids)
  N_STATE_FEATURES = len(state_feature_ids)
  N_ACTIONS = len(action_ids)
  N_ACTION_FEATURES = len(action_feature_ids)

  state_transitions = load_state_transitions(cursor, STUDY_ID, FOLD_ID, state_ids, action_ids)

  end_state_ids, end_state_indices = load_end_states(cursor, STUDY_ID, state_ids)
  start_states_p = load_start_states(cursor, population_ids[0], state_ids)

  # Load sequences. The population should represent training data population.
  sequence_ids = load_sequences(cursor, population_ids[0])

  TRAINING_DATA_SIZE = len(sequence_ids)/BATCH_SIZE

  random.shuffle(sequence_ids)

  batch_state_counts, batch_action_counts = load_sequence_data(cursor, population_ids[0],
sequence_ids, state_ids, action_ids)


  #============================================================================
  # Initialize variables and placeholders used in the model.
  #============================================================================
  ERROR = tf.placeholder_with_default(0.0001, shape=[], name="ERROR")

  # This is what we are computing.
  theta = tf.constant(np.random.uniform(size=(N_STATE_FEATURES+N_ACTION_FEATURES,)),
shape=[1,N_STATE_FEATURES+N_ACTION_FEATURES], dtype=tf.float32)
  theta_error = tf.constant(np.repeat(np.inf, N_STATE_FEATURES+N_ACTION_FEATURES),
shape=[1,N_STATE_FEATURES+N_ACTION_FEATURES], dtype=tf.float32)
  theta_error_delta = tf.constant(np.repeat(np.inf, N_STATE_FEATURES+N_ACTION_FEATURES),
shape=[1,N_STATE_FEATURES+N_ACTION_FEATURES], dtype=tf.float32)

  # A placeholder. We want to get this from the gradient loop.
  state_action_policy = tf.zeros(shape=[N_ACTIONS,N_STATES], dtype=tf.float32)
```

```
  Ef_states = tf.transpose(tf.sparse_tensor_dense_matmul(tf.sparse_transpose(states),
tf.transpose(batch_state_counts)))
  Ef_actions = tf.transpose(tf.sparse_tensor_dense_matmul(tf.sparse_transpose(actions),
tf.transpose(batch_action_counts)))
  Ef_hat = tf.concat([Ef_states, Ef_actions], axis=1)

  d_s_placeholder = tf.zeros_like(tf.sparse_tensor_to_dense(start_states_p))

  stochastic_gradient_descent = tf.while_loop(gradient_descent_condition,
gradient_descent_body, [Ef_hat, theta, theta_error, theta_error_delta, state_action_policy,
state_transitions, d_s_placeholder, tf.constant(0, dtype=tf.int32)])

  #================================================================================
  # Data loaded. Graph created. Signal run start.
  #================================================================================
  cursor.execute("UPDATE `Run` SET updated_timestamp = NOW() WHERE id = %s;", [run_id])
  db.commit()

  cursor.close()
  db.close()

  init = tf.global_variables_initializer()

#   sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
  sess = tf.Session()
  with sess.as_default():
    sess.run(init)
    result = sess.run(stochastic_gradient_descent)
    final_theta = result[1][0]
    final_policy = result[4]
    final_d_s = result[6][0]

    fold_theta = []
    for i in range(0,len(final_theta)):
      feature_id = -1
      if i < N_STATE_FEATURES:
        feature_id = state_features_idx[i]
      elif i < N_STATE_FEATURES+N_ACTION_FEATURES:
        feature_id = action_features_idx[i-N_STATE_FEATURES]
      else:
        pass

      fold_theta.append((fold_id, feature_id, final_theta[i],))

    db = MySQLdb.connect(host="archer.assist.cs.cmu.edu", user="human_routines",
passwd="human_routines2468!", db="human_routines", compress=True)
    cursor = db.cursor()

    cursor.executemany("INSERT INTO `Fold_Feature_Theta` (fold_id, feature_id, theta) VALUES
(%s, %s, %s);", fold_theta)
    db.commit()

    cursor.execute("INSERT INTO `Policy` (fold_id, policy_type_id, algorithm_id) VALUES (%s, 1,
1);", [fold_id])
    policy_id = db.insert_id()

    fold_policy = []
    fold_policy_count = 0
    for state_idx in range(0,N_STATES):
      for action_idx in range(0, N_ACTIONS):
        p = final_policy[action_idx][state_idx]

        if p > 0.0:
          state_id = states_idx[state_idx]
          action_id = actions_idx[action_idx]
          fold_policy.append((policy_id, state_id, action_id, p,))
          fold_policy_count = fold_policy_count + 1
          if fold_policy_count == 100000:
            cursor.executemany("INSERT INTO `State_Action_Policy` (policy_id, state_id,
action_id, probability) VALUES (%s, %s, %s, %s);", fold_policy)
            fold_policy = []
            fold_policy_count = 0
    if fold_policy_count > 0:
      cursor.executemany("INSERT INTO `State_Action_Policy` (policy_id, state_id, action_id,
probability) VALUES (%s, %s, %s, %s);", fold_policy)
```

```
    db.commit()

    fold_d_s = []
    for i in range(0,len(final_d_s)):
      fold_d_s.append((fold_id, states_idx[i], final_d_s[i],))

    cursor.executemany("INSERT INTO `Fold_State_Counts` (fold_id, state_id, expected_count)
VALUES (%s, %s, %s);", fold_d_s)
    db.commit()

    cursor.execute("UPDATE `Run` SET updated_timestamp = NOW(), completed_timestamp = NOW()
WHERE id = %s;", [run_id])
    db.commit()

    cursor.close()
    db.close()

# Load study populations.
def load_populations(cursor, study_id, population_id=None):
  if population_id is None:
    cursor.execute("SELECT id, population_type_id FROM Population WHERE study_id = %s;",
[study_id])
  else:
    cursor.execute("SELECT id, population_type_id FROM Population WHERE study_id = %s AND
Population.id = %s;", [study_id, population_id])

  population_ids = [row[0] for row in cursor]

  return population_ids

# Initializes states matrix.
def load_states(cursor, study_id):
  cursor.execute("SELECT count(*), max(State.feature_count) FROM State WHERE State.study_id =
%s ORDER BY State.id;", [study_id])
  state_dims = list(cursor.fetchone())

  feature_ids = {}
  feature_idx = {}
  i = 0
  cursor.execute("SELECT id FROM Feature WHERE study_id = %s AND Feature.is_in_model = 1 AND
feature_type_id = 1 ORDER BY feature_index;", [study_id])
  for row in cursor:
    feature_id = row[0]
    feature_ids[feature_id] = i
    feature_idx[i] = feature_id
    i += 1

  state_ids = {}
  state_idx = {}
  i = 0
  cursor.execute("SELECT State.id FROM State WHERE State.study_id = %s ORDER BY State.id;",
[study_id])
  for row in cursor:
    state_id = row[0]
    state_ids[state_id] = i
    state_idx[i] = state_id
    i += 1

  indices = []
  values = []

  cursor.execute("SELECT State.id, Feature.id AS feature_id, State_Feature.feature_value FROM
State INNER JOIN State_Feature ON (State.id = State_Feature.state_id) INNER JOIN Feature ON
(Feature.id = State_Feature.feature_id) WHERE State.study_id = %s AND Feature.is_in_model = 1
ORDER BY State.id, feature_index;", [study_id])
  for row in cursor:
    state_id = row[0]
    feature_id = row[1]
    feature_value = row[2]

    indices.append([state_ids[state_id], feature_ids[feature_id]])
    values.append(feature_value)

  states = tf.SparseTensor(indices=indices, values=tf.cast(values, tf.float32),
dense_shape=state_dims)
```

```
  return state_ids, state_idx, feature_ids, feature_idx, states

def load_state_transitions(cursor, study_id, fold_id, state_ids, action_ids):
  indices = []
  values = []

  if fold_id == -1:
    cursor.execute("SELECT from_state_id, action_id, to_state_id, probability FROM
State_Transition INNER JOIN State ON (State_Transition.`from_state_id` = State.id) WHERE
State.study_id = %s ORDER BY action_id, from_state_id, to_state_id;", [study_id])
  else:
    cursor.execute("SELECT from_state_id, action_id, to_state_id, probability FROM
Fold_State_Transition WHERE fold_id = %s ORDER BY action_id, from_state_id, to_state_id;",
[fold_id])

  for row in cursor:
    from_state_id = row[0]
    action_id = row[1]
    to_state_id = row[2]
    probability = row[3]

    indices.append([action_ids[action_id], state_ids[from_state_id], state_ids[to_state_id]])

    values.append(probability)

  transitions = tf.SparseTensor(indices=indices, values=tf.cast(values, tf.float32),
dense_shape=[len(action_ids), len(state_ids), len(state_ids)])

  return transitions

# Initialize action vector.
def load_actions(cursor, study_id):
  cursor.execute("SELECT count(*), max(Action.feature_count) FROM Action WHERE Action.study_id
= %s ORDER BY Action.id;", [study_id])
  action_dims = list(cursor.fetchone())

  feature_ids = {}
  feature_idx = {}
  i = 0
  cursor.execute("SELECT id FROM Feature WHERE study_id = %s AND Feature.is_in_model = 1 AND
feature_type_id = 2 ORDER BY feature_index;", [study_id])
  for row in cursor:
    feature_id = row[0]
    feature_ids[feature_id] = i
    feature_idx[i] = feature_id
    i += 1

  action_ids = {}
  action_idx = {}
  i = 0
  cursor.execute("SELECT Action.id FROM Action WHERE Action.study_id = %s ORDER BY Action.id;",
[study_id])
  for row in cursor:
    action_id = row[0]
    action_ids[action_id] = i
    action_idx[i] = action_id
    i += 1

  indices = []
  values = []

  cursor.execute("SELECT Action.id, Feature.id AS feature_id, Action_Feature.feature_value FROM
Action INNER JOIN Action_Feature ON (Action.id = Action_Feature.action_id) INNER JOIN Feature
ON (Feature.id = Action_Feature.feature_id) WHERE Action.study_id = %s AND Feature.is_in_model
= 1 ORDER BY Action.id, feature_index;", [study_id])
  for row in cursor:
    action_id = row[0]
    feature_id = row[1]
    feature_value = row[2]

    indices.append([action_ids[action_id], feature_ids[feature_id]])
    values.append(feature_value)

  actions = tf.SparseTensor(indices=indices, values=tf.cast(values, tf.float32),
dense_shape=action_dims)
```

```
  return action_ids, action_idx, feature_ids, feature_idx, actions

def load_sequences(cursor, population_id):
  sequence_ids = []
  cursor.execute("SELECT Sequence.id FROM Sequence INNER JOIN Participant ON
(Sequence.participant_id = Participant.id) INNER JOIN Participant_Population ON (Participant.id
= Participant_Population.participant_id) INNER JOIN Population ON
(Participant_Population.population_id = Population.id) WHERE Population.id = %s;",
[population_id])
  for row in cursor:
    sequence_ids.append(int(row[0]))

  return sequence_ids

# Initialize sequences.
def load_sequence_data(cursor, population_id, sequence_ids, state_ids, action_ids):
  global TRAINING_DATA_SIZE

  if TRAINING_DATA_SIZE == 0:
    sequence_ids_batches = np.array(sequence_ids, ndmin=2)
    TRAINING_DATA_SIZE = 1
  else:
    sequence_ids_batches = np.array_split(np.array(sequence_ids), TRAINING_DATA_SIZE)

  batch_state_counts = None
  batch_action_counts = None

  for sequence_id_batch in sequence_ids_batches:
    sequence_state_counts = np.zeros(N_STATES)
    sequence_action_counts = np.zeros(N_ACTIONS)

    batch_ids_placeholder = ', '.join(itertools.repeat('%s', len(sequence_id_batch)))
    sql = "SELECT from_state_id, action_id, count FROM Sequence_Transition INNER JOIN Sequence
ON (Sequence.id = Sequence_Transition.sequence_id) INNER JOIN Participant ON
(Sequence.participant_id = Participant.id) INNER JOIN Participant_Population ON (Participant.id
= Participant_Population.participant_id) INNER JOIN Population ON
(Participant_Population.population_id = Population.id) WHERE Population.id = %s AND sequence_id
IN (%s) ORDER BY Sequence.participant_id, sequence_id;" % ('%s', batch_ids_placeholder)
    cursor.execute(sql, [population_id] + sequence_id_batch.tolist())
    for row in cursor:
      from_state_id = row[0]
      action_id = row[1]
      count = row[2]

      from_state_idx = state_ids[from_state_id]
      action_idx = action_ids[action_id]

      sequence_state_counts[from_state_idx] = sequence_state_counts[from_state_idx] + count
      sequence_action_counts[action_idx] = sequence_action_counts[action_idx] + count

    # Add last sequence batch.
    if batch_state_counts is None:
      batch_state_counts = np.array(sequence_state_counts/len(sequence_id_batch), ndmin=2)
    else:
      batch_state_counts = np.append(batch_state_counts,
np.array(sequence_state_counts/len(sequence_id_batch), ndmin=2), axis=0)

    if batch_action_counts is None:
      batch_action_counts = np.array(sequence_action_counts/len(sequence_id_batch), ndmin=2)
    else:
      batch_action_counts = np.append(batch_action_counts,
np.array(sequence_action_counts/len(sequence_id_batch), ndmin=2), axis=0)

  return tf.constant(batch_state_counts, dtype=tf.float32, shape=[TRAINING_DATA_SIZE,
N_STATES]), tf.constant(batch_action_counts, dtype=tf.float32, shape=[TRAINING_DATA_SIZE,
N_ACTIONS])

def load_start_states(cursor, population_id, state_ids):
  indices = []
  values = []

  cursor.execute("SELECT start_state_id, avg(probability) AS count FROM Initial_State INNER
JOIN Participant_Population ON (Initial_State.participant_id =
Participant_Population.participant_id) WHERE population_id = %s GROUP BY start_state_id;",
[population_id])
  for row in cursor:
```

```
    state_id = row[0]
    p = row[1]

    indices.append([0,state_ids[state_id]])
    values.append(p)

  start_state_probabilities =  tf.SparseTensor(indices=indices, values=tf.cast(values,
tf.float32), dense_shape=[1,len(state_ids)])

  return start_state_probabilities

def load_end_states(cursor, study_id, state_ids):
  end_state_ids = []
  end_state_indices = []

  cursor.execute("SELECT state_id FROM End_State WHERE study_id =  %s ORDER BY state_id;",
[study_id])
  for row in cursor:
    state_id = row[0]

    end_state_ids.append(state_id)
    end_state_indices.append(state_ids[state_id])

  return end_state_ids, end_state_indices

if __name__ == "__main__":
  main()
```

10.2 Study Materials

10.2.1 Visual Model Validation Study Questionnaire

Thank you for taking part in the study. In this study you will complete a questionnaire and then answer questions about two tasks related to human routine behavior.

Please answer the following:
Participant #:
Age:
Occupation:
Experience with (place X next to the ones you have experience with):
Machine Learning
Data Mining
Activity Recognition
Human Routine Behavior
Other relevant (please specify):

Thank you. Please move on to the next page.

Task 1
In this task you will explore driving routine behavior of two populations of drivers: non-aggressive and aggressive drivers. You will explore their driving behavior when navigating intersections in 4 stages: 1) approaching intersection, 2) just before entering intersection, 3) leaving intersection, and 4) driving away from the intersection.

Please answer the following questions in order:

Please describe the driving routine of non-aggressive drivers when going straight through an intersection. Please describe at least one deviation from the main routine. Note that there might be more than one routine that describes this behavior.

Please describe the driving routine of aggressive drivers when going straight through an intersection. Please describe at least one deviation from the main routine. Note that there might be more than one routine that describes this behavior.

Based on your findings, please describe the differences between the routines of non-aggressive drivers and aggressive drivers.

Task 2
In this task you will identify a person's daily routine for a given day.

1. Please describe the person's routine for WEDNESDAYS.

2. Please describe this person's likely deviation on MONDAY.

Thank you for your time! Please let us know if you have any last comments.

10.2.2 Driving Instructor Detection and Simulation Validation Questionnaire

This is a two-page excerpt from Driving Instructor Detection and Simulation Questionnaire illustrating the questions instructors answered in the study.

DriveCap: Naturalistic Driver Data (Driving Instructors)

* Required

1. **Participant #** *

2. **Date** *

   Example: December 15, 2012

3. **Time** *

   Example: 8:30 AM

4. **Driver code**

Biographical Information

Please tell us about yourself.

5. **Age** *

6. **Gender** *

7. **How long have you been driving?** *

8. **How long have you been a driving instructor?** *

Driving Animations Task 1

In this task you will review a series of driving scenarios. You will be asked if each scenario represents aggressive driving or non-aggressive driving.

Warm-up Scenarios

You will complete 5 warm-up scenarios. Please complete questions about the following scenarios. After each scenario click continue to move to the next scenario.

Scenario 1

Please use the tablet to explore the scenario and answer the questions below. You can replay the scenario as many times as you want.

9. **Is the driver aggressive, neutral, or non-aggressive?** *
   Mark only one oval.

	1	2	3	
Aggressive	◯	◯	◯	Non-aggressive

Scenario 2

Please use the tablet to explore the scenario and answer the questions below. You can replay the scenario as many times as you want.

10. **Is the driver aggressive, neutral, or non-aggressive?** *
    Mark only one oval.

	1	2	3	
Aggressive	◯	◯	◯	Non-aggressive

Scenario 3

Please use the tablet to explore the scenario and answer the questions below. You can replay the scenario as many times as you want.

11. **Is the driver aggressive, neutral, or non-aggressive?** *
    Mark only one oval.

	1	2	3	
Aggressive	◯	◯	◯	Non-aggressive

Scenario 4

Please use the tablet to explore the scenario and answer the questions below. You can replay the scenario as many times as you want.

12. **Is the driver aggressive, neutral, or non-aggressive?** *
    Mark only one oval.

	1	2	3	
Aggressive	◯	◯	◯	Non-aggressive

Scenario 5

Please use the tablet to explore the scenario and answer the questions below. You can replay the scenario as many times as you want.

13. **Is the driver aggressive, neutral, or non-aggressive?** *
    Mark only one oval.

	1	2	3	
Aggressive	◯	◯	◯	Non-aggressive

10.2.3 Aggressive Driver Assessment Study Modified DBQ Questionnaire

This is a three-page excerpt from Driver Assessment study DBQ illustrating the questions participants answered in the study.

DriveCap: Naturalistic Driving (Drivers)

* Required

1. **Participant #** *

2. **Condition** *
   Mark only one oval.
   - All data without non-aggressive simulation (Baseline)
   - Aggressive data with non-aggressive simulation

3. **Date** *
   Example: December 15, 2012

4. **Time** *
   Example: 8:30 AM

Biographical Information

Please tell us about yourself.

5. **Age** *

6. **Gender** *

7. **How long have you been driving?** *

8. **How often do you drive each week?** *

9. **How long is your average daily trip (e.g., to work)?** *

Your Driving Behaviors

Please answer the following questions about your driving expertise and quality.

10. **How would you rate your driving expertise?** *
Mark only one oval.

	1	2	3	4	5	6	
Very inexperienced	◯	◯	◯	◯	◯	◯	Very experienced

11. **How would you rate your driving quality?** *
Mark only one oval.

	1	2	3	4	5	6	
Very aggressive	◯	◯	◯	◯	◯	◯	Very non-aggressive

Driving Behaviors
In the following section, please rate how often you do the driving behaviors below, and what you think about that behavior in terms of driving expertise and quality.

Driving Behaviors

12. **How often do you check the speedometer and discover that the car is unknowingly traveling faster than the legal limit?** *
Mark only one oval.

- ◯ Never
- ◯ Hardly ever
- ◯ Occasionally
- ◯ Quite often
- ◯ Frequently
- ◯ Nearly all the time

13. **How would you rate your driving expertise in regards to this behavior and frequency?** *
Mark only one oval.

	1	2	3	4	5	6	
Very inexperienced	◯	◯	◯	◯	◯	◯	Very experienced

14. **How would you rate your driving quality in regards to this behavior and frequency?** *
Mark only one oval.

	1	2	3	4	5	6	
Very aggressive	◯	◯	◯	◯	◯	◯	Very non-aggressive

Driving Behaviors

15. **How often do you become impatient with a slow driver in the outer lane and overtakes on the inside?** *
Mark only one oval.

- Never
- Hardly ever
- Occasionally
- Quite often
- Frequently
- Nearly all the time

16. **How would you rate your driving expertise in regards to this behavior and frequency?** *
Mark only one oval.

	1	2	3	4	5	6	
Very inexperienced							Very experienced

17. **How would you rate your driving quality in regards to this behavior and frequency?** *
Mark only one oval.

	1	2	3	4	5	6	
Very aggressive							Very non-aggressive

Driving Behaviors

18. **How often do you drive especially close or "flash" the car in front as a signal for that driver to go faster or get out of the way?** *
Mark only one oval.

- Never
- Hardly ever
- Occasionally
- Quite often
- Frequently
- Nearly all the time

19. **How would you rate your driving expertise in regards to this behavior and frequency?** *
Mark only one oval.

	1	2	3	4	5	6	
Very inexperienced							Very experienced

10.3Behavior Dashboard Design Materials

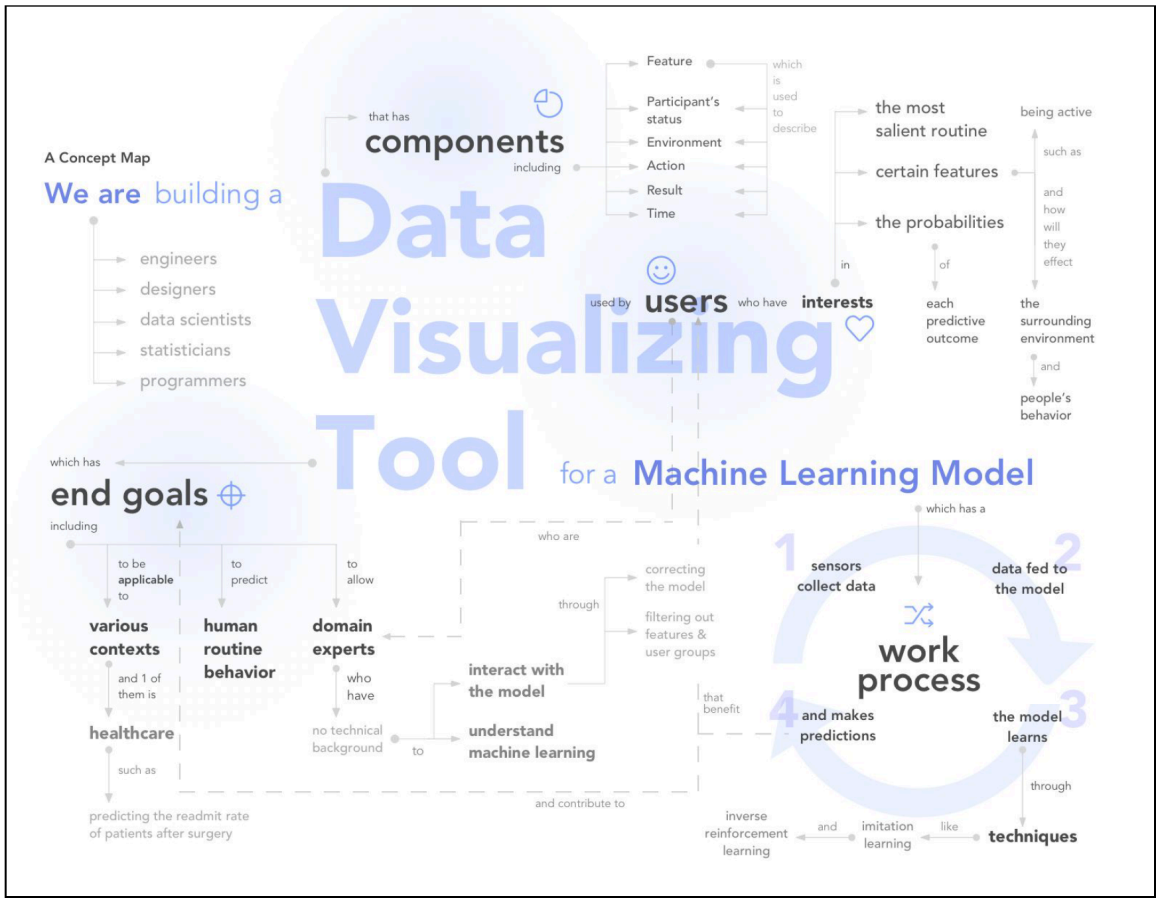

Figure 21. Concept Map showing concepts we have identified during our design process that have influenced our design decisions in creating Behavior Dashboard.

Primary Persona: Travis Foster

Age 57
Occupation Doctor at UPMC
Technology

Bio

Dr. Foster is an expert at cardiovascular field. Knowing the potential of artificial intelligence, he wants to use machine learning model to predict the outcomes of patients with heart disease for chronic disease management, potentially lowering costs and improving outcomes for patients with longer-term needs.

Dr. Foster has no previous knowledge with machine learning and he expects the tool to be as simple and intuitive as possible.

Data

Patient status (Situation):

- Demographic – age, gender, race, occupation,
- Health – admissions, vital signs, hereditary disease, diagnoses, patients' reaction, mood
- DNA makeup – the whole genome of 46 chromosome
- Others – lab results

Treatment (Action):

- Medications prescribed
- Concentration and dosage of the drug
- Treatment procedures
- Activity level

Goal

1. **To find the general trend of what kind of patients are most likely to be hospitalized**
2. **To get the predicted results of individual patient.**
   - What's the prediction of John's outcome?
3. **To identify what are the possible outcomes of the population and the likelihood of them.**
4. **To leverage his knowledge and the model to find out the best intervention.**
   - What can we do to help, based on existing data from other similar patients?
   - What if we increase the amount of exercise everyday? What will be the chance of re-hospitalization after a month?
   - Filter/subdivide features
     - e.g. What are the trend for males and for females?
     - What kind of patients will experience the most optimized outcome? What do they have in common?
5. **To find out the existing outliers.**
   - Who are the patients that had great improvement without proper treatment? What's special about them?

Figure 22. A primary persona we identified through our design process. This imaginary user allows us to consider user needs and requirements of actual people we design Behavior Dashboard for.

BIBLIOGRAPHY

AAAFoundation.org, Foundation for Traffic Safety, & AAAFoundation.org. (2009). *Aggressive Driving: Research Update*. *American Automobile Association Foundation*. Retrieved from https://scholar.google.com/scholar?q=Aggressive+driving%3A+Research+update&btnG=&hl=en&as_sdt=0%2C39

Abdul, A., Vermeulen, J., Wang, D., Lim, B. Y., & Kankanhalli, M. (2018). Trends and Trajectories for Explainable, Accountable and Intelligible Systems: An HCI Research Agenda. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, 1–18. http://doi.org/10.1145/3173574.3174156

Adar, E., Teevan, J., & Dumais, S. T. (2008). Large scale analysis of web revisitation patterns. *Proceeding of the Twenty-Sixth Annual CHI Conference on Human Factors in Computing Systems - CHI '08*, 1197. http://doi.org/10.1145/1357054.1357241

Agre, P. E., & Shrager, J. (1990). Routine Evolution as the Microgenetic Basis of Skill Acquistion. In *Twelfth Annual Conference of the Cognitive Science Society* (p. 694701). Retrieved from http://www.citeulike.org/group/4917/article/2623697

Aigner, W., Miksch, S., Thurnher, B., & Biffl, S. (2005). PlanningLines: Novel glyphs for representing temporal uncertainties and their evaluation. In *Proceedings of the International Conference on Information Visualisation* (Vol. 2005, pp. 457–463). http://doi.org/10.1109/IV.2005.97

Ajzen, I. (1985). From intentions to actions: A theory of planned behavior. *Action Control*. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-69746-3_2

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, *50*(2), 179–211. http://doi.org/10.1016/0749-5978(91)90020-T

Banovic, N., Brant, C., Mankoff, J., & Dey, A. K. (2014). ProactiveTasks : the Short of Mobile Device Use Sessions. *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services - MobileHCI '14*, 243–252. http://doi.org/10.1145/2628363.2628380

Banovic, N., Rao, V., Saravanan, A., Dey, A. K., & Mankoff, J. (2017). Quantifying Aversion to Costly Typing Errors in Expert Mobile Text Entry. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*.

Baratchi, M., Meratnia, N., Havinga, P. J. M., Skidmore, A. K., & Toxopeus, B. a. K. G. (2014). A hierarchical hidden semi-Markov model for modeling mobility data. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct*, 401–412. http://doi.org/10.1145/2632048.2636068

Becker, M. C. (2004). Organizational routines: A review of the literature. *Industrial and Corporate Change*, *13*(4), 643–677. http://doi.org/10.1093/icc/dth026

Bellman, R. (1957). A Markovian decision process. *Journal Of Mathematics And Mechanics*. http://doi.org/10.1007/BF02935461

Beyer, H., & Holtzblatt, K. (1998). *Contextual design : defining customer-centered systems*. Morgan Kaufmann. Retrieved from https://dl.acm.org/citation.cfm?id=2821566

Bishop, C. (2006). Pattern recognition. *Machine Learning*. Retrieved from http://www.academia.edu/download/30428242/bg0137.pdf

Brdiczka, O., Su, N., & Begole, J. (2010). Temporal task footprinting: identifying routine tasks by their temporal patterns. *Of the 15th International Conference on …*. Retrieved from http://dl.acm.org/citation.cfm?id=1720011

Breiman, L. (2001). Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, *16*(3), 199–231. http://doi.org/10.1214/ss/1009213726

Bulling, A., Blanke, U., & Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, *1*(June), 1–33. http://doi.org/http://dx.doi.org/10.1145/2499621

Buono, P., Aris, A., Plaisant, C., Khella, A., & Shneiderman, B. (2005). Interactive Pattern Search in Time Series. In *Proceedings of the Conference on Visualization and Data Analysis (VDA 2005)* (Vol. 5669, pp. 175–186). http://doi.org/10.1117/12.587537

Buthpitiya, S., Dey, A., & Griss, M. (2014). Soft authentication with low-cost signatures. *Pervasive Computing And*. Retrieved from http://ieeexplore.ieee.org/abstract/document/6813958/

Cakmak, M., & Thomaz, A. (2011). Mixed-initiative active learning. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.352.2122

Capra, R. (2011). HCI browser: A tool for administration and data collection for studies of web search behaviors. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 6770 LNCS, pp. 259–268). http://doi.org/10.1007/978-3-642-21708-1_30

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of HCI. Lawrence Erlbaum Associates Inc. Publishers*. of {HCI}.

Casarrubea, M., Jonsson, G. K., Faulisi, F., Sorbera, F., Di Giovanni, G., Benigno, A., & Crescimanno, G. (2015). T-pattern analysis for the study of temporal structure of animal and human behavior: A comprehensive review. *Journal of Neuroscience Methods*, *239*, 34–46. http://doi.org/10.1016/j.jneumeth.2014.09.024

Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-Supervised Learning. *IEEE Transactions on Neural Networks*, *20*(3), 542. http://doi.org/10.1109/TNN.2009.2015974

Clear, A. K., Shannon, R., Holland, T., Quigley, A., Dobson, S., & Nixon, P. (2009). Situvis: A visual tool for modeling a user's behaviour patterns in a pervasive environment. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5538 LNCS, pp. 327–341). http://doi.org/10.1007/978-3-642-01516-8_22

Cook, S., Conrad, C., Fowlkes, A. L., & Mohebbi, M. H. (2011). Assessing Google Flu trends performance in the United States during the 2009 influenza virus A (H1N1) pandemic. *PLoS ONE*, *6*(8), e23610. http://doi.org/10.1371/journal.pone.0023610

Davidoff, S. (2010). Routine as resource for the design of learning systems. *Proceedings of the 12th ACM International Conference ...*, (May). Retrieved from http://dl.acm.org/citation.cfm?id=1864486

Davidoff, S., Ziebart, B. D., Zimmerman, J., & Dey, A. K. (2011). Learning Patterns of Pick-ups and Drop-offs to Support Busy Family Coordination. *The ACM CHI Conference on Human Factors*, 1175–1184. http://doi.org/10.1145/1978942.1979119

Davidoff, S., Zimmerman, J., & Dey, A. K. (2010). How routine learners can support family coordination. *Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI '10*, *4*, 2461. http://doi.org/10.1145/1753326.1753699

Dey, A. (2001). Understanding and using context. *Personal and Ubiquitous Computing*. Retrieved from http://link.springer.com/article/10.1007/s007790170019

Dumais, S., Jeffries, R., Russell, D. M., Tang, D., & Teevan, J. (2014). Understanding User Behavior Through Log Data and Analysis. In *Ways of Knowing in HCI* (pp. 349–372). http://doi.org/10.1007/978-1-4939-0378-8_14

Eagle, N., & Pentland, A. S. (2009). Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, *63*(7), 1057–1066. http://doi.org/10.1007/s00265-009-0739-0

Farrahi, K., & Gatica-Perez, D. (2012). Extracting mobile behavioral patterns with the distant N-gram topic model. In *Proceedings - International Symposium on Wearable Computers, ISWC* (pp. 1–8). http://doi.org/10.1109/ISWC.2012.20

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, *17*(3), 37. http://doi.org/10.1609/aimag.v17i3.1230

Fekete, J.-D., Wijk, J. J. Van, Stasko, J. T., & North, C. (2008). The Value of Information Visualization. *Information Visualization*, *4950*(2), 1–18. http://doi.org/10.1007/978-

3-540-70956-5_1

Feldman, M. S. (2000). Organizational Routines as a Source of Continuous Change. *Organization Science*, *11*(6), 611–629. http://doi.org/10.1287/orsc.11.6.611.12529

Feldman, M. S., & Pentland, B. T. (2003). Reconceptualizing Organizational Routines as a Source of Flexibility and Change. *Administrative Science Quarterly*, *48*(1), 94–118. http://doi.org/10.2307/3556620

Ferreira, D., Kostakos, V., & Dey, A. K. (2015). AWARE: Mobile Context Instrumentation Framework. *Frontiers in ICT*, *2*(April), 1–9. http://doi.org/10.3389/fict.2015.00006

Gaver, B., Dunne, T., & Pacenti, E. (1999). Design: Cultural probes. *Interactions*, *6*(1), 21–29. http://doi.org/10.1145/291224.291235

González, D., Pérez, J., Milanés, V., & Nashashibi, F. (2016). A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *17*(4), 1135–1145. http://doi.org/10.1109/TITS.2015.2498841

Good, I. J. (1983). The Philosophy of Exploratory Data Analysis. *Philosophy of Science*, *50*(2), 283–295. http://doi.org/10.1086/289110

Google. (2017). Google Analytics. Retrieved April 10, 2017, from http://www.google.com/ analytics/

Gray, C., Kou, Y., Battles, B., Hoggatt, J., & Toombs, A. (2018). The Dark (Patterns) Side of UX Design. *Chi'2018*, (April), 1–14. http://doi.org/10.1145/3173574.3174108

Hamermesh, D. S. (2003). Routine. *NBER Working Paper Series*, (9440). Retrieved from http://www.sciencedirect.com/science/article/pii/S0014292104000182

Hodgson, G. M. (1997). The ubiquity of habit and rules. *Cambridge Journal of Economics*, *21*(6), 663–683.

Hodgson, G. M. (2009). Choice, habit and evolution. *Journal of Evolutionary Economics*, *20*(1), 1–18. http://doi.org/10.1007/s00191-009-0134-z

Hong, J.-H., Margines, B., & Dey, A. K. (2014). A smartphone-based sensing platform to model aggressive driving behaviors. *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI '14*, 4047–4056. http://doi.org/10.1145/2556288.2557321

Hurst, A., Mankoff, J., & Hudson, S. E. (2008). Understanding pointing problems in real world computing environments. *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, (1), 43–50. http://doi.org/10.1145/1414471.1414481

Ivanov, Y. (2001). Expectation maximization for weakly labeled data. *MACHINE LEARNING-* .... Retrieved from http://alumni.media.mit.edu/~yivanov/Papers/ICML01/icml2001.pdf.gz

Jaynes, E. T. (1955). Information Theory and Statistical Mechanics. *Physical Review*. http://doi.org/10.1103/PhysRev.108.171

Jin, J., & Szekely, P. (2010). Interactive querying of temporal data using a comic strip metaphor. In *VAST 10 - IEEE Conference on Visual Analytics Science and Technology 2010, Proceedings* (pp. 163–170). http://doi.org/10.1109/VAST.2010.5652890

Jordan, M. I., & Mitchell, T. M. (2015, July 17). Machine learning: Trends, perspectives, and prospects. *Science*. American Association for the Advancement of Science. http://doi.org/10.1126/science.aaa8415

Kahneman, D. (2011). *Thinking, fast and slow*. Retrieved from https://books.google.com/books?hl=en&lr=&id=SHvzzuCnuv8C&oi=fnd&pg=PP2&dq=Thinking,+Fast+and+Slow&ots=NRxfPG2gIF&sig=XabIk-qqShDTUWZ_4S_1Bzx2kB8

Keim, D., Andrienko, G., Fekete, J. D., Görg, C., Kohlhammer, J., & Melançon, G. (2008). Visual analytics: Definition, process, and challenges. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 4950 LNCS, pp. 154–175). http://doi.org/10.1007/978-3-540-70956-5_7

Kleinberg, J., Ludwig, J., Mullainathan, S., & Obermeyer, Z. (2015). Prediction Policy Problems. *American Economic Review: Papers & Proceedings*, *105*(5), 491–495. http://doi.org/10.1257/aer.p20151023

Koehler, C., Banovic, N., Oakley, I., Mankoff, J., & Dey, A. K. (2014). Indoor-ALPS: An adaptive indoor location prediction system. *UbiComp 2014 - Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 171–181. http://doi.org/10.1145/2632048.2632069

Krumm, J., & Horvitz, E. (2006). Predestination: Inferring destinations from partial trajectories. In *UbiComp'06* (pp. 243–260). http://doi.org/10.1007/11853565_15

Kuutti, K. (1995). Activity Theory as a potential framework for human- computer interaction research. *Context and Consciousness: Activity Theory and Human-Computer Interaction*, 17–44. http://doi.org/citeulike-article-id:634717

Lazer, D., Kennedy, R., King, G., & Vespignani, A. (2014). 29113Ad5-B65E-4E09-8445-E5Dd6C792C5a, *343*(March), 1203–1205.

Li, N., Kambhampati, S., & Yoon, S. (2009). Learning Probabilistic Hierarchical Task Networks to Capture User Preferences. *IJCAI*. Retrieved from http://www.aaai.org/ocs/index.php/IJCAI/IJCAI-09/paper/download/417/874

Low, C. A., Bovbjerg, D. H., Ahrendt, S., Haroon Choudry, M., Holtzman, M., Jones, H. L., … Bartlett, D. L. (2018). Fitbit step counts during inpatient recovery from cancer surgery as a predictor of readmission. In *Annals of Behavioral Medicine* (Vol. 52, pp. 88–92). Oxford University Press. http://doi.org/10.1093/abm/kax022

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, *7*(1), 48. http://doi.org/10.1207/s15327051hci0701_3

Magnusson, M. S. (2000). Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods, Instruments, & Computers : A Journal of the Psychonomic Society, Inc*, *32*(1), 93–110. http://doi.org/10.3758/Bf03200792

Mann, G. S. G., & Mccallum, A. (2010). Generalized Expectation Criteria for Semi-Supervised Learning with Weakly Labeled Data. *JMLR*, *11*, 955–984. Retrieved from http://www.jmlr.org/papers/v11/mann10a.html

Mcfowland Iii, E., Speakman, S., & Neill, D. B. (2013). Fast Generalized Subset Scan for Anomalous Pattern Detection. *Journal of Machine Learning Research*, *14*, 1533–1561. Retrieved from http://www.jmlr.org/papers/volume14/mcfowland13a/mcfowland13a.pdf

Melnik, R. (2015). *Mathematical and Computational Modeling: With Applications in Natural and Social Sciences, Engineering, and the Arts*. Retrieved from https://www.wiley.com/en-us/Mathematical+and+Computational+Modeling%3A+With+Applications+in+Natural+and+Social+Sciences%2C+Engineering%2C+and+the+Arts-p-9781118853986

Millen, D. R., & R., D. (2000). Rapid ethnography. In *Proceedings of the conference on Designing interactive systems processes, practices, methods, and techniques - DIS '00* (pp. 280–286). New York, New York, USA: ACM Press. http://doi.org/10.1145/347642.347763

Monroe, M., Lan, R., Lee, H., Plaisant, C., & Shneiderman, B. (2013). Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, *19*(12), 2227–2236. http://doi.org/10.1109/TVCG.2013.200

Ng, A., & Russell, S. (2000). Algorithms for inverse reinforcement learning. *Proceedings of the Seventeenth International Conference on Machine Learning*, 663–670. http://doi.org/10.2460/ajvr.67.2.323

O'Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Retrieved from http://www.amazon.com/dp/B019B6VCLO/ref=wl_it_dp_o_pC_nS_ttl?_encoding=UTF8&colid=ZK884WM2L344&coliid=I29DOBU158QJKB

Pasquale, F. (2015). *The Black Box Society: The Secret Algorithms That Control Money and Information*. Retrieved from http://www.worldcat.org/title/black-box-society-

the-secret-algorithms-that-control-money-and-information/oclc/880831105

Pentland, B. T., & Rueter, H. H. (1994). Organizational Routines as Grammars of Action. *Administrative Science Quarterly*, *39*(3), 484–510. http://doi.org/10.2307/2393300

Perer, A., & Gotz, D. (2013). Data-driven exploration of care plans for patients. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13* (p. 439). New York, New York, USA: ACM Press. http://doi.org/10.1145/2468356.2468434

Peter Stone, Rodney Brooks, Erik Brynjolfsson, Ryan Calo, Oren Etzioni, Greg Hager, Julia Hirschberg, Shivaram Kalyanakrishnan, Ece Kamar, Sarit Kraus, Kevin Leyton-Brown, David Parkes, William Press, AnnaLee Saxenian, Julie Shah, Milind Tambe, and A. T. (2016). Artificial Intelligence and Life in 2030. *One Hundred Year Study on Artificial Intelligence: Report of the 2015-2016 Study Panel*, 52. Retrieved from https://ai100.stanford.edu/sites/default/files/ai100report10032016fnl_singles.pdf

Pirolli, P., & Card, S. (2005). The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. *Proceedings of International Conference On*. Retrieved from https://www.e-education.psu.edu/geog885/sites/www.e-education.psu.edu.geog885/files/geog885q/file/Lesson_02/Sense_Making_206_Camera_Ready_Paper.pdf

Plaisant, C., Milash, B., Rose, A., Widoff, S., & Shneiderman, B. (1996). LifeLines: visualizing personal histories. *Proceedings of the {SIGCHI} Conference on Human Factors in Computing Systems: Common Ground*. http://doi.org/10.1145/238386.238493

Puterman, M. (1994). *Markov decision processes : discrete stochastic dynamic programming*. Retrieved from https://books.google.com/books?hl=en&lr=&id=VvBjBAAAQBAJ&oi=fnd&pg=PT9&dq=Markov+decision+processes:+discrete+stochastic+dynamic+programming&ots=rqoxuPO1TO&sig=Bghho2REoV3uYPrRxExhXhwgpu0

Rashidi, P., & Cook, D. J. (2010). Mining and monitoring patterns of daily routines for assisted living in real world settings. *Proceedings of the ACM International Conference on Health Informatics - IHI '10*, 336. http://doi.org/10.1145/1882992.1883040

Reason, J., Manstead, A., Stradling, S., Baxter, J., & Campbell, K. (2011). Errors and violations on the roads: a real distinction? *Ergonomics*, *33*(10–11), 1315–1332. http://doi.org/10.1080/00140139008925335

Rieman, J., Franzke, M., & Redmiles, D. (1995). Usability Evaluation with the Cognitive Walkthrough. *Conference Companion on Human Factors in Computing Systems*, 387–388. http://doi.org/10.1145/223355.223735

Robinson, R., & Hudali, T. (2017). The HOSPITAL score and LACE index as predictors of 30 day readmission in a retrospective study at a university-affiliated community hospital. *PeerJ*, *5*, e3137. http://doi.org/10.7717/peerj.3137

Ronis, David L., J., Yates, F., & Kirscht, J. P. (1989). Attitudes, decisions, and habits as determinants of repeated behavior. In *Attitude Structure and Function* (pp. 213–239). Retrieved from https://books.google.com/books?hl=en&lr=&id=fiOvSm50Z7kC&oi=fnd&pg=PA213&dq=Attitudes,+decisions,+and+habits+as+determinants+of+repeated+behavior&ots=5s28663OSH&sig=jvsi6ldExKczO138vwu2krnGG8k

Russell, D. M., Stefik, M. J., Pirolli, P., & Card, S. K. (1993). The cost structure of sensemaking. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '93*, 269–276. http://doi.org/10.1145/169059.169209

Sadilek, A., & Krumm, J. (2012). Far Out: Predicting Long-Term Human Mobility. *26th AAAI Conference on Artificial Intelligence*, 814–820. http://doi.org/10.1.1.224.6709

Salakhutdinov, R. (2009). Learning Deep Generative Models. *University of Toronto, Toronto, Ont., Canada*, *2*(1), 1–84. http://doi.org/10.1146/annurev-statistics-010814-020120

Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., … Summers, R. M. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, *35*(5), 1285–1298. http://doi.org/10.1109/TMI.2016.2528162

Shinar, D., & Compton, R. (2004). Aggressive driving: An observational study of driver, vehicle, and situational variables. *Accident Analysis and Prevention*, *36*(3), 429–437. http://doi.org/10.1016/S0001-4575(03)00037-X

Shmueli, G. (2010). To explain or to predict? *Statistical Science*, *25*, 289–310. http://doi.org/10.1214/10-STS330

Shneiderman, B. (2003). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *The Craft of Information Visualization* (pp. 364–371). http://doi.org/10.1016/B978-155860915-0/50046-9

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., … Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. http://doi.org/10.1038/nature16961

Starbird, K., & Palen, L. (2010). Pass it on?: Retweeting in mass emergency. *Proceedings of the 7th International ISCRAM Conference*, (December 2004), 1–10. http://doi.org/10.1111/j.1556-4029.2009.01231.x

Suh, J., & Com, S. M. (2016). The Label Complexity of Mixed-Initiative Classifier Training. In *ICML*. Retrieved from

http://www.jmlr.org/proceedings/papers/v48/suh16.pdf

Taylor, R. (1950). Purposeful and non-purposeful behavior: A rejoinder. *Philosophy of Science*. Retrieved from http://www.journals.uchicago.edu/doi/pdfplus/10.1086/287108

The Economist. (2015). Artificial intelligence - Rise of the machines. http://doi.org/10.1126/science.349.6245.248

Tukey, J. (1977). Exploratory data analysis. *Addison-Wesley Series in Behavioral Science:* Retrieved from http://adsabs.harvard.edu/abs/1977eda..book.....T

Viswanath Venkatesh , Michael G . Morris , Gordon B . Davis, F. D. . D., Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS Quarterly*, *27*(3), 425–478. http://doi.org/10.2307/30036540

Wattenberg, M. (2002). Arc diagrams: Visualizing structure in strings. In *Proceedings - IEEE Symposium on Information Visualization, INFO VIS* (Vol. 2002–Janua, pp. 110–116). http://doi.org/10.1109/INFVIS.2002.1173155

Weber, M., Alexa, M., & Müller, W. (2001). Visualizing time-series on spirals. *Infovis*, 7. http://doi.org/10.1109/INFVIS.2001.963273

Weiss, Y. (1996). Synchronization of work schedules. *International Economic Review*, *37*(1), 157–179. Retrieved from http://www.jstor.org/stable/2527251

Wu, X., & Zhang, X. (2016). Automated Inference on Criminality using Face Images. *ArXiv:1611.04135*. Retrieved from http://arxiv.org/abs/1611.04135

Yang, Q., Banovic, N., & Zimmerman, J. (2018). Mapping Machine Learning Advances from HCI Research to Reveal Starting Places for Design Innovation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18* (pp. 1–11). New York, New York, USA: ACM Press. http://doi.org/10.1145/3173574.3173704

Zhao, J., Chevalier, F., Collins, C., & Balakrishnan, R. (2012). Facilitating Discourse Analysis with Interactive Visualization. *IEEE Transactions On*, *18*(12), 2639–2648. Retrieved from http://ieeexplore.ieee.org/abstract/document/6327270/

Ziebart, B. (2010). *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. *Thesis*. Retrieved from http://repository.cmu.edu/dissertations/17/

Ziebart, B. D., Bagnell, J. A., & Dey, A. K. (2013). The principle of maximum causal entropy for estimating interacting processes. *IEEE Transactions on Information Theory*, *59*(4), 1966–1980. http://doi.org/10.1109/TIT.2012.2234824

Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Behavior, O. C. (2008). Navigate Like A

Cabbie : Probablistic Reasoning from Observed Context-Aware Behavior Navigate Like a Cabbie : Probabilistic Reasoning from.

Ziebart, B. D., Maas, A. L., Dey, A. K., & Bagnell, J. A. (2008). Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *10th international conference on ubiquitous computing* (pp. 322–331). http://doi.org/10.1145/1409635.1409678

Ziebart, B., Ratliff, N., & Gallagher, G. (2009). Planning-based prediction for pedestrians. *Intelligent Robots and ...*. Retrieved from http://ieeexplore.ieee.org/abstract/document/5354147/