UNIVERSITY OF CALIFORNIA,
IRVINE

Interactive Event-driven Knowledge Discovery from Data Streams

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Laleh Jalali

Dissertation Committee:
Professor Ramesh Jain, Chair
Professor Gopi Meenakshisundaram
Professor Nalini Venkatasubramanian

2016

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First and for most I would like to thank my advisor, professor Ramesh Jain, whose constant encouragement was vital in making this dissertation a reality. We spent endless hours discussing my research topic and he always gave me excellent suggestions. He taught me how to be a great researcher and encouraged me to dream big. I was always full of motivation after leaving his office. He is a fantastic listener. He was always there to listen and to give me sincere advice about anything I could have hoped for. He never gave up on me and always preserved his confidence in my ability to complete an outstanding Ph.D. We went to multiple conferences together. His companionship and sense of humor have left many of the best memories. He was and remains my best friend and my best role model as a scientist, mentor, and teacher.

Next, I would like to thank my thesis committee members, Prof. Gopi Meenakshisundaram and Prof. Nalini Venkatasubramanian. Gopi is an outstanding researcher. I am especially thankful for his comments to improve my thesis and his encouragements regarding my future career path. Nalini is a knowledgable professor. I am thankful for her insightful questions, which incented me to widen my research from various perspectives.

I would like to thank my fellow labmates, Siripen Pongpaichet, Hyungik Oh, Mengfan Tang, and Pranav Agrawal for the stimulating discussions and their support of my work. I would like to specially thank Siripen for her friendship through these years and for simply being there when I needed someone to consult or talk to. I had some of the best memories with her.

Finally, I would like to thank my parents for their love and support throughout my life and giving me strength to reach for the stars and chase my dreams. My sister and brother deserve my wholehearted thanks as well. I would like to thank my husband, Ramin, for his endless love and encouragement throughout this process as well.

CURRICULUM VITAE

Laleh Jalali

EDUCATION

Doctor of Philosophy in Computer Science — **2016**
University of California, Irvine (UCI) — *Irvine, CA*

Master of Science in Computer Science — **2015**
University of California, Irvine (UCI) — *Irvine, CA*

Master of Science in Computer Engineering — **2010**
Iran University of Science and Technology (IUST) — *Iran*

Bachelor of Science in Computer Sciences — **2007**
Isfahan University of Technology (IUT) — *Iran*

RESEARCH EXPERIENCE

Graduate Research Assistant — **2012–2016**
UCI, *Social Life Networks Lab* — *Irvine, California*

Graduate Research Intern — **2014**
National University of Singapore, *Sensor Enhanced Social Media Lab* — *Singapore*

Graduate Research Assistant — **2008–2010**
IUST, *Audio and Speech Recognition Lab* — *Tehran, Iran*

TEACHING EXPERIENCE

Teaching Assistant / Reader — **2010–2012**
University of California, Irvine — *Irvine, CA*

2009–2010
Iran University of Science and Technology — *Tehran, Iran*

REFEREED MAGAZINE AND JOURNAL PUBLICATIONS

Estimating Qualitative Causal Models from Observational Data **2016**
Laleh Jalali, Ramesh Jain, *submitted to* International Journal of Data Science and Analytics (JDSA)

Objective Self **2014**
Ramesh Jain, Laleh Jalali *Vision and Views* IEEE MultiMedia

Building Social Life Networks **2013**
Ramesh Jain, Laleh Jalali, Siripen Pongpaichet, and Amarnath Gupta, IEEE Data Engineering Bulletin

REFEREED CONFERENCE PUBLICATIONS

A Framework for Human Behavior Analysis from Multimodal Data Streams **Oct. 2016**
Laleh Jalali, Huyong Oh, Ramesh Jain, Ramin Moazeni. *submitted to* ACM Multimedia

Bringing Deep Causality to Multimedia Data Streams **Oct. 2015**
Laleh Jalali and Ramesh Jain, **Brave New Ideas** in ACM Multimedia

Using Photos as Micro-Reports of Events **June 2016**
Siripen Pongpaichet, Mengfan Tang, Laleh Jalali, and Ramesh Jain, **Brave New Ideas** in ACM International Conference in Multimedia Retrieval

Exploring Spatio-Temporal-Theme Correlation between Physical and Social Streaming Data for Event Detection and Pattern Interpretation from Heterogeneous Sensors **Sep. 2015**
Minh-Son Dao, Koji Zettsu, Siripen Pongpaichet, Laleh Jalali, Ramesh Jain, in IEEE International Conference on Big Data

An Intelligent Notification System Using Context from Real-time Personal Activity Monitoring **June 2015**
Huyong Oh, Laleh Jalali, Ramesh Jain , in International Conference on Multimedia and Expo

A Real-time Complex Event Discovery Platform for Cyber-Physical-Social Systems **April 2014**
Minhson Dao, Siripen Pongpaichet, Laleh Jalali, Kyoungsook Kim, Ramesh Jain, and Koji Zettsu, in ACM International Conference in Multimedia Retrieval

From Health-persona to Societal Health. **May 2013**
Ramesh Jain, Laleh Jalali, and Mingming Fan, in The International World Wide Web Conference

REFEREED WORKSHOP PUBLICATIONS

Complex Asthma Risk Factor Recognition from Heterogeneous Data Streams **2015**
Laleh Jalali, Minh-Son Dao, Ramesh Jain, Koji Zettsu, in Workshop on Multimedia Services and Technologies for E-health at ICME

Personicle: Personal Chronicle of Life Events **2014**
Laleh Jalali and Ramesh Jain, in Workshop on Personal Data Analytics in the Internet of Things (PDA@IoT)

Observing Real-World Phenomena through EventWeb over Space, Time and Theme **2015**
Siripen Pongpaichet, Mingfan Tang, Laleh Jalali, Ramesh Jain, in International Workshop on Building Web Observatories (B-WOW)

Building health persona from personal data streams **2013**
Laleh Jalali, Ramesh Jain, in ACM Workshop on Personal Data Meets Distributed Multimedia (PDM'13)

ABSTRACT OF THE DISSERTATION

Interactive Event-driven Knowledge Discovery from Data Streams

By

Laleh Jalali

Doctor of Philosophy in Computer Science

University of California, Irvine, 2016

Professor Ramesh Jain, Chair

With the proliferation of sensor data, a critical challenge is to interpret and extract knowledge from large-scale heterogeneous observational data. Most knowledge discovery frameworks relay on data mining techniques to extract interesting patterns. The problem of finding such patterns is NP-complete and the property of *interestingness* is not monotone since a pattern may be interesting, even if its subpatterns are not. In this dissertation a framework for interactive knowledge discovery from heterogeneous high-dimensional temporal data is presented. First, a high-level pattern formulation language is introduced. The language consists of an event model for fusing and abstracting data streams, a semi-interval time model for effectively representing temporal relations, and a set of expressive operators. Based on these operators, a visual and interactive framework is proposed which combines data-driven (bottom-up) and hypothesis-driven (top-down) analyses.

This framework takes advantage of data-driven operators for pattern mining and investigating *unknown unknowns* to generate a basic model and derive a preliminary knowledge. It also uses domain expert knowledge to guide the process of revealing *known unknowns*. An expert can seed a hypothesis, based on prior knowledge or the knowledge derived from data-driven analysis, and grow it interactively using hypothesis-driven operators. In the context of the pattern mining component, novel time efficient algorithms are introduced which

allow discovery of hidden event co-occurrences from multiple event streams. A prototype of the framework is implemented as a web based system which can be utilized as an effective tool for explanation and decision making in almost all disciplines. The applicability of this framework is evaluated in a healthcare application for asthma risk management and a human behavior understanding application, called Objective Self. These applications and experiments highlight the actionable knowledge that the framework can help uncover.

Chapter 1

Introduction

With the proliferation of sensor data, a critical challenge is to explore and extract knowledge from these large-scale heterogeneous data streams. Most of existing techniques and concepts for knowledge discovery are rooted in the time when we had scarcity of data and limited computing and communication technology. But with the emergence of Big Data, our challenge has shifted towards handling this abundance and harnessing the true potential of this ever-growing data stream. This data spreads over heterogeneous sources containing unstructured, semi-structured, or structured information. Before any analysis, unified signals need to be generated from asynchronous, multivariate, and multimodal information. Conceptual models are then constructed to account for data in different application domains. In order to extract insight from this data two major challenges arise: 1) How to fuse these modalities into a human understandable abstracted signals that not only preserve the semantics of the underlying system but also facilitate data analysis? 2) How to extract knowledge and build conceptual models in an abstraction process for the purpose of *understanding* and *explaining* the underlying complex systems?

In various disciplines, information about an underlying phenomenon might be acquired from different types of sensors. Rarely a single modality can provide complete knowledge of the phenomenon of interest due to rich characteristics and complexity of that phenomenon. What is common for all these data streams, is that they all capture and convey information about a real-life event. However, data from different sources usually result in different silos that do not communicate with each other and are indexed using data-centric approaches. In the current form it's not possible to make sense of these diverse sources of data. So data needs to be fused and transformed to a representation that not only induces an structure over unstructured and scattered data but also makes it suitable for knowledge extraction. Figure 1.1 shows the process of going from real-world data to abstraction of that data in a model building process. The models can then be used for explaining and understanding real-world complex situations.

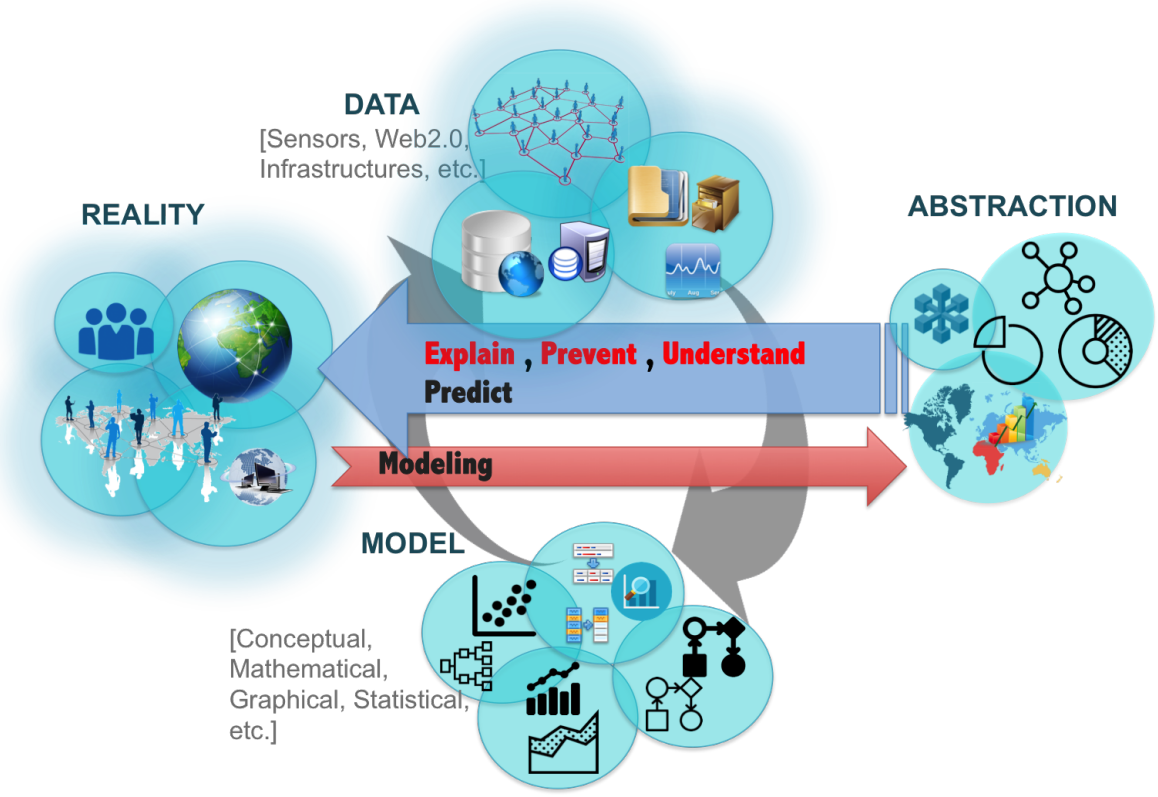


Figure 1.1: From data to abstractions with model building process. Models are used not only for prediction but for understanding and explaining.

Building models involve many decisions such as determining model selection strategy, defining a model structure, and selection of data and transformation applied to it. Most of these decisions involve reliance on theoretical or empirical results, that is expert's domain knowledge, and cannot be learned by a system itself solely from available input data. Hence, it is often necessary to incorporate human judgment into this modeling process. When we encounter large volumes of disparate information, data-driven discovery techniques are quite useful in refining human judgment. So, when scientist doesn't have any idea what hypothesis to generate or how to proceed in data analysis task, automatic data-driven techniques provide significant insight. By seeding a hypothesis based on this insight, analyst can incorporate her own domain knowledge and formulate a refined hypothesis quickly, systematically, and *grow* a hypothesis iteratively to generate a comprehensive model. The former is called *unknown unknowns* problem and the latter is called *known unknowns* problem [85]. The best practice is to create a balance between these two. A comprehensive modeling framework, on one hand, takes advantage of data-driven processing to find unknown unknowns, and on the other hand, uses domain expert knowledge to guide the process of revealing known unknowns with hypothesis-driven processing.

1.1 Data-driven vs. Hypothesis-driven

In model building and knowledge discovery process, a hypothesis-driven approach is one of the main methods for using data to test and ultimately prove (or disprove) assertions. To do that, researchers collect sufficient amount of data on a specific application domain and then approach it with a specific hypothesis in mind. There is also data-driven analysis, where data analysts dive into data in search of patterns. Such analysis is important to transform data into knowledge automatically. It does not start from a pre-conception or a specific question like hypothesis testing. Instead, it aims to automatically extract useful information

from large volumes of data via exploratory search, making it highly applicable for automatic knowledge discovery.

The avalanche of sensory data has led scientists to envision a future in which automated modeling techniques, or *data-driven discovery* will eventually rival the traditional hypothesis-driven research that has dominated research areas for at least the past century. Nowadays, there is much attention to deep learning and deep neural networks to model high level abstractions in data. However, these models lack reasoning, and are unable to perform unsupervised learning. They require a substantial amount of data for training or they overfit. Also, the number of hyperparameters to tune (e.g. number of layers, number of hidden units, optimization algorithm, activation function), make them inapplicable in certain fields [51]. Training them has historically been one of the biggest challenges. Only with the use of GPU's and techniques like dropout did these models gain momentum [1]. These modeling techniques work as black box. In some domains, interpretability is quite important. Despite the community's efforts, neural networks remain rather difficult to interpret, especially for input domains that do not consist of images. This makes many domain experts hesitant to adopt them, particularly in fields like biology. The scale of a net's weights (and of the weight updates) is very important for performance. When the features are of the same type (pixels, word counts, etc), this is not a problem. However, when the features are heterogeneous, the weights and updates will all be on different scales.

The interpretation of data and model is usually a challenging step in knowledge discovery process. There is a semantic gap between data-driven modeling techniques and the final domain expert or analyst who is assumed to use the extracted models. In general, bypassing the analyst and carrying knowledge discovery process as a black box via learning or statistical techniques have several problems:

- Hiding the inference from the analysts reduce their understanding of the model. Also lack of understanding might result in overconfidence, with potentially catastrophic consequences when learned models actually fail in real-world situations.
- Available methods typically cannot give advise to the analyst as how to refine the model and what to do next.
- By using available products, it is easy to fall into two traps: either the system acts at a very high level, and it is difficult to deviate from the way the developers of the product thought was the proper way to approach the problem, or the system acts at a very low level, and the analyst has to know a great deal in order to use the system meaningfully.
- Changing even a small parameter in a learned model, either for a specific circumstances or for including individual judgment, is difficult and needs the whole model to be re-learned. Even if the source code is available, the persons qualified to do the changes may not have the time to tap into the code successfully.

1.2 Knowledge Discovery from Temporal Data

Figure 1.2 shows the cycle of knowledge discovery at very high level as an iterative cycle between data, hypothesis, and knowledge. Hypothesis-driven modeling and reasoning uses background knowledge to construct a hypothesis and design experiments to collect relevant data. On the other hand, inductive reasoning is purely data-driven and generates insight from the underlying data. However, with induction alone, many spurious associations in the data might arise which results in false scientific discoveries and false statistical inferences. We argue that scientific advances in knowledge discovery and model building should exploit both hypothesis-driven and data-driven reasoning approaches in an iterative and human-understandable cycle.

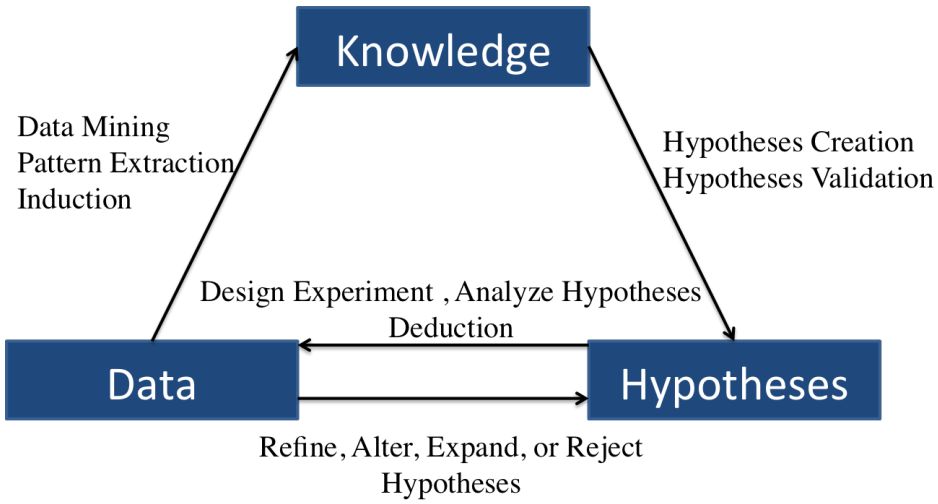


Figure 1.2: The cycle of knowledge discovery.

A large volume of diverse, noisy, complex, unstructured, and longitudinal data are continuously generated from sensors, internet transactions, social networks, videos and images, and click streams. Before, one of the major issues regarding Big Data was storage, especially with respect to the exponential growth and size of unstructured data that did not fit into databases. Today, however, the competitive landscape is very different. Proper storage is merely a pre-condition to finding the real value in Big Data. Turning data from massive streams into knowledge and thereby an actionable intelligence is the ultimate goal. A crucial observation is that this data is fundamentally *temporal*. Further, many common analytics queries are also fundamentally temporal, requiring the ability to perform time-ordered processing over the data. For example, a public health analyst wishes to understand *how the increase of pollution and pollen effects the number and intensity of asthma attacks within 7 days in asthmatic patients over a 365-day dataset*. The vague nature of such questions require data-driven analysis with special focus on temporal elements in the data to bring hidden patterns to surface. There might be situations that the analyst's questions can be formulated to well-defined queries. However, these queries are non-trivial temporal sequence computations that are fundamentally difficult to capture using traditional database-style set-oriented languages.

In this dissertation we propose framework for **interactive Knowledge Discovery from Data Streams (interactive KDDS)** that has the important effect of coupling both data-driven and hypothesis-driven modeling, keeping human in the loop while taking advantage of data-driven analysis when needed. In this sense, it is a new hybrid modeling approach. We emphasize on framework's capabilities in fusing and analyzing temporal data to capture dynamic characteristics of complex systems. The framework is built on a high-level descriptive pattern formulation and pattern mining language. The language is composed of a well-defined set of operators that facilitates pattern analysis. Bottom-up (data-driven) operators bring hidden interesting patterns to surface and top-down (hypothesis-driven) operators facilitate knowledge formulation and pattern query. In the processing phase, the operators are translated to their corresponding automata. Innovative temporal data mining algorithms are designed to process query automata. An analyst can interact with the framework and formulate query patterns using these operators. Data-driven operators are used for exploratory analysis to generate a basic model and derive a preliminary insight. Then analyst can seed a hypothesis and grow it step by step using top-down operators. A good hypothesis is not the one that is necessarily correct, but one that opens new paths of investigation. In complex problem domains this path cannot be fully perceived in advance. So analyst must be provided with appropriate operators to carry out new analyses based on the original hypothesis.

1.3 Contributions

This dissertation contributes towards the problem of knowledge discovery and model building process from temporal data streams. We present a knowledge discovery framework that facilitates deep computational understanding and model building in different domains using traditional as well as emergent new big data streams. This framework is a new tool for

empirical research and offers a natural environment for the study of structural and temporal inter-relations between multiple heterogeneous data sources. The main contributions are:

1. Abstracting data streams to conceptual events as a unified data representation that is understandable for experts and facilitates the manipulation of a model's structure. These events are **structuralization** of the timeline using semantics rather than the uniform structuralization as imposed by calendars.

2. Using semi-intervals to encode temporal facet of an event. Semi-intervals allow a flexible representation where partial or incomplete knowledge can be handled since operation is on parts of an interval and not the whole. As a result, temporal aspect of patterns can be a mixture of intervals and semi-intervals.

3. Designing a high-level declarative language which allows for formulation, query, and mining complex patterns from the combination of a unique set of well-defined operators. The language is very concise, contains a small number of operators, and is still very expressive. The input to the operators are event streams and the output is an event stream as well. Thus operators can be cascaded to formulate complex patterns. The pattern specifications are translated to corresponding pattern matching automata.

4. Providing data-driven processing by applying innovative temporal data mining techniques to extract frequent patterns. Since the complete set of frequent patterns contains a lot of redundant information, we introduce novel visualization to organize and present the results properly to help analyst explore results easily and gain insight.

5. Design and develop a comprehensive GUI that engage analysts in an interactive, online process, allows them to gain insight from data and iteratively use their knowledge to generate and evaluate new hypotheses to build effective model.

1.4 Thesis Outline

Chapter 2 explains knowledge extraction process in complex systems. It reviews some knowledge discovery software tools and dig deeper into knowledge discovery in healthcare applications. Chapter 3 discusses the related work and scopes the focus of this dissertation. Chapter 4 describes the data representation and pattern operators defined to formulate and query descriptive patterns. Chapter 5 explains a generic framework for interactive Knowledge Discovery from Data Streams (interactive KDDS) and a high-level pattern mining and query language using those operators. Chapter 6 discusses temporal data mining algorithms for pattern mining and pattern query. Chapter 7 explains how our knowledge discovery framework can be applied in Objective Self application for human behavior analysis. Chapter 8 explains how this framework can be used for understanding and predicting asthma risk factors at population level. Chapter 9 discusses the future challenges and concludes this dissertation.

Chapter 2

Understanding Knowledge Discovery Process

We live in a data abundance era. Availability of large volume of multimodal data streams (ranging from video, to tweets, to human's activity and location, and to environmental variables) can now be used to solve many critical societal problems. As the emphasis on collecting data increases, there is a need for new generation of techniques, frameworks and algorithms to assist researchers, analysts, and decision makers in extracting knowledge from such variety of data.

Previously, the techniques and tools for model building and knowledge extraction had been the subject of Knowledge Discovery in Databases (KDD) field. KDD has evolved from interaction among different fields such as machine learning, pattern recognition, statistics, artificial intelligence, database, and knowledge representation. The main idea in KDD is to discover high level knowledge from low level raw data. Most of our existing techniques and concepts in this direction are rooted in the time when we had scarcity of data and limited computing and communication technology. During the advent of KDD it was needed to

emphasize the focus on storage of data sets in databases and efficient algorithms compared to exploratory data analysis in statistics. That is why the suffix *database* is present in the term KDD. Today, however, databases have become an inevitable tool. Although raw data storage is crucial in knowledge extraction, the form that raw data is stored does not necessarily help in knowledge extraction process. In fact, since heterogeneous raw data is stored in separate data silos, knowledge extraction frameworks need to be able to pull data from disparate silos and unify, transform, and fuse them before any analysis.

This chapter starts with comparing several definitions of knowledge discovery in KDD field. Then explains what are the new considerations for data analysis and knowledge extraction for applications that involve heterogeneous and unstructured data. The chapter proceeds to comparing commonalities and differences between several data mining and knowledge discovery software tools. Today, healthcare applications are great example of systems that need to dead with increasingly large and complex sets of heterogeneous, high-dimensional, and unstructured data to extract meaningful knowledge. We investigate multiple projects along this line. Based on these investigations, we identify the requirements of an effective framework that supports knowledge extraction from such data.

2.1 Knowledge Discovery Definitions

Data mining and knowledge discovery have evolved from a collaboration of multiple fields such as artificial intelligence, statistics, machine learning, pattern recognition, knowledge acquisition, reasoning, and data visualization. An early definition of KDD is given by Fayyad et al. [42] where he defines KDD as "[...] *the KDD field is concerned with the development of methods and techniques for making sense of data.* [...] *At the core of the process is the application of specific data-mining methods for pattern discovery and extraction.* They also mention that "[...] *KDD refers to the overall process of discovering useful knowledge from*

data, and data mining refers to a particular step in this process. Data mining is the application of specific algorithms for extracting patterns from data". In their view KDD refers to the whole process of discovering useful knowledge from data, and data mining is a particular step in this process. Fayyad defines data mining as *"the application of specific algorithms for extracting patterns from data. The patterns need to be valid on new data, novel, potentially useful,and understandable"*. Since data mining is considered as an important step in knowledge discovery, sometimes the boundaries between these tow are not easy to draw and they are often used interchangeably. For example Hand et al. defines data mining as *"the analysis of large observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner"* [60]. They put emphasis on the fact that data sets are usually large, creating algorithmic challenges and distinguishing data mining from classical exploratory data analysis. Also the relationships found in the data are required to be novel and understandable. From database perspective, Dunham describes data mining as *"finding hidden information in a database"* [38]. The authors mention that knowledge discovery in databases and data mining are often used interchangeably but they adopt the view of Fayyad et al. and define KDD as *"the process of finding useful information and patterns in data"* while data mining is a sub-step in the process.

The perspective that above mentioned authors adopt is that KDD is an automatic, exploratory analysis and modeling of large data repositories. It is the organized process of identifying valid, novel, useful, and understandable patterns from large datasets and data mining is the core of the KDD process, involving the algorithms that explore the data and discover previously unknown patterns. In general, the process of KDD consists of an iterative sequence of the following steps [88] (as shown if Figure 2.1):

- Data Selection
- Data pre-processing and cleaning

- Data Mining
- Evaluation and Interpretation
- Knowledge presentation

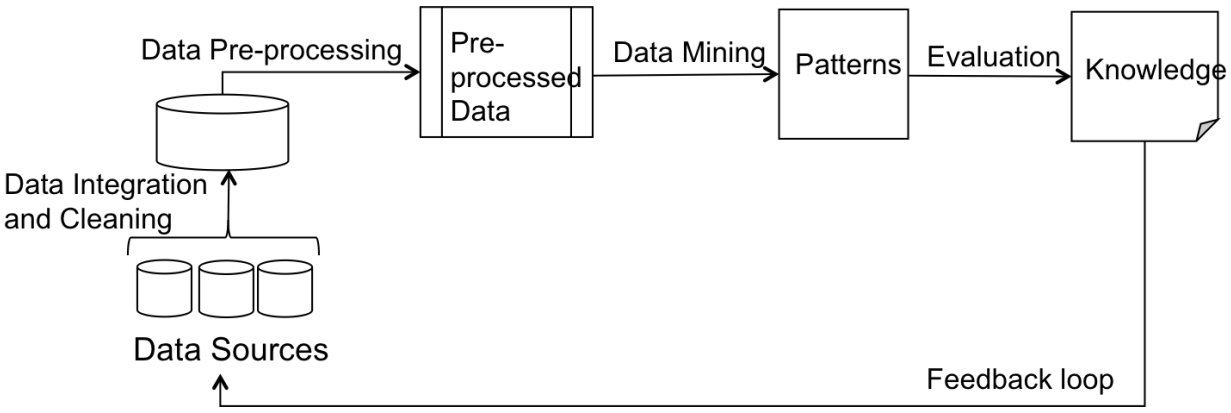


Figure 2.1: The process of Knowledge Discovery in Databases.

2.2 New Paradigm: Actionable Knowledge Extraction

A large volume of diverse, noisy, complex, unstructured, and longitudinal data are continuously generated from sensors, internet transactions, social networks, videos and images, and click streams. The speed and scale of generation of such data will even increase further in future. This requires new frameworks that support high performance computing, novel processing techniques (for fusing and mining heterogeneous data to uncover hidden patterns and unknown correlations), scalable software tools, and useful visualizations (that help analysts and decision makers understand the results better). Such process results in information lately referred to as *actionable knowledge* from the massive volume of complex raw data [106][33].

Knowledge discovery process from complex big data involve many complicated decisions such as determining model selection strategy, defining a model structure, defining criteria

for model goodness, selection of data and transformation applied to it, tuning learning parameters, etc. Most of these decisions involve reliance on theoretical or empirical results, that is expert's domain knowledge, and cannot be learned by a system itself solely from available input data. Moreover, many spurious associations might arise from learned models which results in false scientific discoveries and false statistical inferences. There is evidence that in some application domain humans outperform automatic machine learning algorithms, e.g., instantaneous interpretation of complex patterns in radiology imaging[10]. A promising approach for knowledge extraction in complex systems is to adopt a human-in-the-loop approach in the data processing step. This integrates the high-level expert knowledge into the knowledge discovery process by acquiring her relevance judgments regarding a set of initial retrieval results. Despite apparent benefits of such perspective, frameworks that facilitate a seamless interaction between a domain expert and traditional KDD process are not well-studied. The main question is: When and how in the knowledge extraction and model building process, an expert can incorporate her knowledge to guide the KDD process?

As mentioned in chapter 1, proper data storage is merely a pre-condition for knowledge extraction. So we drop the term *Databases* from KDD process and and add the term *Data streams* to emphasize on the importance on temporal data in this process. We define **interactive Knowledge Discovery from Data Streams (interactive KDDS)** as a *human-centered actionable knowledge extraction process from temporal data*. Knowledge is either a verified *hypothesis* or machine-generated *patterns* that are novel, interesting, and human-understandable. We emphasize that knowledge is required to be representable in a linguistic form, that is understandable by humans and automatically usable by knowledge-based systems. Knowledge can either be domain expert's hypotheses, or it can be derived from previously unknown, useful, and understandable patterns. The conversion of sub-symbolic patterns and trends in data to a symbolic human-understandable form is the most critical part of such interactive data analysis.

Figure 2.2 shows where is the human-in-the-loop and when do we need the human-in-the-loop in an interactive knowledge discovery process.

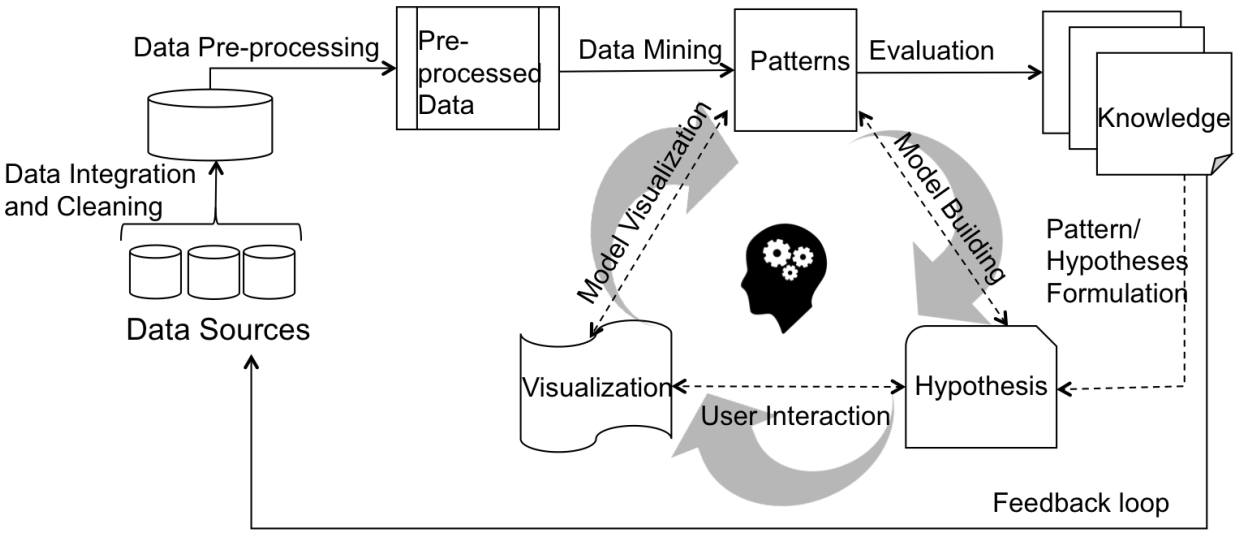


Figure 2.2: Interactive Knowledge Discovery (IKDD) process.

2.3 Data Mining and Knowledge Discovery Software Tools

This section provides an overview of frameworks that facilitate knowledge discovery and data mining with visual and interactive capabilities.

ForceSPIRE [40] is a visual analytics system to generate meaningful spatializations from text data, i.e., laying out documents visually such that the layout reflects user notions of similarity and distance.

Environment for Developing KDD-Applications Supported by Index-Structures (ELKI) [4] is an open source data mining software with focus on cluster analysis and outlier detection. ELKI offers data index structures such as the R*-tree. The modular design makes ELKI extendable. However data visualization and user interaction are static.

KEEL (Knowledge Extraction based on Evolutionary Learning) [11] provides a simple GUI and evolutionary algorithms (e.g. evolutionary rule learning models, evolutionary neural networks, genetic programming). KEEL contains a variety of classical knowledge extraction algorithms and pre-processing techniques such as data selection, feature selection, discretization, and statistical methodologies for contrasting experiments.

CMSR Data Miner [2] stands for Cramer Modeling, Segmentation and Rules. It is an integrated environment for predictive modeling, segmentation, data visualization, statistical data analysis, and rule-based model evaluation. It supports several algorithms, such as Self Organizing Maps (SOM), neural clustering, Radial Basis Function (RBF) with rule engine, segmentation and gains analysis, correlation analysis, and so on. It also support multiple visualization charts such 3D bars, histograms, histobars, scatterplots, boxplots, etc.

Unstructured Information Management (UIMA) [5] is a component software architecture for the development, discovery, composition, and deployment of multimodal analytics for the analysis of unstructured information. UIMA processing occurs through a series of modules called analysis engines. The result of analysis is an assignment of semantics to the elements of unstructured data, for example, the indication that the phrase "Washington" refers to a person's name or that it refers to a place. One potential use of UIMA is in a logistics analysis software system that could convert unstructured data such as repair logs and service notes into relational tables. These tables can then be used by automated tools to detect maintenance or manufacturing problems. Another use of UIMA is in systems that are used in medical contexts to analyze clinical notes.

PyBrain [123] is a modular python-based machine learning library. It implements learning algorithms and architectures ranging from reinforcement learning to optimization and evolutionary methods. The basic structure of the library enables a compositional approach to machine learning.

Coordinated Graph Visualization (CGV) [132] [6] is an interactive graph visualization system. The main focus of the CGV system is on user interaction, addressing the problem of how to interact with a graph that is too large to be displayed on a computer screen. CGV is based on hierarchy tree computations and graph macro-views that can be displayed in the available display space. Multiple views allow the user to explore, interact and access the data from different perspectives simultaneously. The main canvas of main graph view uses a node-link representation of a macro-view graph.

Massive Online Analysis (MOA) [21] is a data stream mining tool that provides dynamic visualization of data stream clustering. MOA includes a collection of offline and online methods as well as evaluation tools. In particular, it implements several classifier methods such as: Naive Bayes, Decision Stump, Hoeffding Tree, Hoeffding Option Tree, Bagging, Boosting, and Bagging using Adaptive-Size Hoeffding Trees. Although the current focus in MOA is on classification, they plan to extend the framework to includevdata stream clustering, regression, and frequent pattern learning.

Bak et al. [17] uses a combination of interactive visual analysis methods in order to effectively analyze multivariate datasets with demographic data. The authors use a Self-Organizing Map (SOM) to visualize each cluster. Opacity bands within this plot illustrate the variance within a cluster. The background color coding is used to correlate the cluster with a target value. User interaction is facilitated at different stages of the analysis as well as within the visualization.

All the software tools mentioned above follow a general pattern where they import one or more sets of data, perform analysis, and provide the output in different formats. The following general themes are investigated for each of these tools:

- **Input**: The input data could be either homogeneous (HO) or diverse, multimodal, and heterogeneous (HT). The input aggregation in heterogeneous data is either Deep

or Shallow. If the software aggregates data from multiple sources and transforms data to a higher abstraction, it is called deep aggregation.

- **Processing**: Different classifiers, clustering, and data analysis techniques can be applied to the data. User either actively or passively interact with the data during the knowledge extraction process. Active interaction allows user for interactive adjustment of the data mining and model building parameters.

- **Output**: Output could be presented in multiple ways, e.g. 2D or 3D charts and histograms, graphs, clusters, or text. Many applications allow users to query the system for knowledge extraction.

- **Other**: These tools might be extendable with new functionalities with minimal or no effects on its internal structure and data flow.

Table 2.1 shows the result of comparison.

Table 2.1: Comparison between different data mining and knowledge discovery software tools

Software Tool	Input		Processing		Output		Other	
	Data Type	Aggregation	Analysis	User Interaction	Visualization	Query	Extendable	Specification
ForceSPIRE	HO	Shallow	Clustering	Passive	2D-projection	No	No	Text Analysis
ELKI	HO	Shallow	Clustering	Passive	1D-histogram, 2D-projection	No	Yes	Outlier Detection, Clustering
KEEL	HT	Shallow	Evolutionary Alg.	Passive	Charts & Tables	No	Yes	Knowledge Extraction
CMSR	HT	Shallow	NN & Clustering	Passive	2D & 3D charts	Yes	No	Cluster Analysis
UIMA	HO	Shallow	Classification	Passive	Text	No	Yes	Text Mining
PyBrain	HO	Shallow	Reinforcement Learning	Passive	Dashboard	No	Yes	Machine Learning, Numerical Optimization
CGV	HT	Deep	Graph Hierarchy	Active	Graph Viz	No	No	Graph Analysis
MOA	HT	Shallow	Multiple Classifiers	Active	Dashboard	No	Yes	Stream Analysis
Bak et al.	HT	Deep	SOM	Active	Cluster Visualization	No	No	Data Analysis

2.4 Knowledge Discovery in Healthcare Applications

Knowledge discovery tools can be utilized in a wide range of applications in Engineering (intrusion detection and network security, flow classification), business (fraud detection, decision support systems, forecasting market trend), environmental science (flood prediction, urban environmental analysis), and medicine and population health (study of drug implications, disease outbreak). There is a wealth of data available within the healthcare industry that would benefit from the application of knowledge extraction tools and techniques. Many providers use Electronic Health Records (EHRs) to store a large quantity of patient's data on test results, medications, prior diagnoses, and other medical history. Also, with the advent of wearable sensors and widespread use of smartphone apps, huge amount of data can be collected from patients. These data can be converted to useful knowledge to improve patient's experience by employing knowledge extraction techniques. This section provides an overview of interactive knowledge discovery frameworks with focus on healthcare applications.

Many researchers have developed knowledge discovery methods for temporal events in the healthcare domain with specific focus on visualization. Early works focused mostly on visualizing an individual's medical record, e.g. LifeLines [114] and [9] , or individual's care plan [54]. Such tools typically organize data hierarchically to summarize the complex set of values associated with an individual patient. More recently, attention has shifted to visualizations of information related to a group of patients. This includes a range of tools for visualizing, querying, and sorting through groups of patient's event data [144][145].

Wongsuphasawat et al. introduced *Outflow* [143], a visualization framework summarizing temporal event data extracted from multiple patient medical records to show aggregate disease evolution statistics for a group of patients. Outflow aggregates event sequences into an *Outflow graph* which is analogous to a state diagram or state transition graph. The states are the unique combinations of symptoms that were observed in the data. Edges

capture symptom transitions. User interaction capabilities such as panning and zooming, symptom selection, filtering, and brushing are incorporated into the framework. Although the visualization aspects of the framework is interesting, it lacks pattern mining techniques and user interaction is limited to simple capabilities such as panning and zooming, symptom selection, and filtering. Also when the numbers of event types exceeds certain number(e.g., over 20), it can lead to an extremely complex web of event pathways.

Fails et al. [41] propose an interface, called *PatternFinder*, for visual query and result visualization for searching temporal patterns within multivariate and categorical clinical data sets. A dashboard UI can be used to select an arbitrary number of events and timelags between events to create a pattern. The pattern is then translated to an SQL query and results are shown as a timeline for each patient. Event are considered as instantaneous points at *day level* as lowest time granularity. Each event has a type and a values, where values can be numeric, such as for a systolic blood pressure reading, or categorical, such as normal/abnormal blood sugar. Although this interface provides comprehensive dashboard in terms of pattern formulation and query, it is restricted to sequential pattern, it lacks any pattern discovery technique to help analyst bring hidden patterns to surface. Also the framework lacks any data fusion and data transformation module.

Gotz et al. [53] present a methodology for interactive mining and visual analysis of clinical event patterns using electronic health record data. They apply a conventional pattern mining technique, Sequential PAttern Miner (SPAM) [15], for pattern discovery and display an ad-hoc visualization of discovered patterns. The framework address three issues: visual query capabilities to interactively specify pattern definitions; pattern mining techniques to discover important patterns; and interactive visualization techniques that help uncover events that impact the outcome most. One major problem with the framework is that events are point events and there is no notion of temporal constraint between events.

Pastrello et al. [109] note the importance and the challenges involves in integrating heterogeneous data from multiple experiments. They focus on network analysis as a key technique to integrate, visualize and extrapolate relevant information from diverse data. Authors consider networks as nodes and edges, where individual entities (e.g., genes, proteins, drugs) are represented by nodes, and relationships between components (e.g., drug-targeting) are represented by edges. They mention that network visualization and analysis can be utilized to integrate diverse data which results in an insightful representation of the system being studied. They integrate different data sets from the literature in one network to obtain insight in gastric cancer problem.

Koelling et al. [72] introduce a web-based tool for visual data mining colocation patterns in multivariate bioimages. The system called *Web-based Hyperbolic Image Data Explorer (WHIDE)*. Authors emphasize that bioimaging techniques rapidly develop toward higher resolution and higher dimension. The analysis of such Multivariate Bio-Images (MBIs) requires new techniques to support end users in the analysis of both aspects of such images: space and molecular colocation or interaction. Their approach combines principles from machine learning, dimension reduction, and visualization.

Bowman et al. [23] introduce a neuroimaging visualization framework, called *INVIZIAN*, for the graphical rendering and the dynamic interaction with the contents of large-scale neuroimaging data sets. Their system graphically displays brain surfaces as points in a coordinate space. The user can interact with the elements in this space, search over meta-data features, select one or more brain surfaces, and add create groups to generate new hypotheses that is worth new experimentation. This process enables the user to rapidly explore large collections of neuroimaging data for identifying interesting trends across features and attributes.

2.5 Design Requirements for an Interactive Knowledge Discovery Framework

After considering the nature of different knowledge discovery and data mining tools and taking into account the requirements of healthcare applications, we identify the following design principles for an effective knowledge extraction framework that is able to support complex applications.

Data Fusion and Transformation Information about an underlying phenomenon need to be acquired from different types of sensors and multiple sources. Rarely a single modality can provide complete knowledge of the phenomenon of interest due to rich characteristics and complexity of that phenomenon. The knowledge extraction framework needs to collect, aggregate and process heterogeneous data from multiple sources. Data fusion and data transformation are essential to convert data to a representation that is suitable for knowledge extraction. Also, in order to facilitate human interaction with the framework, data needs to be converted to a human-understandable form. This will bridge the gap between flow of information in real world and how data is captured, managed, and processed in cyber world.

Data-Driven Analysis The knowledge discovery framework should keep human in the loop while emphasizing on using advanced machine-based and data-driven model building techniques. These techniques bring hidden patterns to surface and help an analyst manage unknown unknowns situations.

Expressive Query Language The analyst or end user must be able to interact easily with knowledge discovery system while having a high-level view of the data. Query capabilities are needed to filter, select, and formulate query patterns. Patterns are composed of complex data types and temporal primitives. These functionalities need to be embedded in the language in an expressive and human understandable form. The expressive nature of patterns and

query language enhance understandability of model building process and help analyst step into the process rather than relying on a blackbox and *one click prediction model building*.

Interactive Modeling Approach The iterative querying capability not only allows the analyst to apply the data-driven algorithms on the data to discover interesting patterns, but also formulate hypotheses and investigate these patterns further for a deeper analysis. This is an iterative process which allows the analyst to use the models not only as static knowledge to be presented as result, but as an active element of the process used to go deeper in the data understanding.

Visualization and Result Interpretation The interpretation of data and model is usually a very complex part of the knowledge discovery process. The main point is that the interpretation of the mined patterns depend on the application context. Indeed, there is a wide semantic gap between the mining algorithms and the final domain expert user who is assumed to use the extracted models. The objective of the framework is to fill this gap providing the final user with meaningful, interpretable, and understandable patterns.

Table 2.2 compares the above mentioned systems , from data fusion and transformation to the ability to discover hidden relationships and interactive visualizations.

Table 2.2: Survey of various healthcare-specific interactive knowledge discovery frameworks and their characteristics. Legend: × = Not supported, ✓= Supported, ∘ = Partial support.

	Data Fusion	Data Transformation	Data-Driven Analysis	Querying	User Interaction	Visualization
Wongsuphasawat et al.	×	×	×	∘	∘	✓
Fails et al.	×	×	×	✓	✓	✓
Gotz et al.	×	×	✓	✓	✓	✓
Pastrello et al.	✓	✓	×	×	∘	✓
Koelling et al.	×	∘	✓	×	∘	✓
Bowman et al.	✓	✓	×	✓	✓	✓
interactive KDDS	✓	✓	✓	✓	✓	✓

Chapter 3

Literature Review

Knowledge discovery from temporal data involves many steps, and temporal data mining is an important step among those. In particular, the accommodation of time into mining techniques provides a window into the temporal arrangement of events and, thus, introduce the ability to suggest cause and effect that are overlooked when the temporal component is ignored or treated as a simple numerical attribute. Moreover, temporal data mining has the ability to mine the behavioral aspects of a system as opposed to mining rules that describe their states at a point in time. Hence, temporal data mining helps understanding *why* rather than merely understanding *what*.

As explained in the previous chapter, the emphasis is on combining data-driven processing and hypothesis-driven processing in one platform, and bringing human in the loop for model building and knowledge discovery process. The former concerns with pattern mining/pattern discovery and the latter concerns with pattern query since a hypothesis needs to be explicitly translated to a query. So in temporal data mining, we focus on related literature that cover these two paradigms. We stress on the importance of involving human in the knowledge

discovery process and human can understand and relate to qualitative data more effectively. We briefly review qualitative reasoning and modeling at the end of this chapter.

3.1 Temporal Data Mining

In general, temporal data is considered as a sequences of a primary data type, most commonly numerical or categorical values and sometimes multivariate information. Examples of temporal data are numerical time series (e.g., EEG, stock price), event sequences (e.g., sensor readings, online click streams, medical records), and temporal databases (e.g., timestamped tuples). The literature related to data mining tasks in temporal data are scattered across different domains. The objectives of temporal data mining, however, can be characterized as: 1)pattern analysis; 2)search and retrieval; and 3)prediction and trend analysis; 4)temporal classification; 5)temporal clustering.

As explained in the previous chapter, we are interested in knowledge discovery from heterogeneous data, and introduced the concept of event streams (e.g a sequence of events, where each event has multiple properties) to fuse multivariate data from isolated silos and to facilitate knowledge discovery in a uniform framework. We define events as a complex piece of information with symbolic event types and event properties. So among different categories in temporal data mining, pattern analysis and pattern discovery, which consider pattern as a string of characters, is more relevant to the direction of this dissertation.

3.1.1 Definitions and Concepts

Temporal data mining is the mining of data that has some temporal aspect. In general, temporal data mining covers a large area of problems and mining methods. Applying traditional data mining techniques by treating the temporal aspects of data as static features and

ignoring the temporal structure of the data is not enough for knowledge acquisition. Lin et al. describe temporal data mining as *a single step in the process of Knowledge Discovery in Temporal Databases that enumerates structures (temporal patterns or models) over the temporal data*[...]. They list several mining tasks such as classification, clustering, and induction and identify two problems namely similarity problem and periodical problem in temporal data mining tasks [84]. Morik et al. emphasize on the importance of the representation and transformation of the temporal data into a form appropriate for further learning [98].

Temporal mining covers a wide spectrum of paradigms for knowledge discovery. For example, consider the following temporal rule: *When the stock price of company A increases, the stock price of company B increases within the next 1 day.* This rule can be the output of a trend discovery algorithm that explores the similarities between the two time series. As another example, consider a temporal association rule indicating that bread and milk are purchased together during spring. This rule shows an association between the two products, with a temporal aspect *during spring* indicating that the rule occurs more frequently in certain time frames. The discovery of correlations among events along time dimension is another domain of temporal mining, e.g. the discovery of web usage patterns. This direction is usually referred to as frequent sequences mining.

To address such aspects of temporal data mining in a comprehensive literature, different temporal data categories and the algorithms applied to each category are shown in Figure 3.1. The data subjected to knowledge discovery can be two types: 1) numerical time series data, such as stock prices, where data has specific timestamps; 2) sequence of events, where sequence relationships such as before and after (or richer relationships described by Allen [13] or Freksa [46]) can be considered to mine patterns. With respect to the mining process employed in the discovery of pattern, we can identify different approaches. They include methods of supervised and unsupervised classification, methods for the discovery of associ-

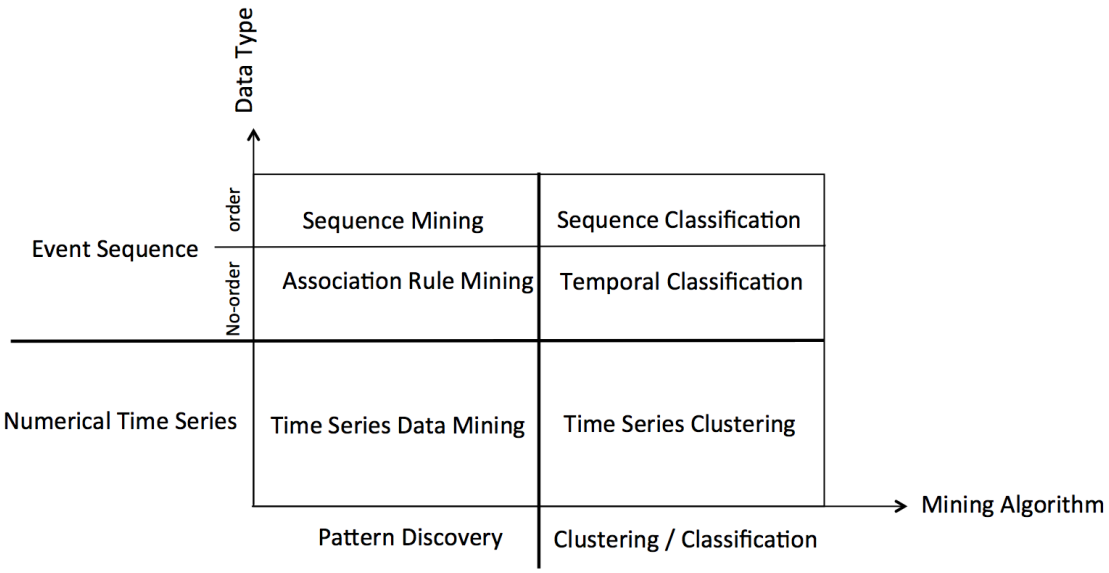


Figure 3.1: Categorization of input data type and temporal data mining algorithms.

ation rules, and frequent sequences, as well as the mining languages for the specification of patterns to be detected on the temporal sequences.

Figure 3.2 shows the taxonomy of this section.

3.1.2 Pattern Discovery

In pattern discovery the objective is to bring all interesting patterns to surface. So pattern discovery has an exploratory and unsupervised nature. Although knowledge discovery results in some kind of model, that is a global and high-level abstraction of the data, a pattern indicates a local structure between some of data points. In the literature, there is no universal criteria for measuring the *interestingness* of a pattern. However, requency of patterns is a useful criteria for this purpose. A frequent pattern is one that occurs many times in the data. Much of data mining literature is concerned with formulating useful pattern structures and developing efficient algorithms for discovering patterns which occur frequently in the data. There are two broad areas in pattern discovery: 1)Sequence mining; and 2)Frequent

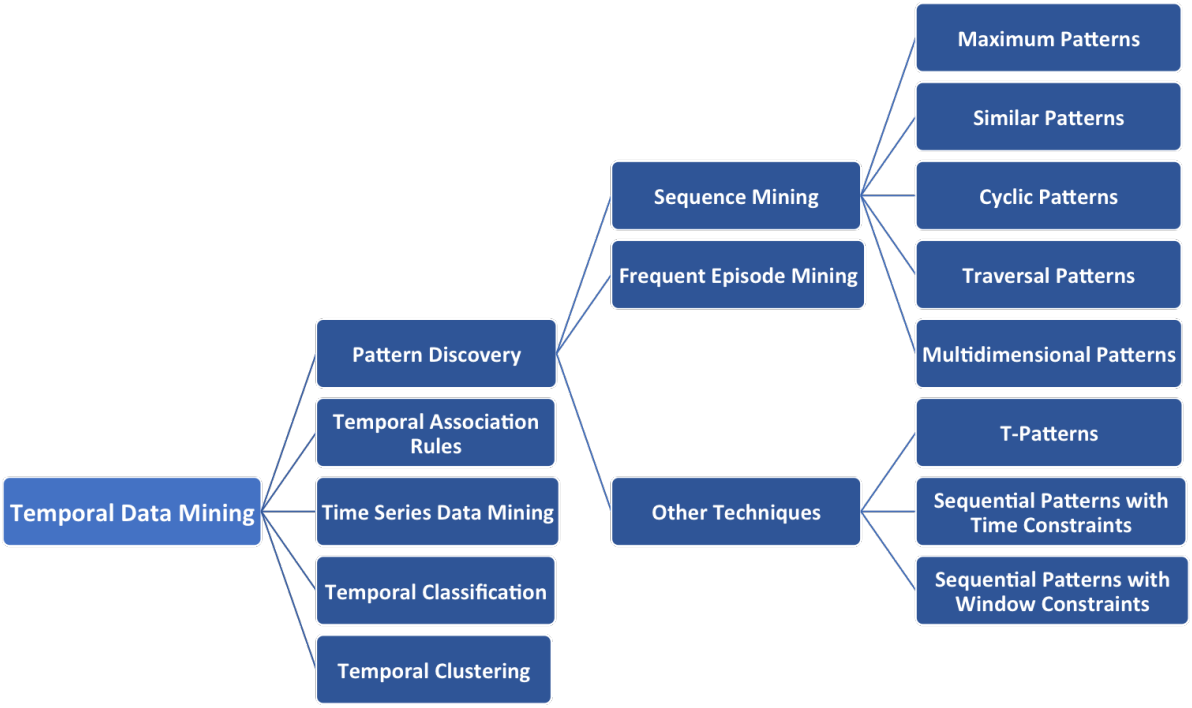


Figure 3.2: Taxonomy of temporal Data mining.

episode mining. Next we explain these two approaches, the primary algorithms, and the improvements proposed by across the literature.

Sequence Mining

Sequence mining was first proposed by Agrawal and Srikant [8]. It assumes that the customer transaction database is given, where each transaction includes the customer ID, the transaction time, and the item set purchased in the transaction. The goal is to discover the sequential patterns that occur frequently in the database. An *itemset* is a nonempty set of items and a *sequence* is an ordered list of itemsets. For example an itemset i denoted as $(i_1, i_2, ..., i_k)$ where i_i is an item. A sequence $s =< s_1, ..., s_n >$ where s_j is an itemset. A sequence $a =< a_1, ..., a_n >$ is *contained* in another sequence $b =< b_1, ..., b_m >$ if there exist integers $i_1 \leq i_2 \leq ... \leq i_n$ such that $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i_2}$,..., $a_n \subseteq b_{i_n}$. For example the sequence $a =< (3)(4,6)(8) >$ is contained in $b =< (3,2)(4,7,6)(12,8) >$. Sequence a shows that item

3 is bought, and then item 4 and 6 bought together. In a set of sequences, a sequence s is *maximal*, if s is not contained in any other sequence. All the transactions of a customer can together be viewed as a sequence where each transaction corresponds to a set of items and the list of transactions, ordered by increasing transaction time, corresponds to a sequence that is called a customer-sequence.

The support for any arbitrary sequence, s, of itemsets, is the fraction of customer transaction sequences in the database D which contain s. The user can specify a minimum support threshold θ and sequences that their support is greater than θ are called a *large sequence*. If a sequence a is large and maximal (among the set of all large sequences), then it is regarded as a *sequential pattern*. Now the problem is, how to find all such sequential patterns in D. Two general algorithms are presented by Agrawal and Srikant and referred to as *AprioriAll* and *AprioriSome* algorithms [8]. The AprioriAll algorithm counts all the large sequences and then prunes out the non-maximal sequences in a post-processing step. In the first pass through the data the large 1-sequences patterns are obtained. Then candidate 2-sequences are constructed by combining large 1-sequences itemsets in all possible ways. The next pass identifies the large 2-sequences. Then large 3-sequences are obtained from large 2-sequences, and so on. The AprioriSome algorithms intelligently exploit the maximality constraint. Since the search is only for maximal sequences, one can avoid counting sequences which would anyways be contained in longer sequences. For this longer sequences shall be counted first. Thus, the AprioriSome algorithm have a forward phase, in which all frequent sequences of certain lengths are found, and then a backward phase, in which all the remaining frequent sequences are discovered.

Since then, the sequence mining problem has drawn a great deal of attention and many extensions have been proposed in various directions as follow:

- Different variants of sequential patterns including maximum patterns [8], similar patterns [7], [79], cyclic patterns [58], [86], traversal patterns [111], and multidimensional patterns [148].
- Improved Algorithms for mining sequential patterns [59], [15], [110].
- Mining sequential patterns with constraints [99], [112], [142].

Some of these methods are being proposed as the improvement upon the performance of the algorithm by Agrawal and Srikant. For example, Lin and Lee [83] introduce a system for interactive sequential pattern discovery, where the user queries with several minimum support thresholds iteratively and discovers a set of patterns corresponding to the last threshold.

Another well-known method is *PrefixSpan* algorithm [110] that uses a divide-and-conquer strategy to solve the sequence mining problem for point events. First, it scans the database to find the frequent 1-patterns (L_1). Second, suppose there are $|L_1|$ patterns in L_1 . The original database, therefore, is divided into $|L_1|$ partitions, where each partition is the projection of the sequence database with respect to the corresponding 1-pattern. Ten, similarly to the first step, each partition is treated as the original one and finds all large 1-patterns in this partition. Appending these large 1-patterns, will generate frequent patterns with the length increased by 1. This way, the prefixes are successfully extended. Finally, recursively running step two and step three until prefixes cannot be extended will get all frequent sequential patterns. The *SPAM* [15] approach is a depth-first search using a memory resident bitmap representation to achieve a high speedup compared to *PrefixSpan* for large data sets. Another algorithm, *CloSpan* [147] is also proposed to mine only closed sequential patterns (patterns with no super patterns of the same frequency). To free the user from choosing the minimum frequency, Tzvetkov et al. propose to mine the top k closed patterns and changing the threshold dynamically [136].

Frequent Episode Mining

Another approach for temporal pattern discovery in sequential data is frequent episode mining [89]. In the sequential patterns mining explained in previous section, a collection of sequences is given, and the task is to discover ordered sequences of items that occur in those sequences repeatedly. In the frequent episode mining, the data is given in a single long sequence and the task is to discover temporal patterns, called episodes, that occur frequently in that sequence.

The data is referred to as an event sequence, denoted by $< (E_1, t_1), (E_2, t_2), ... >$, where E_i is a point event with timestamp t_i. An episode is a partial ordered set of events. When the order among the event types of an episode is total, it is called a serial episode and when there is no order at all, the episode is called a parallel episode. So parallel episodes are similar to itemsets where order between items does not matter. For example $(A \rightarrow B \rightarrow C)$ is a 3-node serial episode and (ABC) is a 3-node parallel episode. Given a pre-defined frequency threshold θ, all episode with frequency higher that θ are considered frequent. The process of frequent episode discovery is a level-wise algorithm that starts with discovering frequent 1-node episodes. These are then combined to form candidate 2-node episodes and then by counting their frequencies, 2-node frequent episodes are obtained and so on.

Mannila et al. [89] applies a windows-based frequency measure for counting episodes. [27] propose another frequency measure where instead of the window width in windows-based technique, the user chooses the maximum inter-node distance allowed and the algorithm automatically adjusts the window width based on the length of the episodes being counted. Laxman et al. [75] propose frequency counts for non-overlapped episode occurrence and the non-interleaved episode occurrence that is based on directly counting some suitable subset of occurrences of episodes. Graph-theory approaches have also been explored to locate episode occurrences in a sequence [134][62]. The idea is to employ a pre-processing step to build

an automaton called the DASG (Directed Acyclic Subsequence Graph), which accepts a string if and only if it is a sub-sequence of the given input sequence. Once the DASG is constructed, an episode can be located in the sequence in linear time. These algorithms, however, are more suited for pattern matching applications rather than for discovery of all frequent episodes.

Other Techniques

In pattern discovery domain, we are interested in symbolic temporal pattern mining where the focus is discovering interesting patterns among symbolic temporal data. *T-patterns* developed by Magnusson [87] is another approach where a sequence of point events will occur within certain time windows, e.g., $A_1 \xrightarrow{T_1} A_2 \xrightarrow{T_2} A_3$ for time intervals T_1 and T_2. T-patterns are used as the basis of pattern representation in $THEME$ software [1] where each T is set through various statistical methods.

Almost all of the research mentioned above deals with point-based event data. However, in real-life applications many events cannot be treated as a spontaneous point in time but rather, they span over a time interval in form of interval events with a specific start and end timestamps. Discovering temporal patterns from interval-events can reveal more interesting patterns. For example, The pattern: *event a occurs during the occurrence duration of event b*, can not be found if events were considered as instantaneous point in time. In medical applications, a temporal pattern may be that patients frequently start having *asthma attack* when they start to *exercise* and these symptoms all occur when air pollution exceeds a certain threshold.

Very little attention has been paid to interval-events in mining sequential patterns. A well-known method in this direction is *TPrefixSpan* algorithm [146] that extends *PrefixSpan*

[1] `http://patternvision.com/`

algorithm to suit interval-event data. Another method by Kam and Fu [67] utilizes 13 Allen's temporal relationships [13] between any two interval events to mine events with multiple passes over data. After Allen, Freksa revisited interval relationships at the semi-interval level [46]. Semi-intervals allows a flexible representation where partial or incomplete knowledge can be handled since operations are on parts of an interval and not the whole. Freksa's semi-intervals are explained the basis of this work and explained thoroughly in chapter 4.2. Very little work on pattern discovery has focused on using semi-intervals. The most notable was completed by Morchen and Fradkin in [95] where they explored semi-intervals for use in unsupervised pattern mining.

3.1.3 Temporal Association Rules

While sequential patterns describe the concept of order in itemset sequences, temporal association rules combine traditional association rules with temporal aspects. Often the time stamps are explicitly used to describe the validity, periodicity, or change of an association.

Chen and Petrounias combine association rules with temporal constraints on the validity rules [29]. The temporal features are used to define and describe an absolute time, a periodical time, or a specific time. Ale and Rossi [12] used time for a more meaningful calculation of association rule support. The lifetime of an item is defined as the time between the first and the last occurrence. The temporal support is calculated with respect to this interval. This way rules are found that are only active during a certain time. Inter-transaction association rules [135] merge all itemsets within a sliding time window augmented with the occurrence times. The resulting rules can express occurrences with relative time differences. Based on that, a faster algorithm is proposed for this problem [63]. Özden et al. [108] brings the importance of temporal granularity to attention. The authors argue that when searching over several months, a co-occurrence of, e.g., coffee and doughnuts might be relatively low.

But by considering hourly sales, they might be more frequent in the morning hours. In their algorithm, the temporal granularity needs to be specified by the user.

3.1.4 Time Series Data Mining

Time series is sequence of a quantity obtained at successive times, often with equal intervals between them. Application of time series analysis techniques in temporal data mining is often called Time Series Data Mining. Pattern discovery in time series of continuous data can be performed by comparing time series and searching for similar shapes, according to some notion of similarity. This process has roots in prediction analysis, e.g. if one time series shows the same trend as another but with a known time lag, observing the trend of the latter allows assessments about the future behavior of the former.

The notion of the similarity between two time series, can be investigated by considering different time series representations. Many common representations support the approximate calculation of the Euclidean distance of the original time series. In some applications, time series are represented by symbolic values. In that case the selection of a distance measure shall be consistent with the selection of time series representation. [14] provides a comprehensive survey of temporal data representations and Similarity Measures for such data.

To discover similarities in time series one needs to only examine the data within a time window. Large windows imply a coarser and faster search. It, however, risks overlooking similar time series with small dissimilarities. A small window size implies a more detailed search with many more hits in the first step of the mining process. The minimum length threshold specifies how long the similar part of two time series should be, at a minimum, to consider them interesting. One motivation for similarity discovery in time series is the detection of common trends. These trends can be the common upward or downward moves. Such moves might shape a specific template. Thus, pattern discovery in time series data

can be converted to template matching problem in which the analyst specifies the shapes that should be looked for. Agrawal et al. define a Shape Definition Language (SDL) to describe patterns or shapes occurring in historical data [115]. The algorithm compares every two consecutive values in a time series and decides the movement direction in the interval between the values. Based on the SDL, authors define a query language for the specification of time series patterns and trends. [117] use a gradient alphabet to capture shapes and description of movement directions in time series. So instead of assessing the direction among consecutive values, the algorithm discovers sub-series conforming to the desired trend, which is expressed as a sequence of symbols from the alphabet.

3.1.5 Temporal Classification and Clustering

Classification is the most typical supervised learning technique. But it has not received much attention for temporal data mining tasks. Keogh et al. introduce a method that utilizes an extended representation of time series and a *merge* operator [68]. The representation consists of piece-wise linear segments to represent shape and a weight vector that contains the relative importance of each individual linear segment. In the classification context, the weights are learned automatically as part of the training cycle. The merge operator allows one to combine information from two sequences, and repeated application of the merge operator allows combining information from multiple sequences. The basic idea is that the merge operator takes two sequences as input, and returns a single sequence whose shape is a compromise between the two original sequences, and whose weight vector reflects how much corresponding segments in each sequence agree.

[65] develops a new technique for temporal classification of multivariate time series data. A feature construction technique is introduced that parameterises sub-events of the training set and clusters them to construct features, called meta-features, for a propositional learner.

Along the same direction [78] discovers frequent sub-sequences from temporal data and then using them, as the relevant features to classify sequences with traditional methods. However, treating the temporal aspects of data as static features and ignoring the temporal structure of the data generally looses the temporal semantics in the data.

The studies mentioned thus far concentrate on the classification of time series. For the classification of general temporal sequences, Zaki et al. propose *FeatureMine*, a feature extraction mechanism that serves as a pre-processor to a classification algorithm [78]. The goal of FeatureMine is to reduce the number of potentially useful features to be considered in the classification phase where a feature in this context is a sequence of items or itemsets. Weiss and Hirsh [139] propose a supervised learning technique to predict rare events in sequences. Their machine learning system, called *timeweaver*, uses an event sequence as training set to train a genetic algorithm-based miner and a test set to tune its performance. Timeweaver is used to predict hardware failure. The aim is to find a sequence of events that precede a failure and generate a prediction pattern from such sequences. Events in these sequences are subject to ordering and temporal constraints.

Ketterlin [69] proposes a hierarchical clustering method, COBWEB, to cluster temporal sequences databases. The algorithm first groups the elements of the sequences, and then groups the sequences themselves. Grouping the sequences in the second phase requires defining a generalization mechanism for sequences. Such mechanism has to be able to choose the most specific description for what is common to different sequences.

3.2 Temporal Reasoning

The development of temporal reasoning systems in AI has matured to the point that time is fully integrated into many industrial applications, primarily in planning and scheduling.

In most cases of temporal reasoning, time points are assumed to be the basic temporal entities. However, instantaneous points of time are not suitable to properly reason and extract knowledge about real-world events, which spans over a time interval. So in this section we review the literature on interval-based temporal logic and briefly introduce event calculus and situation calculus as a logic formalism designed for representing and reasoning about dynamical domains.

3.2.1 Interval-based Temporal Logic

One of the first formalisms of interval-based logical is Interval Temporal Logic (ITL), introduced by Moszkowski in [100]. ITL is like discrete linear-time temporal logic but includes time intervals. He proposed this logic to investigate the behavior of programs and hardware devices. An extension of ITL, called Duration Calculus (DC) [28], uses the integrated duration of states within 2 given interval of time to describe the requirements of real-time embedded systems. While Duration Calculus is one of the most popular and applicable interval-based logical formalisms, its semantics is essentially built on a point-based temporal characteristics. The most famous work in the formal study of purely interval-based temporal logic and reasoning in AI is by Allen [13]. He considers a family of binary relations, called Allen's relations, arising between two intervals in a given linear order. These relations are studied thoroughly in the next chapter.

Halpern and Shoham [56] introduce modal temporal logic based on time intervals, a logic which can be viewed as a generalization of point based modal temporal logic. They expressed all 13 Allen's relations between two distinct intervals by six modal operators $< B >$, $< E >$ and $< A >$ (for *begin*, *end*, and *after*) and their transposes $< \overline{B} >$, $< \overline{E} >$ and $< \overline{A} >$. In their semantics, time could be discrete, continuous, linear, or branching.

3.2.2 Event Calculus

An *event* represents an event or action in the real world. In event calculus terminology, the words *event* and *action* are used interchangeably. A *fluent* represents a time varying property of the word and a *timepoint* represents an instance of time. In event calculus, time is considered to be linear. An event may occur at a timepoint. A fluent has a truth value at a timepoint or within a time interval. Truth values are boolean (True or False). After the occurrence of an event, the truth value of the fluents might change. The system has commonsense knowledge about the effect of events on fluents. An example of this knowledge is: The event of picking up a plate initiates the fluent of holding the plate, and the event of setting down the plate terminates the fluent of holding the plate. Such knowledge is represented in first order logic as follows:

$HoldsAt(f, t) \equiv$ Fluent f is true at timepoint t.

$Happens(e, t) \equiv$ Event e occurs at timepoint t.

$Initiates(e, f, t) \equiv$ If event e occurs at timepoint t, then fluent f will be true after t.

$Teminates(e, f, t) \equiv$ If event e occurs at timepoint t, then fluent f will be false after t.

In order to illustrate how event calculus works, we use a simple example with the following one axiom:

$Happens(e, t_1) \wedge Initiates(e, f, t_1) \wedge (t_1 < t_2) \wedge \nexists e, t(Happens(e, t) \wedge (t_1 < t) \wedge (t < t_2) \wedge Teminates(e, f, t))) \Longrightarrow HoldsAt(f, t_2)$

The axiom means that if an event occur and it is known to initiate a particular fluent, then that fluent will be true from after the moment the event occurs, until the moment an event occurs that terminate the fluent.

In this approach, knowledge is represented as logic formulas declaratively rather than procedurally as computer codes. A collection of axioms that describe the commonsense domain is called *axiomatization*. Several types of reasoning maybe be performed using the event

Table 3.1: Event Calculus Reasoning Systems

Description	Technique	Reasoning Type
Event Calculus Planner [125]	Abductive Logic Programming	Abduction
Event Calculus Planner [126]	Propositional Satisfiability	Abduction
Discrete Event Calculus Reasoner [101][102]	Propositional Satisfiability	Deduction, Abduction, Postdiction
Discrete Event Calculus Theorem Prover [103][130]	First-order Logic	Deduction

calculus, such as deduction, abduction, and postdiction. Deduction consists of determining the state that results from performing a sequence of actions. For example given that a fan is set on a table and turned on, deduction determines that the fan is on the table and turning. Abduction consists of determining what events might lead from an initial state to a final state. Suppose that an initial state is that a person is inside the kitchen in the house and the final state is that the person is outside in the garden. Abduction will produce a set of actions in which the person walks out of kitchen, opens the door, and walks outside to the garden. Postdiction consists of determining the initial state given a sequence of events and a final state. For example, given that a person threw a ball and the ball hit the wall, Postdiction determines that the person was previously holding the ball.

Table 3.1 shows several systems that perform automated reasoning in the events calculus.

3.2.3 Situation Calculus

The situation calculus is a logical language for representing changes, especially change associated with actions. Situation Calculus has no explicit representation of time. The change, however, is represented via the quantification over states or situations. Situation Calculus was first introduced by McCarthy and Hayes in 1969 [91]. The basic concepts in the situation calculus are situations, actions and fluents. Situations are a sequence of actions. For example `do(goto(Frank), S0)`, where $S0$ is the initial situation. Actions are what make the dynamic world change from one situation to another. Fluents are situation-dependent functions used to describe the effects of actions and are of two kinds: relational fluents and functional fluents. The former have only true or false values while the latter can take a range of values. For instance, ¬`HasCoffee(Frank, S0)` which is true in a situation if the robots hand is not holding coffee.

To describe a dynamic domain in the situation calculus, one has to decide on the set of actions available for the agents to perform, and the set of fluents needed to describe the changes these actions will have on the world. For example, consider the classic blocks world where some blocks of equal size can be arranged into a set of towers on a table. We can have the following actions [104]:

- stack(x,y): put block x on block y, provided the robot is holding x, and y is clear, i.e. there is no other block on it;
- unstack(x,y): pick up block x from block y, provided the robots hand is empty, x is on y, and x is clear;
- putdown(x): put block x down on the table, provided the robot is holding x;
- pickup(x): pick up block x from the table, provided the robots hand is empty, x is on the table and clear.

To describe the effects of these actions, we can use the following relational fluents:

- handempty: true in a situation if the robots hand is empty;
- holding(x): true in a situation if the robots hand is holding block x;
- on(x,y): true in a situation if block x is on block y;
- ontable(x): true in a situation if block x is on the table;
- clear(x): true in a situation if block x is the top block of a tower, i.e. the robot is not holding it, and there is no other block on it.

Usually we have a set of special domain independent predicates and functions. `Poss(a,s)` is a special binary predicate with the intended interpretation that action a is executable in situation s. `Holds(p,s)` means fluent p is true in situation s. The function `do(a,s)` denoting the situation obtained by doing action a in situation s. Other special predicates and functions may be introduced based on the requirements of an application. For instance, to specify causal relations among fluents, one can use another ternary predicate `Caused(p, v, s)`, meaning that fluent p is caused to have truth value v in situation s.

Many researchers believe that when people remember the effects of an action, they do not explicitly store the facts that are not changed by the action, rather they just remember the changes that this action will bring about. Consequently, when we axiomatize an action, we should only need to specify the changes that will be made by the action. But if we specify only the changes that an action will make, there is a problem of how to derive those that are not changed by the action. This problem is called *the frame problem.* The frame problem is one of the most well-known AI problems and a lot of work has been done to solve it [120, 90, 81, 131].

3.3 Qualitative Models and Qualitative Reasoning

Wikipedia defines qualitative reasoning as an area of research within Artificial Intelligence (AI) that automates reasoning about continuous aspects of the physical world, such as space, time, and quantity, for the purpose of problem solving and planning using qualitative rather than quantitative information. In the literature, qualitative reasoning is mainly focused on explaining and predicting physical systems. It is assumed that the description (model) of the physical system is known, and given an initial state the output behavior of the system is produced. So there is an important distinction between model building and model simulation. Model-building starts with a description of a physical situation and builds an appropriate simplified model. Model-simulation, on the other hand, starts with a model and predicts the possible behaviors consistent with the model.

An early version of qualitative reasoning theory was formulated by Johan de Kleer [35] for analyzing an object moving on a roller coaster. Later more sophisticated forms were developed in parallel by de Kleer and Brown [36] for analyzing electronic circuits; by Forbus [44] for analyzing varieties of physical processes; and by Kuipers [133] as a mathematical formalism. Although traditionally understanding physical systems was the main focus, researchers extended the applications of qualitative reasoning to other domains such as education [24], automotive industry [127], and finance [57]. Qualitative reasoning literature has been more focused on qualitative simulation and model representation rather than the process of constructing qualitative models (e.g. qualitative modeling). In this section we discuss some ideas of qualitative abstraction and simulation. We also discuss qualitative data mining as an approach to build qualitative models from numerical data, and Q^2 learning which combines qualitative and quantitative approaches to construct models.

3.3.1 Qualitative Models

Classification models predict categorical classes and regression models can predict numerical quantities. Qualitative models, on the other hand, are models that describe qualitative relations between the observed variables. For instance: $M^+(x, y) \equiv y$ increases monotonically with x. Such models do not predict exact numerical values and their descriptions are close to the human way of thinking. Numbers are usually abstracted into symbolic values and intervals. For example instead of quantitative statement: water level at 11:59 is 52.4 cm, a qualitative abstraction is water level at $t1$ is $L1$. Adding another quantitative statement to the previous one: water level at 12:00 is 68.6 cm, a qualitative abstraction can show the direction of change as: water level increases within 1 hour from $t1$ to $t2$.

In some cases, exact numerical values and mathematical equations are not required to reason about complex situations. For example the numerical equation that measures the amount of river discharge is:

$$Y(t) = \int_{t_1}^{t_2} X(t-\tau)\lambda e^{-\lambda\tau} d\tau \tag{3.1}$$

Where $X(t)$ denotes the rainfall intensity changing with time, and λ is a constant depending on the cross sectional area of the river. However, "*Observing pouring rain and a river's steadily rising water level is sufficient to make a prudent person take measures against possible flooding - without knowing the exact water level, the rate of change, or the time the river might flood*[2]*"*. So this type of imprecise prediction suffices in many situations to allow people to react appropriately. Therefore such models are easier to explain and can reveal more information than regression models. Typically, they are also more robust since qualitative relations are simpler to model. Thus they are sometimes used as a step before

[2]Yumi Iwasaki

Figure 3.3: Qualitative reasoning in action

regression modeling, where the relations in the qualitative model are used as constraints for the regression model [129].

Not only humans but animals have a qualitative way of thinking and reasoning as well. For example when a dog catches a Frisbee, does it know numerical model of a flying Frisbee? Does it know physical concepts, aerodynamic lift and gyroscopic stability? Of course not. The dog fixes its gaze to the flying object and run towards that (as shown in Fig. 3.3). As Corey Hoffstein mentioned in his blog, *the art of modeling is in creating the simplest possible description of a complex phenomenon that still captures the salient features we are interested in*[3].

3.3.2 Qualitative Simulation

Qualitative simulation (QSIM)[73] generating a set of qualitative states from some given initial state, constituting predictions about possible future behaviors of the system. In QSIM, a physical system is described by a qualitative differential equation (QDE), which corresponds to an infinite set of ordinary differential equations. A QDE consists of a set of variables and a set of constraints on those variables. Possible constraints are $x + y = z$,

[3]https://blog.thinknewfound.com/2014/03/simple-by-design/

$xy = z$, $x = y$, $y = \frac{dx}{dt}$, $M^+(x, y)$, and $M^-(x, y)$, where x, y, and z are variables. The constraint $M^+(x, y)$ means that x is monotonically increasing with y, and $M^-(x, y)$ means that x is monotonically decreasing with y. Each variable is associated with landmark values $l_1 < ... < l_n$. A qualitative value of a variable consists of a landmark value l_i or interval (l_i, l_{i+1}) and a direction, which is increasing, steady, or decreasing. A state assigns a qualitative value to each variable.

Given an initial state and a QDE, the QSIM algorithm produces a behavior tree, or tree of states. The QSIM algorithm operates by taking a state and generating successor states that satisfy the constraints repeatedly. This process can be applied recursively, to derive all of the states that can follow from a given initial qualitative state. Generating all possible categories of behaviors is called envisioning. In general, envisioning is exponential in the number of constituents of the qualitative state. Qualitative reasoning is largely concerned with continuous change and does not support discrete events or indirect and non-deterministic effects of events. QSIM does not *learn* the model or behavior of the system since behavior is determined by predefined QEDs. As M^+ used in QSIM to formulate monotonicity constraints between continuous variables, multivariate monotonic function constraints by Wellman [140], or $\propto_Q$-all predicate suggested by Forbus [44] are other types of formulations for this purpose.

3.3.3 Qualitative Data Mining

Despite the interesting properties of qualitative models, there are not many algorithms to construct such models efficiently. In this section we first review some methods at very high level then we dig deeper into 3 algorithms that shown to be more efficient and effective in detecting qualitative patterns.

FS-QM [19] is a the hybrid approach that results from the integration of qualitative models and fuzzy systems. It uses QSIM [73] to derive qualitative behavior of the system, where

a domain expert supplies a QDE model. As fuzzy approximator, FS-CAD (Fuzzy Systems with Center-Average Defuzzifier) is chosen to map qualitative behavior into fuzzy rules. This approach is used to model metabolic system. The main drawback of this system is that QDE needs to be provided by a domain expert. There are several other approaches to learn qualitative models, most of which learn qualitative models in the form of QDEs that use qualitative relationships to describe dependencies among the system variables. For example MISQRT [119] uses heuristic techniques to break system's behaviors into segments and then learn multiple QDEs, corresponding to different operating regions. QMN [39] uses a simple search procedure to find QDEs that, within some tolerance, fit the data. So it learns a qualitative model from numerical data directly. Next we explain three qualitative data mining approaches that more attentions and demonstrated impressive results on real-valued experimental data.

QUIN

QUIN [128] is a learning algorithm that looks for qualitative patterns in numerical data. Induction of the so-called qualitative trees is similar to the well-known induction of decision trees (e.g. CART [25], C4.5 [118]). The difference is that in decision trees, the leaves are labeled with class labels, whereas in qualitative trees, the leaves contain *monotonic qualitative constraints* (MQC) that define qualitative constraints on the dependent variable. MQCs are a kind of monotonicity constraints that are widely used in the field of qualitative reasoning [44, 36]. Kuipers [73] and Forbus [45] give good overviews and discuss various abstractions of mathematical relations used in qualitative reasoning.

A monotonic qualitative constraint $M^{s_1,...,s_m}$ where $s_i \in \{+,-\}$ stands for an arbitrary relation between the class variable and m attributes, so that such a relation respects the qualitative constraints given by signs s_i . For example, consider the constraint $Y = M^{s_1,...,s_m}(X_1,...,X_m)$. A relation $(Y, X_1, ..., X_m)$ between class Y and m attributes $X_1, ..., X_m$ respects this con-

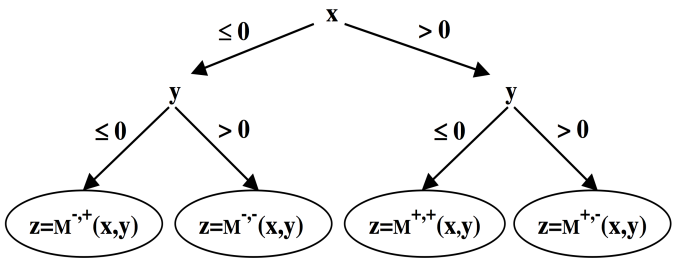


Figure 3.4: A qualitative tree induced from a set of examples for the function z = x2 - y2. The rightmost leaf, applying when attributes x and y are positive, says that z is strictly increasing in its dependence on x and strictly decreasing in its dependence on y [128]

straint if for all $i = 1, ..., m$, class Y is s_i-related to attribute X_i. It is defined that Y is "+"-related (positively related) to an attribute X if for all pairs (y_1, x_1) and (y_2, x_2) of values of X and Y in the projection of the relation on (Y, X): $x1 < x2 \rightarrow y1 < y2$. "Negatively related" is defined analogously. In general, MQCs can have more than one argument. For example, $Z = M^{+,-}(X, Y)$ indicates that Z monotonically increases with X and decreases with Y. If both X and Y increase, then according to this constraint, Z can increase, decrease, or stay unchanged. In such a case, a QCF cannot make an unambiguous prediction of the qualitative change in the Z variable. Fig. 3.4 shows an example of qualitative tree induced from a set of example points for the function $z = x^2 - y^2$.

QUIN algorithm takes as input a set of numerical examples and looks for qualitative patterns in the data. More precisely, QUIN looks for regions in the data space where monotonicity constraints hold. Such a set of qualitative patterns are represented in terms of a qualitative tree. QUIN constructs a tree in the top-down greedy fashion. At each internal node of the tree, QUIN considers all possible splits, that is conditions of the form $X \leq T$ for all the attribute variables X and all possible thresholds T with respect to X. Each such condition partitions the training data into two subsets. QUIN finds the best MQC for each subset according to an error-cost measure for MQCs.

Q^2

Q^2 is a machine learning approach that combines both qualitative and numerical learning [129]. First it uses QUIN algorithm to construct a qualitative model from numerical examples of the behavior of a physical system, then it uses a numerical regression function that respects the qualitative constraints and fits the training data numerically. Q^2 stands for "Qualitatively faithful Quantitative learning". Authors have shown that qualitative model's guidance of the quantitative modeling process leads to predictions that may be considerably more accurate than those obtained by state-of-the-art numerical learning methods.

After creating a qualitative tree using QUIN algorithm, given a set of numerical data and a qualitative tree, Qualitative-to-Quantitative (Q2Q) process attempts to find a regression function that fits the data well numerically, and also respects the qualitative constraints in the tree. This process is called reification of a MQC. For instance consider qualitative constraint $y = M^+(x)$. In order to find a numerical fit, one intuitive solution is to divide the range of variable x with a number of equidistant points (i.e., $\{x_1, x_2, ..., x_n\}$) from which function values y need to be learned. The result is a set of pairs $\{(x_1, y_1), ..., (x_n, y_n)\}$ that defines a piece-wise linear function which can be easily checked for compliance with the given qualitative constraint. One problem is that Q2Q approach described above does not consider the continuity and smoothness between the regression functions that belong to adjacent leaves of a qualitative tree. This causes sharp changes at the borders between leaves of a qualitative tree. Figure 3.5 shows the data and learned qualitative tree. Also the Q2Q-learned regression functions are demonstrated based on two different qualitative explanations of the data. Left, the case with a three leaf qualitative tree; right, the case with a single leaf qualitative tree saying $y = M^-(x)$.

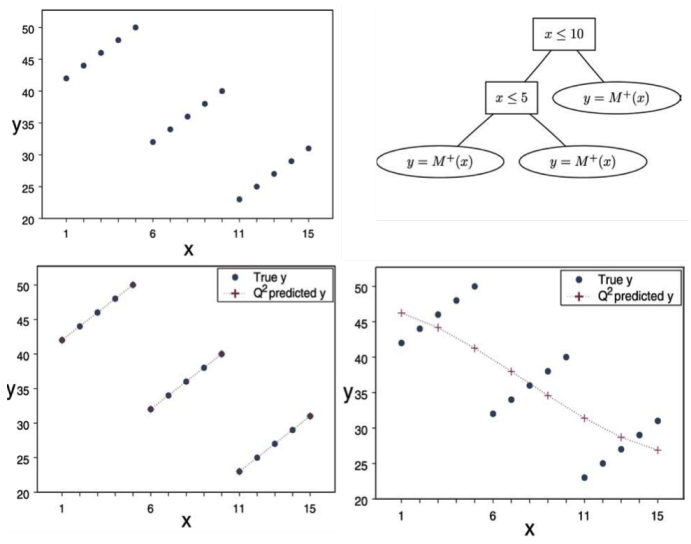


Figure 3.5: The graphs present the data and the Q2Q-learned regression functions based on two different qualitative explanations of the data. Left, the case with a three leaf qualitative tree; right, the case with a single leaf qualitative tree saying $y = M^-(x)$ [129].

Padé

Researchers have found that Q^2 algorithm becomes very slow with the growing number of dimensions on real-world data [150]. Also, it cannot treat time dimension as separate variable from other variables, which decreases its suitability for modeling dynamic systems. Also it is limited to the construction of tree models which is not necessarily the best representation for many practical problems. To solve this issue, an algorithm named Padé [149] is developed based on the approximation of partial derivatives of the sampled multidimensional function. The algorithm computes the derivative at each point where the function is sampled using the points at its vicinity which is defined either by triangulation or by the axis in which direction the derivative is computed. To proceed with qualitative modeling, one can observe only the signs of derivatives. The data pre-processed in this way can be subsequently modeled by any general machine learning algorithm. However, neither QUIN nor Padé algorithm are unable to treat discrete variables. Also an important difference between the algorithms explained previously and Padé is that Padé is a pre-processor while other algorithms produce a model.

Padé output is a data set which can later be used by appropriate algorithms for classification, regression, or visualization.

Chapter 4

Data Model and Pattern Operators

In general, the term *event* has been used in two distinct contexts in the literature. The first relates to the physical world occurrences while the second involves representations of those occurrences in a computer system. However, either of the two "event"s can not be transformed from one to another directly. In this section, the definitions of an event, event stream and event instance are provided. Moreover, the properties of different events and their classifications are discussed in terms of an event model. Since temporal properties of an event are essential in knowledge discovery from time-series data, we propose a time model based on semi-intervals. As suggested in chapter 1, model building and knowledge discovery is the process of understanding relations between underlying events in a system. To formulate such relations, the concept of patterns and pattern formulation operators are discussed as well.

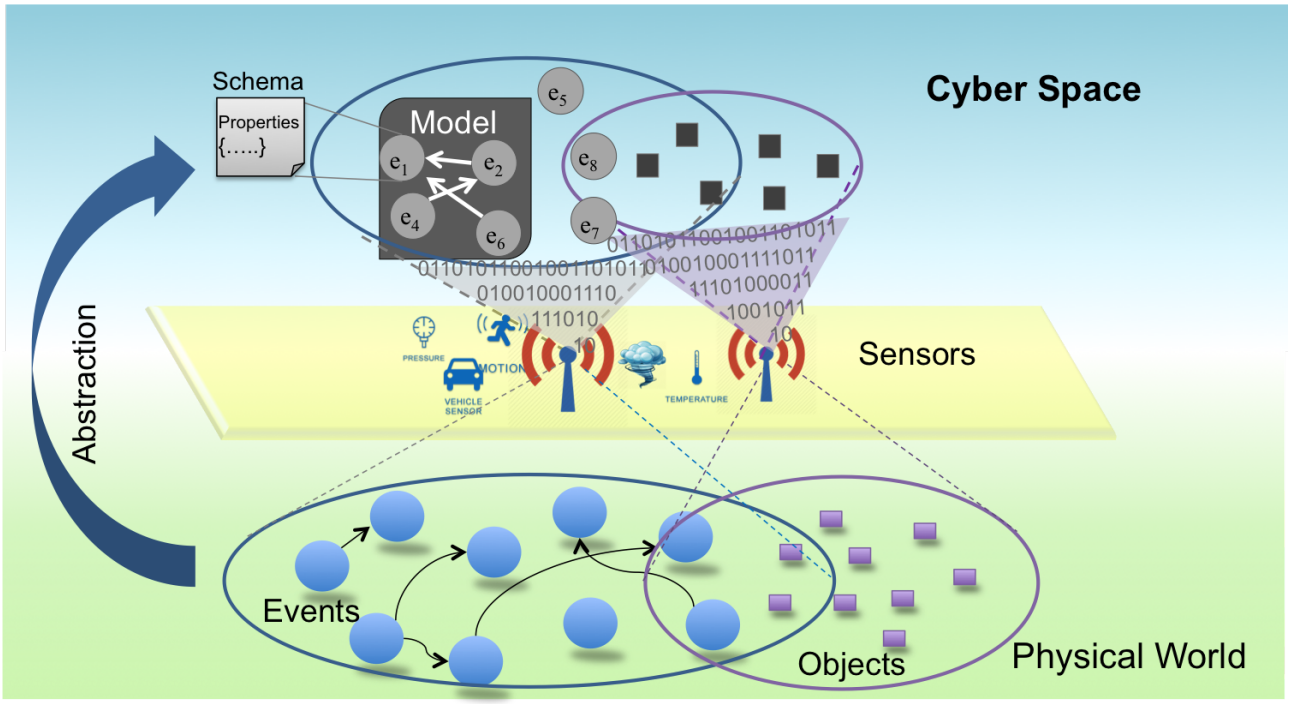


Figure 4.1: Interaction between physical world and cuber world. Sensors act as interface between these two world. Objects and events are recognized in cyber world and effective models are built by understanding relations between cyber events.

4.1 Physical-World vs. Cyber World

In this dissertation, we are interested in two problem universes among all the possible universes. One is the physical world, which mainly consists of events and objects that interact and impact each other. The other one is cyber world which is the digital extension and abstraction of real world in a virtual environment. The mapping and interconnecting objects and events between these two worlds is the foundation of cyber-physical systems. Objects in the physical world indicate concrete things with tangible bodies (e.g. a person, vehicle, tables). Physical events generally consists of interaction between objects, human activities/behaviors, and certain causation indicating event happenings, which are triggered by certain conditions in the physical world. The cyber objects and events are abstraction of those entities from physical world. Sensors act as an interface between these two world and make the abstraction process possible.

4.1.1 Events Perception in Human

The inputs to hearing, vision, and the other senses are continuous, dynamic, and comprise huge amounts of data each second. However, human conscious experience seems to resolve into a manageable number of more-or-less discrete entities, most prominently *objects* and *events*. The term **event perception** encompasses a range of cognitive techniques involving the processing of temporally extended dynamic information [52]. In this process, our brain picks up intervals of time and distinguishes them from other intervals to form meaningful events. Moreover, Our brain tends to automatically seek patterns (relations among events) [138]. With this capability, our brain consolidates multiple events and their relations as a piece of *memory* or *knowledge*.

4.1.2 Events in Cyber World

Events in cyber world provide a natural abstraction of happenings in the real world. They are encoded in the entities that we all have created and shared over cyber space. For example, images of a concert we recently attended, an interesting location we visited during an overseas trip a long time ago, video of an important game, or eye- witness descriptions and infrared satellite images of a devastating tornado that made it to the headlines. All this content comes in different types of media (images, videos, text, etc.) using different data modalities and was created by different sensors. What is common though for all these media items is that they all have captured and convey information about a real-life event.

In various disciplines, information about an underlying phenomenon might be acquired from different types of sensors. Rarely a single modality can provide complete knowledge of the phenomenon of interest due to rich characteristics and complexity of that phenomenon. In order to extract insight from this data a major challenge arise: How to fuse these modalities into a human understandable abstraction signals that not only preserve the semantics of the

underlying system but also facilitates data analysis? Also, the volume and types of data pose new data management challenges. The data from different sources usually result in different silos that do not communicate with each other and are indexed using data-centric approaches. In the current form it's not possible to make sense of these diverse sources of data. We believe that by organizing all these data around meaningful events, one can overcome heterogeneity and variety issues in the data.

Event models help understanding underlying heterogeneous real-life events from the multimodal content and using them in order to better organize, retrieve and discover knowledge in any possible way. Event models assign different properties to an event. Different information and observation sources help in interpretation and population of event properties. For example, exercise event can have multiple properties such as heart rate, calorie consumption, speed, and distance traveled or number of steps climbed. Some of these properties can be recognized from motion sensors (e.g. accelerometer), and some can be recognized from physiological sensors (e.g. heart rate monitoring).

Based on different data streams and models of different events, we can segment timeline into event structure and associate informational and experiential data. So events are **structuralization** of the timeline using semantics rather than the uniform structuralization as imposed by calendars.

The necessity of formal event models for the description of real life events has been acknowledged, and a number of such models have been developed [3]. In summary, events can be characterized by six different aspects [124]: time, space, participation, relations between events (in terms of mereology, causality, and correlation), documentation, and interpretation. Scherp et al. introduced these aspects and analyzed to which extent the existing event models support them. Comparison of Event Models and Event Aspects are displayed in table 4.1.

Table 4.1: Comparison of event models and event aspects [124]

Event Model	Time	Space	Structural Relationships			Information\ Participation	Experience
			Sub-event	Correlation	Causality		
Event Calculus	✓	x	✓	✓	✓	x	x
Situation Calculus	✓	✓	o	x	✓	x	x
Event Ontology	✓	✓	o	x	o	x	x
SEM	✓	✓	o	x	x	x	x
SOUPA	✓	✓	o	x	o	✓	✓
SAWA	✓	✓	✓	x	o	✓	x
CIDOC CRM	✓	✓	o	x	o	✓	x
SsVM	✓	✓	✓	x	o	✓	x
CASE^E	✓	✓	o	x	✓	✓	x
VERL and VEML	✓	✓	o	x	o	✓	x
N. Gkalelis et al.	✓	✓	o	x	✓	✓	o
Eventory	✓	✓	✓	x	o	✓	o
E	✓	✓	✓	x	✓	✓	✓
E*	✓	✓	✓	x	✓	✓	✓
Event-model-F	✓	✓	✓	✓	✓	✓	✓

4.1.3 Bridge the Semantic Gap

As discussed, for humans, event concept is a basic function of human brain to process the sensory information about real-world, produce knowledge about them, and react properly to the outside world. In the cyber space, event is a basic building block to capture knowledge and provide a natural abstraction of happenings in the real world. To bridge the gap between flow of information in real world and how we capture, manage, and process information in cyber world we define an event model that capture multiple aspects of an event and create an event media JSON. So event is not merely a symbol or tag, but it is a **datum** with multiple facets and properties. Figure 4.2 shows a sample event stream and multiple events. Each event has a model with general event facets and tailored event properties.

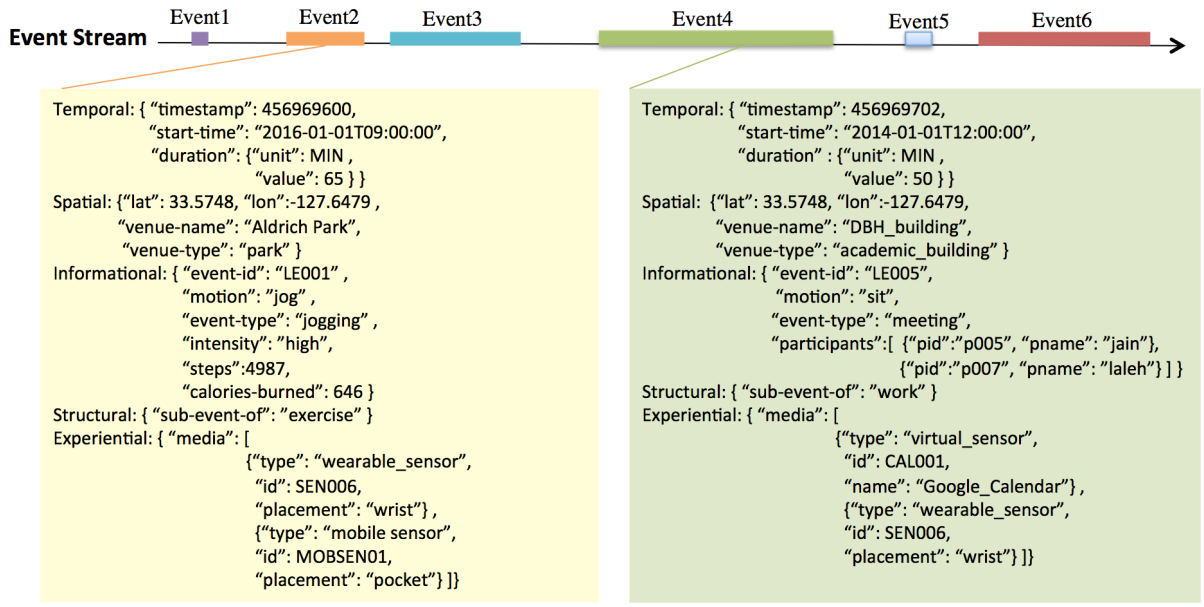


Figure 4.2: Sample event media JSON for jogging and meeting events. Although events have general temporal, spatial, informational, structural, and experiential facets, their informational properties varies between different events.

4.2 Time Model

For the purpose of temporal reasoning, Allen formalized temporal logic on intervals by specifying 13 interval relations [13] and showing their completeness. Any two intervals are related by exactly one of the relations. The operators are: *before, meets, overlaps, starts, during, finishes*, the corresponding inverses *after, met by, over- lapped by, started by, contains, finished by, and equals* (see Figure 4.3). These temporal relations have been used by the majority of research on mining time-interval data [94, 96, 67, 31]. However, researchers identified problems using Allen's relations. The relations are not robust to noise because small shifts of time points lead to different patterns describing similar situations observed in the data. Also the pattern representation is ambiguous because the same pattern can describe quite different situations in the data [97, 146]. After Allen, Freksa revisited interval relationships at the semi-interval level [46]. He generalized the interval relations by using semi-intervals with the following 11 operators: *older, younger, head to head, survives, survived by, tail to tail,*

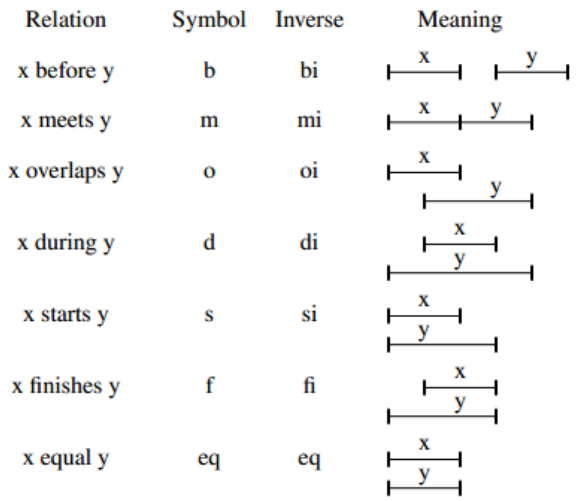


Figure 4.3: Allens interval relations between the intervals X and Y.

precedes, succeeds, contemporary, born before death, and *died before birth*. Semi-intervals allow a flexible representation where partial or incomplete knowledge need to be handled since operations are on parts of an interval and not the whole.

In temporal data mining, very little work has focused on using semi-intervals. The most notable was conducted by Morchen et al. in [95] where they explored semi-intervals for unsupervised pattern mining and proved that semi-interval patterns are more flexible than patterns over full intervals. In our work, we use semi-interval to represent temporal facet of an event. We adopt a time point representation of time intervals [146] in which intervals are represented with their start and end time point.

Definition (*Time Domain*). A time domain $\mathbb{T}$ is a discrete, linearly ordered, countably infinite set of time instants $t \in \mathbb{T}$. We assume that $\mathbb{T}$ is bounded in the past, but not necessarily in the future.

Definition (*Time Interval*). A time interval is a triple $[\partial, t_s, t_e]$ where $\partial \in \Sigma$ is a unique symbol and $t_s, t_e \in \mathbb{T}$ and $t_s \leq t_e$. The finite set of all time intervals is noted $I = \{\ [t_s, t_e] | t_s \leq t_e\ \}$. If $[t_s, t_e] \bigcap [t'_s, t'_e] \neq \varnothing$, intervals are overlapped.

Relation	Label	Inverse	Illustration
X is *older* than Y Y is *younger* than X	ol	yo	XXX???? YY
X is *head to head* with Y	hh	hh	XXX?? YYYY
X *survives* Y Y is *survived by* X	sv	sb	????XXX YY
X is *tail to tail* with Y	tt	tt	??XXX YYYY
X *precedes* Y Y *succeeds* X	pr	sd	XXX? YYY
X is a *contemporary* of Y	ct	ct	?XXX??? ???YYY?
X is *born before death of* Y Y *died after birth of* X	bd	db	XXX????? ?????YYY

Figure 4.4: Eleven semi-interval relationships. Question marks (?) in the pictorial illustration stand for either the symbol denoting the event depicted in the same line (X or Y) or for a blank. The number of question marks reflects the number of qualitatively alternative implementations of the given relation [46].

Definition (*Semi Interval*). A semi interval is a tuple $[\partial^{+/-}, t]$ where $\partial \in \Sigma$ is a unique symbol and interval boundaries are represented with + and signs where ∂^+ and ∂^- are corresponding to the start and end of the interval respectively. $[\partial^+, t]$ represent a semi interval where start time is available and $[\partial^-, t]$ represent a semi interval where end time is available. Using the semi-interval definition, an interval can also be represented with its interval boundaries. Formally in a time interval $[\partial, t_s, t_e]$, $\partial.t_s = \partial^+$ and $\partial.t_e = \partial^-$ and duration of the interval is $d(\partial) = \partial^- - \partial^+$. Also an instantaneous time point can be considered as an interval with zero duration where $\partial^- = \partial^+$.

The most important criteria to be considered about the relation between events in a pattern are:

1. The events occur in a particular order with specific time lag in between.

2. The order is not important but there is a notion of concurrency between events as they occur in parallel.

We define two temporal relations to meet these requirements. Then we use semi-interval operators to represent these temporal relations. The relations as shown in Table 4.2 are:

1. Order: is the sequential occurrence of time points or time intervals. In Freksa's formalism *younger*, *succeeds*, *survives*, and *born after death* relations fall in this category.

2. Concurrency: is a nonempty time period where two or more temporal events occur in no particular order. In Freksa's formalism *head to head*, *tail to tail*, and *contemporary* relations fall in this category.

Table 4.2: Semi-interval relations

Temporal Relation	Temporal Operation	Symbol	Semi-interval Representation	Graphical illustration
Order	Y is *younger* than X	yo	$X^+ < Y^+$	XXX???? YY
	Y *survives* X	sv	$X^- < Y^-$	??XX ??YYY
	Y *succeeds* X	sd	$X^- < Y^+$	XXX? YYY
	Y *died after birth* of X	dab	$X^+ < Y^-$	XXX????? ?????YYY
Concurrency	Y is *head to head* with X	hh	$X^+ = Y^+$	XXX?? YYYY
	Y is *tail to tail* with X	tt	$X^- = Y^-$	??XXX YYYY
	Y is *contemporary* of X	ct	$X \cap Y \neq \emptyset$	?XXX??? ???YYY?

4.3 Event Model

Event model serves as a basis for the pattern language we define in the next section. An event is either an instantaneous occurrence or spans over time. The former is called point

event and the latter is called interval event. Sometimes we have a semi-interval event where information about the event is not complete. For example we might know when event starts but there no information is available about when it ends or vice versa. Event model has a schema ξ that describes a set of attributes a class of events must contain.

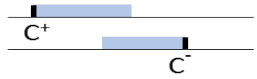
Definition 3 (*Point Event*). A point event (pE) is an event that occurs at an instantaneous point in time. It is a tuple $e = (v, [E, t])$ consists of the name of its type, denoted by an upper-case letter (e.g., 'E'), the time of occurrence $t \in \mathbb{T}$, and a set of values $v \in \xi$.

Definition 4 (*Interval Event*). An interval event (iE) is an event that spans over time. It is a tuple $e = (v, [E, t_s, t_e])$ consists of the name of its type, start and end time $[t_s, t_e] \in I$, and a set of values $v \in \xi$.

Definition 5 (*Semi-interval Event*). A semi-interval event (sE) is a special case of interval event when one of the event boundaries is missing. It is a tuple $e = (v, [E^{+/-}, t])$ consists of the name of its type, start time or end time $t \in \mathbb{T}$, and a set of values $v \in \xi$.

These three categories of events with their graphical illustration are shown in Table 4.3. Each event has a start time value t_s or an end time value t_e or both. Events with complete time interval are represented as (E, t_s, t_e), while events with semi-interval are represented as (E^+, t) when t_s is available, and (E^-, t) when t_e is available. Point events are represented as (E, t). The reason that we differentiate between semi-interval event and point event goes to different semantics of events in different application domains. Semi-interval event is an interval event when a partial knowledge or observation is available. For example in healthcare domain when a patient 'experience symptoms' after taking a medication, this process is an interval event with start and end time, but we might only know when the symptoms start. So experiencing symptoms is recorded as a semi-interval event with the start time associated with it. In the same context, 'taking a pill' event is a point event. Though we can imagine that the duration of this event is several seconds, but with respect

Table 4.3: Three event categories with their data model and graphical illustration.

Category	Event Model	Time Model	Graphic
Point Event (pE)	(A , t)	$A^+ = A^-$	
Interval Event (iE)	(B, t_s, t_e)	$B^+ = B.t_s$ $B^- = B.t_e$	
Semi-interval Event (sE)	($C^{+/-}$, t)	If $C.t_s \neq$ NA then $C^+ = C.t_s$ If $C.t_e \neq$ NA then $C^- = C.t_e$	

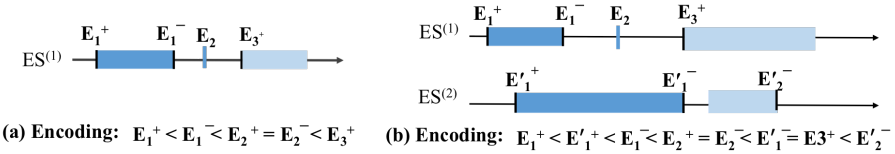


Figure 4.5: (a) Example encoding of a sequence of events. E_1^+ and E_1^- represent the start and end times of event E_1, respectively. Relational operators are used to indicate the ordered relations between start/end times. (b) Example of encoding a multi-event stream from two sequence of events.

to the basic time granularity of an application (e.g. 15 minute or 1 hour), 'taking a pill' is considered as a point event.

Definition 6 (*Event Stream*). An event stream $ES^{(i)} = \{e_1^{(i)}, e_2^{(i)}, ..., e_n^{(i)}\}$ is an ordered set of events where $e_k \in \{pE, iE, sE\} (1 \leq k \leq n)$.

Definition 7 (*Multi-Event Stream*). A multi-event stream $ES = \{ES^{(1)}, ES^{(2)}, ..., ES^{(|I|)}\}$ is a finite set of event streams. Events from multiple event streams have a total order based on their start time so they can be overlapped. The alphabet of multi-event stream is $\Sigma = \{\Sigma^{(1)} \bigcup \Sigma^{(2)} \bigcup ... \bigcup \Sigma^{(|I|)}\}$.

Take an example from Fig. 4.5(a), $ES^{(1)}$ is an event stream with 3 disparate events. E_1 is an interval event represented with E_1^+ and E_1^- for start and end points, E_2 is a point event where $E_2^+ = E_1^-$, and E_3 is a semi-interval event represented with E_3^+ for start point. The events that belong to the same event stream cannot have overlaps. However, as shown in Fig. 4.5(b), events from multiple event streams might be overlapping. This brings us to the next question: How to encode a multi-event stream with overlapping events so the ordering and temporal structures between events are preserved? As explained before, we seamlessly represent semi interval and interval events using instantaneous time points in the interval boundaries. As done in [67], the relationships between events' start and end times can represent all of Allen's principles hence encompassing all relational ordering possibilities. Also all Allen's interval relations can be described using Freksa's semi-interval relations. A simple schema similar to that of [67] can be used to encode a multi-event stream in a single linear data stream (Fig. 4.5) without any ambiguity with respect to the temporal relation between overlapping events. Event E_1 ends before E_2 occurs ($<$), or an event E_1' ends at the same time E_2 starts ($=$).

4.4 Hypothesis-Driven Pattern Operators

We formally define our language by defining an algebraic operation for pattern formulation. These operators are aimed to be the basic operations, combination of which can be used for arbitrarily sophisticated pattern formulation and pattern querying on event streams.

To begin with, each event type $E_i \in \{pE, iE, sE\}$ is a pattern expression. The semantic of base expression for point events pE, represented as $E_i(t)$, is at a given time point t, $E_i(t)$ is true if E_i occurs at time point t. The semantic of base expression for interval events, represented as $E_i(t_s, t_e)$, is at a given time interval $T = [t_s, t_e]$, $E_i(t_s, t_e)$ is true if E_i starts at t_s and ends at t_e. The semantic of base expression for semi-interval events, represented as

$E_i^{+/-} = E_i^{+/-}(t)$, is at a given time t, $E_i^{+}(t)$ is true if E_i starts at t, and $E_i^{-}(t)$ is true if and E_i ends at t.

Arbitrary patterns can be defined by applying operators on individual event types. So each pattern consists of a number of participating events. Suppose pattern ρ has k participating events E_i , where $1 \leq i \leq k$, size of the pattern denotes $|\rho| = k$, start and end timestamps of the pattern denotes $\rho.t_s$ and $\rho.t_e$ respectively and defined as follows:

$$\rho^{+} = \rho.t_s = \{E_1.t_s, E_2.t_s, ..., E_k.t_s\} \qquad \rho^{-} = \rho.t_e = \{E_1.t_e, E_2.t_e, ..., E_k.t_e\} \tag{4.1}$$

Definition 8 (*Pattern*). A pattern ρ is represented as $\rho = (X_1 \odot_1 X_2 \odot_2 ... \odot_{k-1} X_k)$ where X_i is an event, $X_i \in \{pE, iE, sE\}(1 \leq i \leq k)$, and $\odot_i \in \{;, ,, \omega_{\Delta t}, \Vdash, |\}(1 \leq i \leq k-1)$ is a binary operation. In case there are multiple occurrences of a X_i in a pattern, it is necessary to distinguish which two event boundaries (e.g. X_i^{+} and X_i^{-}) represent the same X_i occurrence. However, we assume that each event type has only one occurrence in a pattern and X_i^{+} and X_i^{-} are coming from the same occurrence of X_i. Our language supports a hierarchy of complex patterns by feeding the output of one operator as an input to another. So a pattern operator not only connects events but also connects a number of pattern expressions to form a new expression. We now consider what it means to say a pattern *occurs* in an event stream. Intuitively, the event types of the pattern need to have corresponding events in the stream such that the event types are the same and the operations between events of the pattern are respected and satisfied. The semantic of pattern expression, represented as $\rho(t_s, t_e)$ or $\rho(T)$ at a given time interval $T = [t_s, t_e]$, $\rho(t_s, t_e)$ is true if pattern ρ has an occurrence in time interval T (e.g. starts at t_s and ends at t_e). Frequency of a pattern is the number of occurrences of the pattern in an event stream.

As shown in equation 3.1, a pattern has both start time and end time. So the temporal aspect

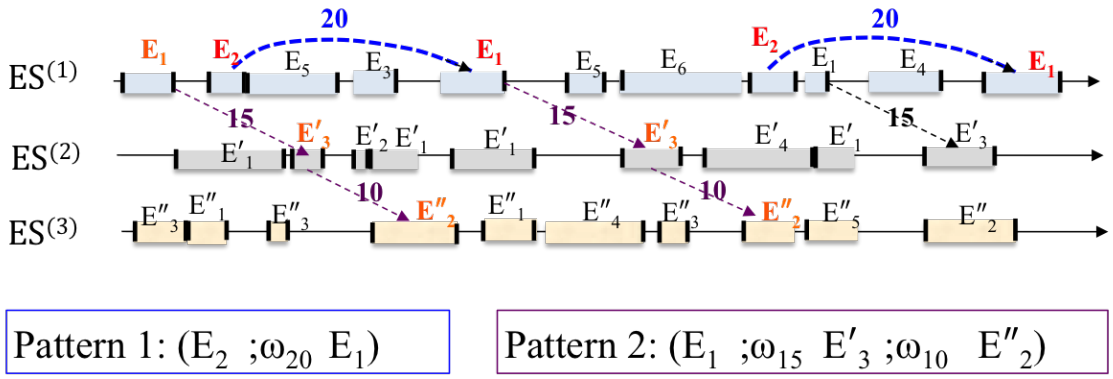


Figure 4.6: Sample event streams $ES^{(1)}$, $ES^{(2)}$, and $ES^{(3)}$ and their corresponding event types. Pattern 1 and 2 are conditional sequential patterns, each one with two occurrences.

of a pattern is a complete time interval. Two order relations can be defined on complete intervals:

Definition 9. A partial order $\prec$ is defined as follows:

$\forall\ T, T' \in I, T \prec T'$ iff $t_e < t'_s$

Definition 10. A total order relations ¡ is defined as follows:

$\forall\ T, T' \in I, T < T'$ iff $t_s < t_s \vee (t_s = t'_s \wedge t_e < t'_e)$

4.4.1 Selection Operation $(\rho.P)$

This operator filters pattern expression on predicate P, where P refers to event attributes contained in the pattern.

$(\rho.P) \equiv \exists\ T = [t_s, t_e]$ such that $\rho(T) \wedge P$

4.4.2 Sequence Operation $(\rho_1; \rho_2)$

Sequence Operation $(\rho_1; \rho_2; ...; \rho_k)$: This operator detects if pattern expression ρ_1 is followed by pattern expression ρ_2 and so on. The operator specifies a particular order in which the patterns of interest should occur. Formally it defines as follow:
$(\rho_1; \rho_2; ...; \rho_k) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}], ..., T_k = [t_{ks}, t_{ke}]$ such that $T_1 < T_2 < ... < T_k$, $\rho_1(T_1) \wedge \rho_2(T_2) \wedge ... \wedge \rho_k(T_k)$.
Considering different temporal relations in Figure 4.4, the sequence operation can have 4 sub-categories:

- $(\rho_2$ yo $\rho_1) \equiv (\rho_1^+; \rho_2^+) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}]$ such that $T_1 < T_2$, $\rho_1(T_1) \wedge \rho_2(T_2) \wedge t_{1s} < t_{2s}$
- $(\rho_2$ dab $\rho_1) \equiv (\rho_1^+; \rho_2^-) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}]$ such that $T_1 < T_2$, $\rho_1(T_1) \wedge \rho_2(T_2) \wedge t_{1s} < t_{2e}$
- $(\rho_2$ sd $\rho_1) \equiv (\rho_1^-; \rho_2^+) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}]$ such that $T_1 < T_2$, $\rho_1(T_1) \wedge \rho_2(T_2) \wedge t_{1e} < t_{2s}$
- $(\rho_2$ sv $\rho_1) \equiv (\rho_1^-; \rho_2^-) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}]$ such that $T_1 < T_2$, $\rho_1(T_1) \wedge \rho_2(T_2) \wedge t_{1e} < t_{2e}$

4.4.3 Conditional Sequence Operation $(\rho_1\ ;\omega_{\Delta t_1}\ \rho_2)$

Conditional Sequence Operation $(\rho_1\ ;\omega_{\Delta t_1}\ \rho_2\ ;\omega_{\Delta t_2}\ ...\ ;\omega_{\Delta t_{k-1}}\ \rho_k)$: It detects if pattern expression ρ_1 is followed by pattern expression ρ_2 *within* Δt time units. Δt is called the time lag or temporal restriction between two successive patterns. Formally it defines as follow:
$(\rho_1\ ;\ \omega_{\Delta t_1}\ \rho_2\ ;\ \omega_{\Delta t_2}\ ...\ ;\ \omega_{\Delta t_{k-1}}\ \rho_k) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}], ..., T_k = [t_{ks}, t_{ke}]$ such that $T_1 < T_2 < ... < T_k$, $\rho_1(T_1) \wedge \rho_2(T_2) \wedge ... \wedge \rho_k(T_k)$, $T_2 - T_1 \leq \Delta t_1 \wedge T_3 - T_2 \leq \Delta t_2 \wedge ... \wedge T_k - T_{k-1} \leq \Delta t_{k-1}$.

- $(\rho_1^+;\omega_{\Delta t}\rho_2^+) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}]$ such that $T_1 < T_2$, $\rho_1(T_1) \wedge \rho_2(T_2)$, $t_{2s} - t_{1s} \leq \Delta t$.

- $(\rho_1^+;\omega_{\Delta t}\rho_2^-) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}]$ such that $T_1 < T_2$, $\rho_1(T_1) \wedge \rho_2(T_2)$, $t_{2e} - t_{1s} \leq \Delta t$.

- $(\rho_1^-;\omega_{\Delta t}\rho_2^+) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}]$ such that $T_1 < T_2$, $\rho_1(T_1) \wedge \rho_2(T_2)$, $t_{2s} - t_{1e} \leq \Delta t$.

- $(\rho_1^-;\omega_{\Delta t}\rho_2^-) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}]$ such that $T_1 < T_2$, $\rho_1(T_1) \wedge \rho_2(T_2)$, $t_{2e} - t_{1e} \leq \Delta t$.

4.4.4 Concurrency Operation ($\rho_1 \between \rho_2$)

Concurrency Operation ($\rho_1 \between \rho_2 \between ... \between \rho_k$): Concurrency detects multiple patterns occur in parallel, and succeeds only if all patterns are detected. Unlike sequence, any order is allowed, and there has to be a non-empty overlap interval among the patterns.
$(\rho_1 \between \rho_2 \between ... \between \rho_k) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}], ..., T_k = [t_{ks}, t_{ke}]$, $\rho_1(T_1) \wedge \rho_2(T_2) \wedge ... \wedge \rho_k(T_k)$, $(T_1 \bigcap T_2 \bigcap ... \bigcap T_k) \neq \phi$.

4.4.5 Alternation ($\rho_1 \quad | \quad \rho_2$)

Alternation ($\rho_1|\rho_2|...|\rho_k$): This operator detects if any pattern expressions ρ_1 to ρ_k matches the input event stream.
$(\rho_1|\rho_2|...|\rho_k) \equiv \exists\ T_1 = [t_{1s}, t_{1e}], T_2 = [t_{2s}, t_{2e}], ..., T_k = [t_{ks}, t_{ke}]$, $\rho_1(T_1) \vee \rho_2(T_2) \vee ... \vee \rho_k(T_k)$.

4.4.6 Time ($\omega_{\Delta t} \quad \rho$)

Time ($\omega_{\Delta t}\rho$): This operator requires a pattern ρ to occur within a certain time interval$\Delta t = [\delta_1, \delta_2]$.

$\omega_{\Delta t} \quad \rho \equiv \exists\ T = [t_s, t_e]$ such that $\rho(T) \wedge \delta_1 \leq t_s \leq \delta_2$.

4.5 Data-driven Operators

The following co-occurrence operators are calculated for every pair of event types in the system. Given two event streams ES and ES' the co-occurrence values are computed between every pair of event types in these streams. If ES contains N event types E_1 to E_N, and ES' contains M event types E'_1 to E'_M, there are $N \times M$ co-occurrence pairs without considering time lags between events. Notice that we use lower case e for event model (considering different properties and temporal information), and upper case E for event type, event symbol, or event tag.

4.5.1 Sequential Co-occurrence $SEQ\_CO_{[\Delta t]}(ES, ES')$

The inputs to this operator are two event streams ES, ES' and a set of pre-defined time resolutions and time lags, and the output is a matrix with the co-occurrence value between each pair of event types in these event streams.

$$SEQ\_CO_{[\Delta t]}(ES, ES') = M_{NxM} \in \mathfrak{R}^{+}_{NxM} \tag{4.2}$$

If $ES = ES'$ it indicates auto co-occurrence on an event stream, otherwise it indicates cross co-occurrence between two event streams.

For a pair of events E_i and E_j, $M_{ij} = Seq\_Co_{[\Delta t]}(E_i, E_j)$, where sequential co-occurrence is defined as follow:

Sequential Co-occurrence: For a pair of events E_i and E_j, sequential co-occurrence with temporal offset Δt is the frequency of E_j follows E_i within Δt time lag.

$$Seq\_Co_{[\Delta t]}(E_i, E_j) = \frac{Count(E_i; \omega_{\Delta t} E_j)}{Count(E_i)} \tag{4.3}$$

with $Seq\_Co_{[\Delta t]}(E_i, E_j) \in [0,1]$. The maximum $|Seq\_Co[\Delta t](E_i, E_j)| = 1$ means that E_i and E_j are always co-occurring within Δt time lag, while values close to zero indicate that there is no co-occurrence within the specified time lag. To check for a significant time lag between E_i and E_j we define:

$$\Delta t^{max} = \underset{\Delta t}{\operatorname{argmax}} \big(Seq\_Co_{[\Delta t]}(E_i, E_j) \big)$$

with $\Delta t \in \{1, 2, ..., \lambda_{CO}\}$ where λ_{CO} is a design parameter indicating the maximum possible time lag between events.

4.5.2 Concurrent Co-occurrence $CON\_CO(ES, ES')$

The inputs to this operator are two event streams and the output is a matrix with the concurrent co-occurrence value between each pair of event types in these event streams.

$$CON\_CO(ES, ES') = M_{NxM} \in \Re^{+}_{NxM} \tag{4.4}$$

For a pair of events E_i and E_j, $M_{ij} = Con\_Co_{[\Delta t]}(E_i, E_j)$, where concurrent co-occurrence is defined as follow:

Concurrent Co-occurrence: For a pair of events E_i and E_j, concurrent co-occurrence is the frequency count of E_i and E_j occur with no particular order while $[E_i.t_s, E_i.t_e] \bigcap [E_j.t_s, E_j.t_e] \neq \varnothing$.

$$Con\_Co(E_i, E_j) = \frac{Count(E_i \Vdash E_j)}{\frac{1}{2}(Count(E_i) + Count(E_j))} \tag{4.5}$$

Chapter 5

Overall Framework

This chapter sketches the workflow for interactive knowledge discovery using the framework. It provides formal description of this process and discusses the design features of the framework. It also gives an overview of pattern formulation and query language and explains different components of User Interface.

5.1 Interactive Knowledge Discovery and Data Mining Process

In this section we provide a formal description of the interactive knowledge discovery process. As described in section 3, the inputs are multimodal data streams (DS) from heterogeneous data sources. In an abstraction process ($\mathcal{A}$), meaningful events (Ev) are extracted and recognized from time-series data. These events conforming to an event schema and create multiple event streams (ES) that feeds to the system. The goal or output of the process is insight or knowledge ($\mathcal{K}$) that ultimately constructs a model $\mathcal{M}$ from observational data. Knowledge is either obtained from visualizations of machine generated patterns (V_M) in

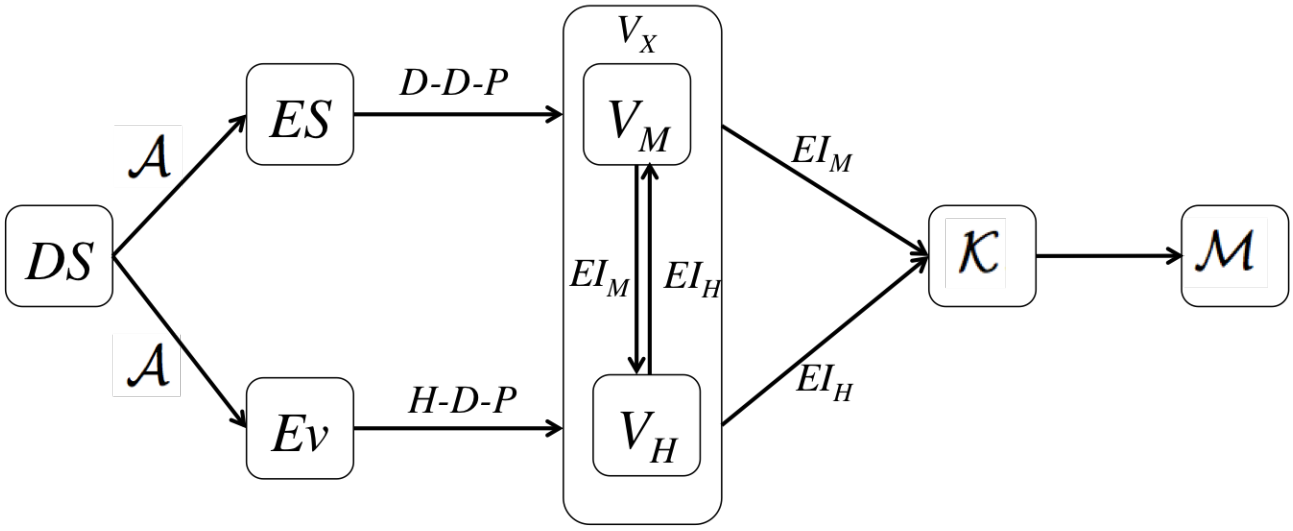


Figure 5.1: Interactive Knowledge Discovery/Data Mining Process

data-driven processing (D-D-P), or through formulation and visualization of a hypothesis (V_H) in hypotheses-driven processing (H-D-P). We illustrate this formalization of interactive KDDS process in Figure 5.1.

Formally, interactive knowledge discovery is a transformation $\mathcal{F} : DS \Rightarrow \mathcal{K}$, from data to knowledge, where $\mathcal{F}$ is a series of inter-connected functions $\mathcal{F} \in \{\mathcal{A}$, D-D-P, H-D-P, V_X, $EI_X\}$ defined as follows:

- $\mathcal{A}$ describes abstraction process where heterogeneous data streams are abstracted to events and event streams, $\mathcal{A} : DS \longrightarrow Ev$, $\mathcal{A} : DS \longrightarrow ES$, where $\mathcal{A} \in \{ML, St\}$ including statistical St, and Machine Learning ML techniques, that are needed to convert time-series data to symbolic event data.

- D-D-P indicates Data-Driven Processing where D-D-$P \in \{SEQ - CO, CON - CO\}$ which are data-driven operators that convert event streams to visualizations of machine-generated patterns D-D-P: $ES \rightarrow V_M$. These operators utilize temporal data mining techniques to bring hidden patterns to surface and facilitate significant pattern extraction process.

- *H-D-P* indicates Hypotheses-Driven Processing where $H\text{-}D\text{-}P \in \{;\ ,\ ;\omega_{\Delta t}\ ,\ \Subset\ ,\ |\}$ which are operators that formulate and evaluate a hypothesis and $H\text{-}D\text{-}P\text{:}\ Ev \rightarrow V_H$.

- V_X , $X \in \{M, H\}$ symbolize the visualization functions, which are either functions visualizing machine-generated patterns V_M or functions visualizing hypotheses V_H.

- EI_X , $X \in \{M, H\}$ indicates Expert-Interaction process that is integral part of knowledge discovery since domain knowledge is the anchor of model building process. Expert interactions can effect significant patterns to produce hypothesis $EI_M : V_M \rightarrow V_H$, can effect significant patterns to produce knowledge $EI_M : V_M \rightarrow \mathcal{K}$, or can effect hypotheses to produce knowledge $EI_H : V_H \rightarrow \mathcal{K}$.

5.2 Re-visiting Design Principles

Based on the discussions so far, we have defined the following design principles essential to interactive knowledge discovery framework.

5.2.1 Human-centered Analysis

The knowledge discovery framework keeps human in the loop while emphasizing on using advanced machine-based and data-driven model building techniques. In order to facilitate human interaction with the framework, data needs to be converted to a human-understandable form. For humans, event concept is a basic function of human brain to process the sensory information about real-world, produce knowledge about them, and react properly to the outside world. In the cyber space, event is a basic building block to capture knowledge and provide a natural abstraction of happenings in the real world as well. To bridge the gap between flow of information in real world and how information is captured, managed, and

processed in cyber world, we define an event model that captures multiple aspects of an event and creates an event interchange format,e.g. an event media JSON. So at the surface (the level of interaction between human and machine) knowledge in acquired by symbolic data analysis. The underlying processing, however, is performed on quantitative data streams.

5.2.2 Expressiveness of the Pattern Query Language

The analyst must be able to interact with the pattern discovery system having a high-level view of the data and the patterns. Pattern formulation and query capabilities are needed to filter and combine data. Therefore temporal primitives must be embedded in the language in form of expressive operators. The compositionality of the operators allows the user to create their own knowledge discovery process combining different operators. The expressive nature of patterns and operators enhance understandability of model building process and help analyst step into the process rather than relying on a blackbox and *one click prediction model building*. So, the analyst must be able to interact with the pattern discovery system having a high-level vision of the data and the patterns.

5.2.3 Interactive Modeling Approach

The iterative querying capability not only allows the analyst to apply the data-driven algorithms on the data to discover interesting patterns, but also formulate hypotheses and investigate these patterns further for a deeper analysis. This is an iterative process which allows the analyst to use the models not only as static knowledge to be presented as result, but as an active element of the process used to go deeper in the data understanding. Also with a high-level language that supports the steps of the knowledge discovery process, the execution process can be materialized. Thus, the output is not only the set of mined patterns, but also the script storing the process which makes the process repeatable on different

datasets. Discovered models and extracted patterns have to be explicitly represented and stored. This allows progressive mining and reusing of the extracted models with different data.

5.2.4 Extensibility

Extensibility is defined as the ability of a system to be extended with new functionality with minimal or no effects on its internal structure and data flow. The framework must provide an easy way to integrate new kind of data and algorithms. Extensions can be through the addition of new data-driven operators as well as hypothesis-driven operators. Extensions can also be made through new visualizations.

5.2.5 Result Interpretation

The interpretation of data and model usually is a very complex part of the knowledge discovery process. The main point is that the interpretation of the mined patterns depends on the application context. Indeed, there is a wide semantic gap between the mining algorithms and the final domain expert user who is assumed to use the extracted models. The objective of the framework is to fill this gap providing the final user with meaningful patterns.

5.3 General System Architecture

Building models involves many decisions such as determining model selection strategy, defining a model structure, defining criteria for model goodness, selection of data and transformation applied to it. Most of these decisions involve reliance on theoretical or empirical results, that is expert's domain knowledge, and cannot be learned by a system itself solely

from available input data. So it is often necessary to incorporate human judgment into this modeling process. As explained in section 5.1, in the formal explanation of interactive knowledge discovery, Expert-Interaction (EI) emphasis on "human-in-the-loop", while Hypotheses-Driven-Processing (H-D-P) enables hypotheses formulation with a set of operators. Also, when scientist doesn't have any idea what hypothesis to generate or how to proceed in data analysis task, automatic bottom-up data analysis techniques or Data-Driven-Processing (D-D-P) are useful in refining human judgment and provide significant insight. These two paradigms are shown in a high-level system architecture in Figure 5.2. Bottom-up data analysis results in visualization of co-occurrence patterns 5.2(a). By seeding a hypothesis based on insight derived from such analysis, expert can incorporate her own domain knowledge and formulate a refined hypothesis quickly, systematically, and 'grow' a hypothesis iteratively to generate a comprehensive model (Fig 5.2(b)). The former is called *unknown-unknown* problem and the latter is called *known-unknown* problem. The best model building practice is to create a balance between top-down and bottom-up analyses as one complements the other, reduces the search space significantly, and helps analyst harvest and understand descriptive pattern.

To illustrate this process consider the following scenario:

A worldwide problem in health systems is how to deal with increasingly large and complex sets of heterogeneous, high-dimensional data and increasing amounts of unstructured information. These information contain patients motion, location, attack history, allergens and pollutants in the environment, etc. As bits and bytes of data flow into the framework, they are transformed into meaningfully defined complex datum: event. Life events encode patient's activity of daily living. For example, exercise, jogging, walking, cycling, working, staying home, sleeping, etc. Environmental events encode states and state transitions in environmental variables such as temperature, air pressure, pollution, pollen, $PM2.5$, $PM10$, CO_2, CO, NO, ozone, etc.

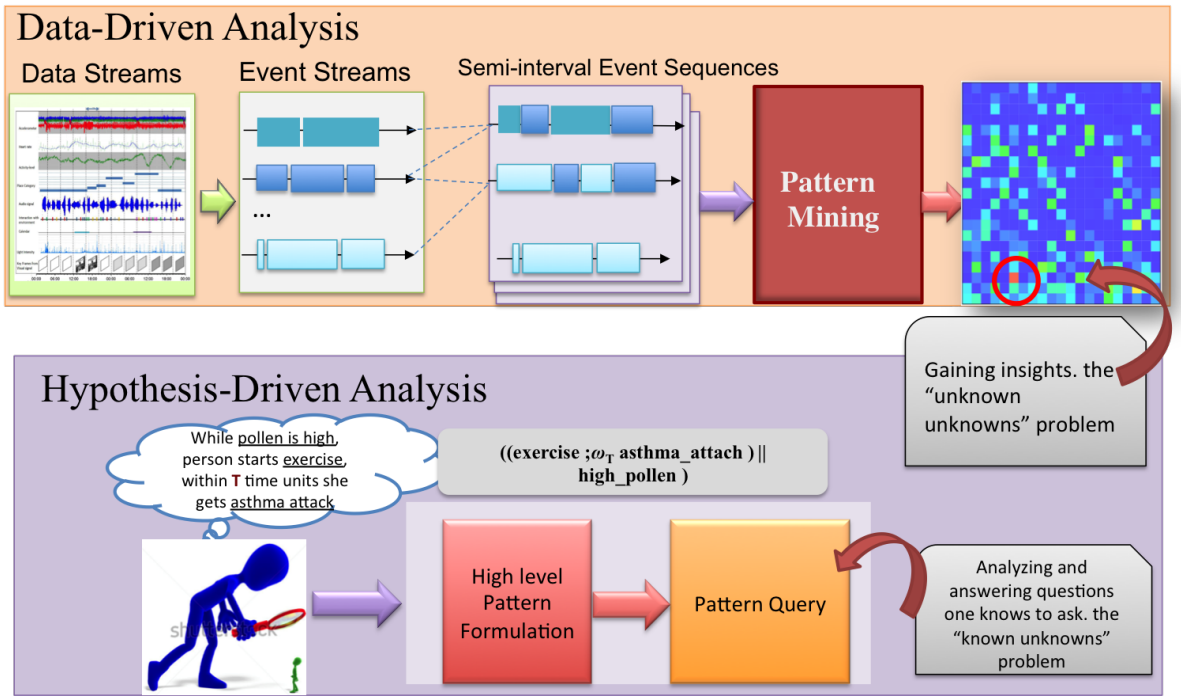


Figure 5.2: High level architecture of the framework.

An asthma specialist is using the framework to understand what effects a patients asthma attack, what conditions worsen patient's disease, and if patient's asthma is under control. At the beginning, he doesn't know what hypotheses to formulate or what questions to ask. So, analyst uses data-driven operators to understand all event co-occurrences patterns. Looking at the visualization, he realizes that the patterns $(Exercise; \omega_{[0-15]} Asthma - attack)$ and $\big((ActivityLevel\_high \mid ActivityLevel\_very\_high); \omega_{[0-30]} \ Asthma - attack\big)$ are significant patterns based on their occurrence frequency. The analyst then seeds a hypothesis based on this knowledge and investigate more expressive patterns using hypothesis-driven operators to formulate the following patterns:

$\big((Exercise; \omega_{[0-15]} Asthma - attack) \perp\!\!\!\perp Pollution\_high\big)$

$\big((Exercise; \omega_{[0-15]} Asthma - attack) \perp\!\!\!\perp Pollen\_high\big)$

$\big((Exercise; \omega_{[0-15]} Asthma - attack) \perp\!\!\!\perp (Pollen\_high \mid Pollution\_high)\big)$

$\big(((ActivityLevel\_high \mid ActivityLevel\_very\_high); \omega_{[0-30]} Asthma{-}attack) \perp\!\!\!\perp Temperature\_low\big)$

Interactive KDDS process helps asthma specialist to narrow down the causes of asthma attack and understand patient's sensitivity to different allergens in a systematic and scientific way.

5.4 Pattern Formulation and Query Language

From asthma management scenario in the previous section we derive several requirements for a high-level language for interactive knowledge discovery.

- The language must be expressive enough when it comes to the specification of complex patterns. Depending on the application domain, analysts will describe patterns of varying complexity.
- The pattern formulation and query language must be usable. From user's perspective, it must be easy to express complex patterns by using language primitives. So there should be balance between usability and being expressive.
- The language must be efficient in terms of user's performance goals, such as high performance, low space and time complexity. In particular, there must be an efficient implementation technique for pattern mining algorithms.

In this section we describe a pattern language that uses a special Finite State Automaton (FSA) extended with support for temporal relationships between events.

Definition 11 (*Automaton*): A finite-state automata (FSA) is a 5-tuple (OS, TS, Ed, s_0, s_f), consisting of a finite set of ordinary states (OS), time states (TS), transitions between states (Ed), a start state $s_0 \in OS$, and a final or acceptance state $s_f \in OS$.

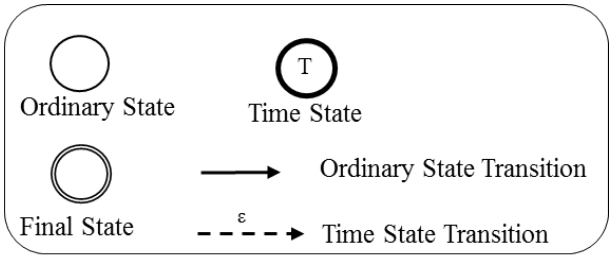


Figure 5.3: Basic Building Blocks of FSA in a high-level pattern formulation and query language

Figure 5.4 shows the building blocks of our FSA. Each operator in the language translates to its corresponding automaton to detect instances of a specific pattern in the input stream. The theory of finite-state automata is rich and finite-state automata techniques have been used in a wide range of domains, such as pattern matching, pattern recognition, speech processing, hand writing recognition, encryption algorithm, and data compression. Using FSA for pattern recognition and pattern matching has several advantages. First, FSA is a well-understood computational model with relatively simple implementation that supports symbolic pattern matching over data streams. Second, the high-level pattern language consists of a set of operators that are tailored towards formulation of complex patterns. The characteristics of these operators can be translated to specifications of different automata. Hence, complex patterns can easily be decomposed to multiple computational automata.

The processing operators defined in the chapter 3 are the basic building blocks of pattern formulation recognition. These operators need to be compiled to a processing unit. In our framework the pattern recognition component employs an extended FSA that supports a time model to address temporal restrictions that are needed in co-occurrence pattern analysis. This automaton contains a finite number of states and state transitions. There are two types of states: ordinary states (OS) and time states (TS). Ordinary state is analogous to states in traditional finite automata. It consume an event from event stream, apply an **EVALUATE()** function and make a transition to the next state if the evaluation is successful. Time state on the other hand keeps track of time constraint requirement by

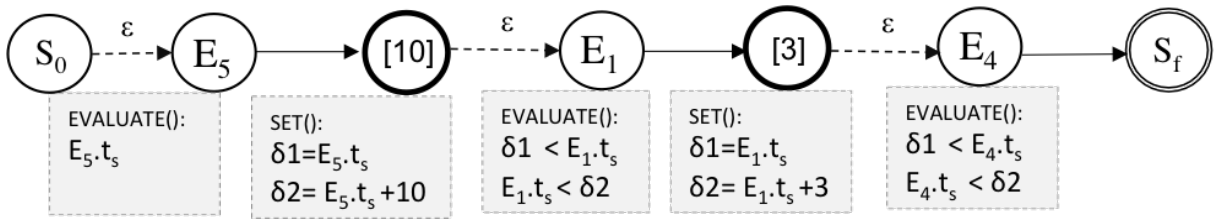


Figure 5.4: The automaton corresponding to pattern $\rho 1$ with 3 event components. It demonstrates 3 ordinary states, 2 time states, and EVALUATE() and SET() functions associated with each state.

applying a **SET()** function on boundaries of time lag (δ_1, δ_2) that is going to be used in the evaluation function of the next ordinary state. Ordinary state is represented by an event type $E_i \in \xi$ and it means that the FSA is waiting for E_i to be seen in the input event stream.

Figure 5.4 demonstrates an initialized FSA corresponding to pattern $\rho_1 = (E_5; \omega_{[10]} E_1; \omega_{[3]} E_4)$. During run time $E_i.t_s$ and $E_i.t_e$ substitute with start and end time of an instance of event E_i from the input event stream. The strategy for counting occurrences of a pattern is straight forward. For a pattern, say ρ, an automaton FSA_ρ is initialized. The initialization process includes translating event types and temporal constraints to ordinary states and time states, and allocating a buffer for EVALUATE() and SET() functions within each state. As we read data from event stream, by considering the output of EVALUATE(), automaton makes earliest possible transitions into the next successive state. Once it reaches its final state, an occurrence of the pattern is recognized and its frequency is increased by one. A fresh automaton is initiated for this pattern when an event corresponding to its first event appears again in the event stream.

5.4.1 Automata Model for Pattern Formulation

A high-level definition of a pattern with implicit structural and temporal information can be translated into automata-based pattern specification using automaton building blocks. We present the language operators and the construction of their corresponding automaton.

For simplicity and without the loss of generality, operators are depicted between two events only. However, these operators can be cascaded to compose complex patterns.

Selection ($E_i.P$)

This operator filters pattern expression on predicate P, where P refers to event attributes contained in the pattern, $P \in \xi$, and event schema ξ describes a set of attributes a class of events can contain. Also E_i, $1 \leq i \leq N$ is an event type where $E_i \subset \Sigma$, and Σ is the alphabet of event types.

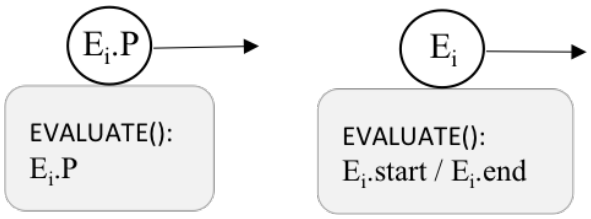


Figure 5.5: Selection operator evaluates an ordinary state on event type E_i and and its attributes P. It can also select an event type E_i without any specific attribute.

Sequence ($E_i; E_j$)

This operator detects if event type E_i is followed by event type E_j.

Conditional Sequence ($E_i; \omega_{[\delta_1,\delta_2]} E_j$)

This operator detects if event type E_i is followed by event type E_j within $[\delta_1, \delta_2]$ time units.

Concurrency ($E_i \between E_j$)

Concurrency detects if E_i and E_j are happening in parallel. Any order between events are acceptable.

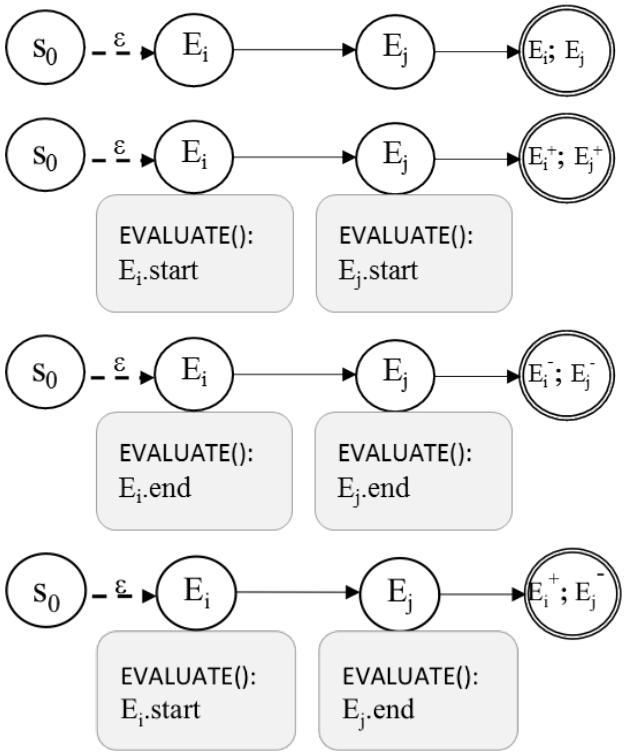


Figure 5.6: Sequence operator evaluates an ordinary state on first event type's start/end followed by second event type's start/end

Alternation ($E_i \quad | \quad E_j$)

Alternation detects if either event E_i or event E_j happens.

5.5 Graphical User Interface

Analyst can import pre-processed data (i.e. event streams) and choose visual bottom-up operators to explore data, generate a basic model and derive a preliminary insight. Then she can seed a hypothesis and grow it step by step using the top-down pattern formulation operators. A good hypothesis is not the one that is necessarily correct, but one that opens up a new path of investigation. In complex problem domains this path cannot be fully

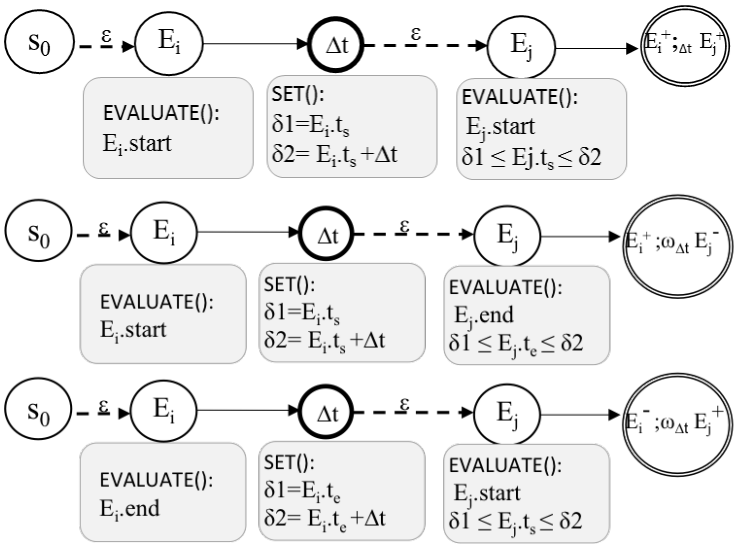


Figure 5.7: Conditional Sequence operator evaluates an ordinary state on first event type's start/end followed by second event type's start/end within $\Delta t = [\delta_1, \delta_2]$ time units. Time state set δ_1 and δ_2 values based on the first event's timestamp in the event stream. These values are used in the evaluation phase of the next ordinary state.

perceived in advance. So analyst must be provided with appropriate operators to carry out new analyses based on the original hypothesis.

Figure 5.10 demonstrates the UI of our framework. The basic components are:

1. **Data Selection Panel**: To select different event and event streams from database.

2. **Data-driven Operator Panel**: These operators are used for pattern mining from event streams.

3. **Hypotheses-driven Operator Panel**: These operators are used for pattern formulation and pattern query from input events.

4. **Visualization panels**: Results of pattern mining and pattern query are displayed visually in form of co-occurrence matrices and histograms.

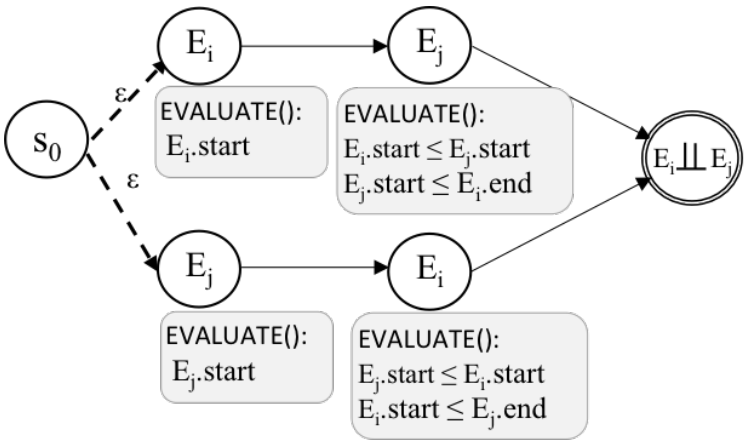


Figure 5.8: Concurrency operator evaluates an ordinary state on first event type's start/end followed by second event type's start/end and checks for a non empty temporal overlap in the second event's evaluation phase.

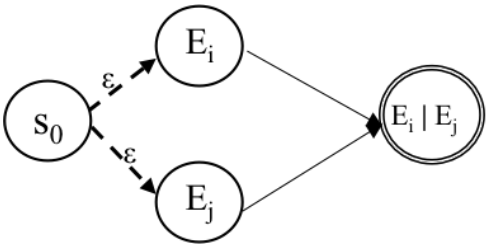


Figure 5.9: alternation operator evaluates multiple ordinary states on given event types.

Figure 5.10: Analytical dashboard of interactive KDDS framework.

Chapter 6

Significant Pattern Extraction

In temporal data mining the data is ordered (typically with respect to time) and the goal is to find patterns that characterize underlying temporal and structural dependencies. In this dissertation we discuss algorithms for finding certain class of complex patterns in symbolic event streams that are defined as an abstraction level on top of time series data. The data here is viewed as a collection of event streams. An event stream is an ordered sequence of events where events are conforming to an event model or schema that defines properties of those events. The data for co-occurrence analysis is viewed as a single sequence of events, denoted by $< (E_1, [t_1.s, t_1.e]), (E_2, t_2), (E_3^+, t_3), ..., (E_n, tn) >$, where n is the number of events in the event stream and E_i denotes an event type. Each event type can be of 3 categories: point event; interval event; and semi-interval event. Depending on the category the event belongs to, the representation differs. For example E_1 is an interval event and $[t_1.s, t_1.e]$ shows start and end times. E_2 is a point event and t_2 represents its timestamp. E_3 is a semi-interval event with a known start time t_3.

The co-occurrence patterns to be discovered in the event sequence are of two general types: sequential and concurrent. A pattern is a partially ordered collection of events occurring

together. These ordered collections of events may carry useful information regarding correlations among events types. In sequential co-occurrence patterns the order and time lag between events matters. In concurrent co-occurrence patterns, however, the temporal overlap between events matter only. A pattern is said to occur in an event stream if there are events of appropriate event types in the input sequence with a time ordering that conforms to the specifications of the pattern. The size of a pattern is equal to the number of participating event types in the pattern.

The computational problem is to find significant patterns up to a given size. A pattern is significant if its number of occurrences exceeds a pre-defined threshold. In any frequent pattern mining algorithm the significant output patterns extracted by the algorithm depends on the frequency threshold used. If a very low threshold is used then many frequent pattern but uninteresting patterns might shown in the output. If the threshold is too high, then significant patterns might not be captured by the algorithm. Hence an important consideration is how to find a frequency threshold so that frequent patterns represent statistically significant correlations. We propose a visual representation of co-occurrence patterns of size 2 in form of co-occurrence matrices in section 6.3 where an analyst can easily investigate structural and temporal relations between events and depending on the application specifications decides whether some displayed frequent patterns are actually interesting or not.

6.1 Co-occurrence Patterns

Fig. 6.1 shows a sample complex pattern. Multiple ordered events with time lags in between generate a sequential co-occurrence pattern. Also some events might happen in parallel and generate a concurrent co-occurrence pattern. Bringing an example from asthma management problem, a complex pattern of asthma attack can be: ([*e1*: patient did not take medication]

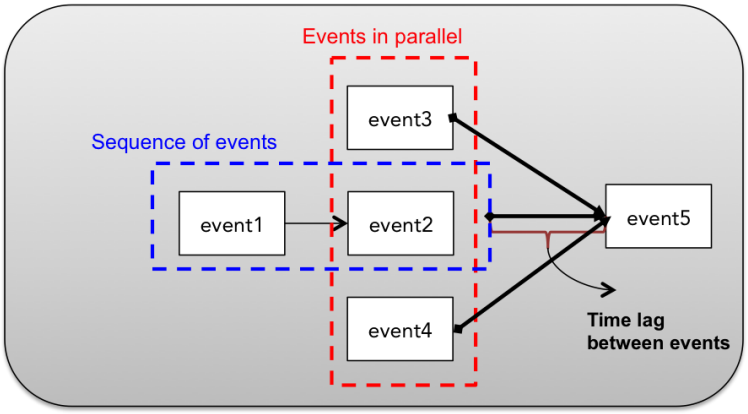


Figure 6.1: Sample structural and temporal relation between multiple events. Such relations can be encoded as sequential and concurrent patterns with specific time lag between events.

within 60 minutes *followed by* [*e2*: engaged in an intense exercise]) *while* ([*e3*: pollution is high] *or* [*e4*: temperature is below 20^o C]) *within* 15 minutes *followed by* [asthma attack].

We provide a high level pattern formulation and query language that contain a set of operators to formulate such complex patterns. Bellow is the asthma attack complex pattern translated into events and operators.

$$\Big(\big(\ e1\ ;\omega_{60}\ e2\ \big)\ \shortparallel\ \big(\ e3\mid e4\ \big)\Big);\omega_{15}\ e5$$

For illustration purposes, consider the following event streams:

$\mathcal{S}$: < $(E_1, 1, 5)$ $(E_2, 8, 11)$ $(E_5, 11, 18)$ $(E_3^-, 12)$ $(E_1^-, 30)$ $(E_5, 35, 40)$ $(E_6^+, 42)$ $(E_2, 53, 57)$ $(E_1^-, 60)$ $(E_4^-, 71)$ $(E1, 73, 76)$ >

The size of a pattern is equal to the number of participating events. The general term $(E_i^{+/-}; \omega_{\Delta t} E_j^{+/-})$ reads: in the occurrence of the pattern, when E_i begins/ends, then within Δt time units later E_j begins/ends. And the general term $(E_i; \shortparallel\ E_j)$ reads: in the occurrence of the pattern, E_i and E_j occur simultaneously. The definition of *simultaneous occurrence* or *concurrent occurrence* of two events depends on event's categories. One of the following situations might happen:

- Both E_i and E_j are point events. In this case concurrent occurrence means the two events have the same timestamp $E_i.t = E_j.t$

- Both E_i and E_j are interval events. In this case concurrent occurrence means there is a non-empty overlap between the two event's temporal dimension. Given $T_i = [E_i.t_s, E_i.t_e]$ and $T_j = [E_j.t_s, E_j.t_e]$ as temporal dimension of E_i and E_j respectively, concurrent occurrence is True if $T_i \bigcap T_j \neq \phi$

- E_i is point event and E_j is interval event (or vice versa). In this case concurrent occurrence is True if $E_j.t_s \leq E_i.t \leq E_j.t_e$.

- E_i is semi-interval event with known start time (or end time) and E_j is interval event. In this case concurrent occurrence is True if $E_j.t_s \leq E_i.t \leq E_j.t_e$.

- E_i is point event and E_j is semi-interval event (or vice versa). In this case concurrent occurrence is vague and there is not a definitive answer whether two events can happen concurrently or not.

Gong back to the above event stream example, the pattern $\rho 1 = (E_2; \omega_{20} E_1^-)$, reads as when E_2 happens within 20 time unites E_1 ends, and it has two instances in $\mathcal{S}$:
$< (E_2, 8, 11)(E_1^-, 30) >$
$< (E_2, 53, 57)(E_1, 73, 76) >$.

Also the pattern $\rho 2 = (E_3 \perp\!\!\!\perp E_5)$ has one instance in $\mathcal{S}$:
$< (E_5, 11, 18)(E_3^-, 12) >$

6.2 Processing Algorithms

In this section, we present new algorithms for frequent pattern discovery based on the frequency count of non-overlapped occurrences of set of candidate patterns. The space com-

plexity is in the order of number of candidate patterns in the input and the time complexity is linear in the number of candidate patterns and the total number of events in the event stream. The algorithms need to process multiple patterns with only one pass through the data. For instance, with N event types, N^2 patterns need to be analyzed to compute auto co-occurrence value between each pair of events and generate a co-occurrence matrix. Considering different time lags, the total number of possible patterns increases substantially. For such batch analysis, our processing algorithm counts the frequency of a collection of candidate patterns efficiently with only one pass through data.

As explained in the previous chapter, patterns are formulated using a set of operators provided by the framework. The input to an operator is one or multiple event streams and the output is an event stream containing the instances of a specific pattern that was built by applying the operator. Operators can be cascaded to formulate more complex patterns. In case of hypothesis-driven operators, one pattern is formulated and feed as the input to the processing component when the number of pattern occurrence and pattern instances are computed.

In case of data-driven operators, a set of candidate patterns are formulated (in form of sequential or concurrent patterns depending on the operator characteristics), and this set of patterns feed as the input to the processing component. The output of the processing is the number of occurrences of each candidate pattern and an event stream containing all instances of those pattern.

6.2.1 Sequential Pattern Mining

Algorithm 1 shows the pseudocode of sequential pattern processing algorithm. One automaton is initialized for each candidate pattern. To efficiently access all automata, we index them using a *waits(.)* list. For each event type E, the automata that are waiting for E are

linked together in a list pointed by *waits(E)*. We allow the automata to make transitions as soon as a relevant event type appears in the event stream. Once an automaton reaches its final state, frequency of the corresponding pattern increases by one and the automaton is re-initialized again to track another occurrence of the pattern.

6.2.2 Conditional Sequential Pattern Mining

Algorithm 2 shows the pseudocode of sequential co-occurrence processing algorithm. One automaton is initialized for each candidate pattern. To efficiently access all automata, we index them using a *waits(.)* list. For each event type E, the automata that are waiting for E are linked together in a list pointed by *waits(E)*. The idea of efficiently indexing automata through a *waits(.)* list was introduced in the windows-based frequency counting algorithm [89]. We initialize one automaton (initially in its start state) for every candidate pattern and then go down the event stream. We allow the automata to make transitions as soon as a relevant event type appears in the event stream to make the transition. Once an automaton reaches its final state, frequency of the corresponding pattern increases by one and the automaton is re-initialized again to track another occurrence of the pattern. At any state, if the validation function fails, automaton will not proceed with the current match and goes back to its start state.

At any stage in the algorithm, there is only one active automaton per pattern which means that there are $|P|$ automata being tracked simultaneously, where $|P|$ is the total number of candidate patterns. Thus the space complexity of this algorithm is $O(|P|)$. The initialization time is $O(|P| + |\Sigma|)$, where $|\Sigma|$ denotes the total number of event types. The time required for the actual data pass is linear in the length, n of the input event stream. Thus, to count frequencies for all candidate patterns the time complexity of this algorithm is $O(n|P|)$.

Algorithm 1 Counting Sequential Patterns

Input: Event streams $\{ES^1\ ES^2\ ...\ ES^K\}$, Collection $\mathcal{P}$ of candidate patterns
Output: Frequency count of patterns in $\mathcal{P}$
Begin
1:$\mathcal{S}$ = $merge(ES^{(1)}, ES^{(2)}, ..., ES^{(K)})$
2: For each event type $E \in \mathcal{S}$, initialize $waits(E) = \phi$
3: For each pattern $\rho \in \mathcal{P}$ do
4: initialize an automaton $FSA_\rho = (OS_\rho, \mathcal{E}_\rho, s_0, s_f)$
5: OS_ρ = Array of event types in ρ
6: $\rho.freq = 0$
7: $\rho.length = OS_\rho.size$
8: Add FSA_ρ to $waits(OS_\rho[1])$
9: Event stream $ES = \phi$
10: For $i = 1$ to n read e_i from $\mathcal{S}$
11: For each FSA_ρ pointed by $waits(e_i)$
12: x=*current_state*
13: if ($OS_\rho[x]$.EVALUATE()==TRUE)
14: x++
15: proceed to $OS_\rho[x]$
16: if $OS_\rho[x] == s_f$
17: $\rho.freq++$
18: Add ρ instance to ES
19: re-initialize FSA_ρ.
20: else
21: add FSA_ρ to $waits(OS_\rho[x])$
22: endif
23: else
24: re-initialize FSA_ρ.
25: endif
26: endfor
27: endfor
28: Return $\mathcal{P}$ and ES
End

Algorithm 2 Counting Conditional Sequential Patterns

Input: Event streams $\{ES^1\ ES^2\ ...\ ES^K\}$, Collection $\mathcal{P}$ of candidate patterns
Output: Frequency count of patterns in $\mathcal{P}$
Begin
1:$\mathcal{S}$ = $merge(ES^{(1)}, ES^{(2)}, ..., ES^{(K)})$
2: For each event type $E \in \mathcal{S}$, initialize $waits(E) = \phi$
3: For each pattern $\rho \in \mathcal{P}$ do
4: initialize an automaton $FSA_\rho = (OS_\rho, TS_\rho \mathcal{E}_\rho, s_0, s_f)$
5: OS_ρ = Array of event types in ρ
5: TS_ρ = Array of time lags in ρ
6: $\rho.freq = 0$
7: $\rho.length = OS_\rho.size$
7: $\rho.current\_state = 0$
8: Add FSA_ρ to $waits(OS_\rho[1])$
9: Event stream $ES = \phi$
10: For $i = 1$ to n read e_i from $\mathcal{S}$
11: For each FSA_ρ pointed by $waits(e_i)$
12: $x = \rho.current\_state$
13: if ($OS_\rho[x]$.EVALUATE()==TRUE)
14: proceed to $TS_\rho[x]$
15: execute $TS_\rho[x]$.SET(), initialize δ_1 and δ_2
16: x++
17: proceed to $OS_\rho[x]$
18: if $OS_\rho[x] == s_f$
19: $\rho.freq++$
20: Add ρ instance to ES
21: re-initialize FSA_ρ.
22: else
23: add FSA_ρ to $waits(OS_\rho[x])$
24: endif
25: else
26: re-initialize FSA_ρ.
27: endif
28: endfor
29: endfor
30: Return $\mathcal{P}$ and ES
End

6.2.3 Concurrent Pattern Mining

For processing concurrent patterns we take the set of pattern candidates and the event stream as inputs, and produce the occurrence frequency of the patterns as output. The main data structure here is once again a *waits(.)* list. The initialization process involves adding ρ to the *waits(.)* lists of every event type in pattern ρ. For example, for patterns $\rho_1 = E_1 \between E_2$ and $\rho_2 = E_3 \between E_2$, will initially add pointers to ρ_1 automaton form $waits(E_1)$ and $waits(E_2)$, and pointers to ρ_2 automaton form $waits(E_2)$ and $waits(E_3)$. As shown in Figure 5.8, concurrent pattern's FSA does not include time state. Each *waits(.)* list can have at most $|P|$ entries. So the space need for this algorithm is $O(N|P|)$, where N is the number of distinct event types in a pattern. the initialization time complexity is $O(|\Sigma| + N|P|)$, where $|\Sigma|$ denotes the total number of event types in the system. The main loop is over n events in the input event stream, and any of the *waits(.)* loops, can at most be over $|P|$ partial occurrences. Re-initialization of appropriate *waits(.)* lists whenever an occurrence is complete takes constant time. This re-initialization needs to be done at most n times for each pattern. Hence, the total worst case time complexity of this algorithm is $O(n|P|)$.

6.3 Visual Analytics Process

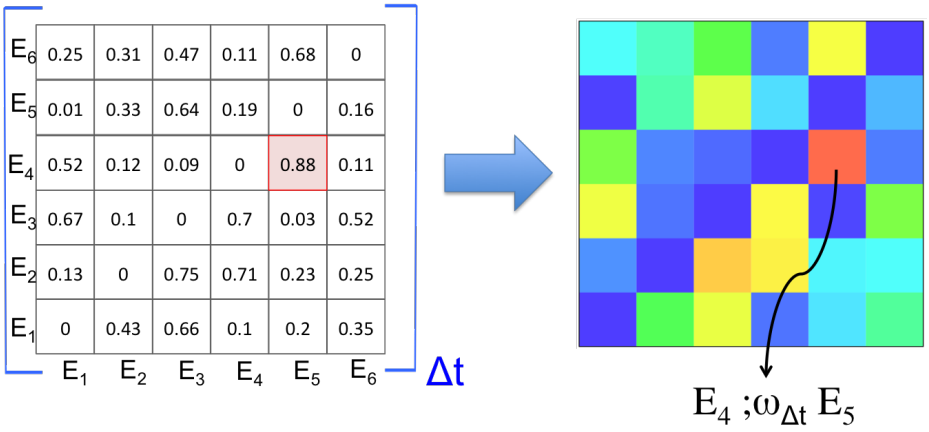
In most pattern mining techniques, occurrence frequency is the main criteria to asses significance of a pattern. However, using high frequency thresholds may reveal only the common knowledge while low thresholds results in an explosion of discovered patterns. To solve this problem, for size-2 patterns we display the occurrence frequency of a candidate pattern as its **confidence value** and we form a co-occurrence matrix to display confidence value of multiple patterns. This helps an expert to visually extract interesting patterns. Sequential Co-occurrence generates a matrix for any given Δt. The confidence value of a size-2

Algorithm 3 Counting Concurrent Patterns

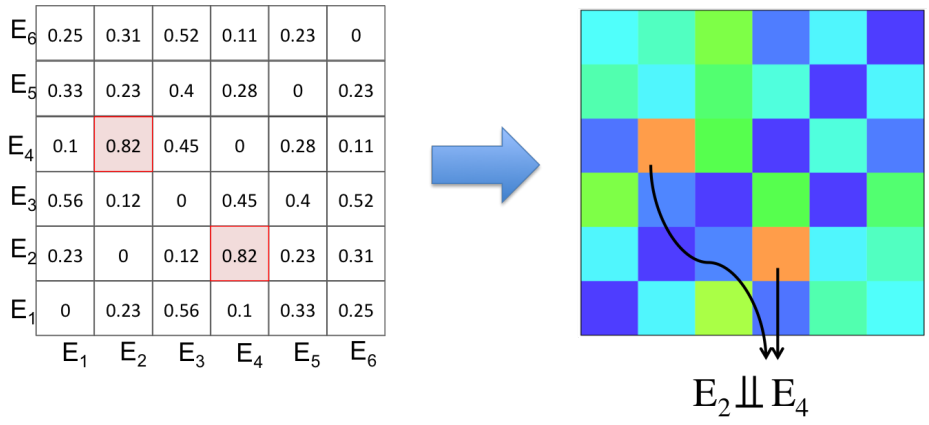
Input: Event streams $\{ES^1\ ES^2\ ...\ ES^K\}$, Collection $\mathcal{P}$ of candidate patterns
Output: Frequency count of patterns in $\mathcal{P}$
Begin

1:$\mathcal{S} = merge(ES^{(1)}, ES^{(2)}, ..., ES^{(K)})$
2: For each event type $E \in S$, initialize $waits(E) = \phi$
3: For each pattern $\rho \in P$, initialize a FSA_ρ, add FSA_ρ to $waits(\rho$.all-event-types)
4: For $i = 1$ to n read e_i from S

5: For each FSA_ρ pointed by $waits(e_i)$
6: if(FSA_ρ.EVALUATE()==TRUE)
7: proceed to next ordinary state.
8: if FSA_ρ is at final state
9: increase $FSA_\rho.frequency$
10: re-initialize FSA_ρ.
11: else
12: E'= event type of ordinary state.
13: add FSA_ρ to $waits(E')$
14: endif
15: else
16: re-initialize FSA_ρ.
17: endif
18: endfor
19: endfor
20: Return $\mathcal{P}$ and ES
End



(a)



(b)

Figure 6.2: Sequential and concurrent Co-occurrence matrices. Each cell shows normalized occurrence frequency of patterns (a) $E_i; \omega_{\Delta t} E_j$ and (b) $E_i \between E_j$

conditional sequential pattern is defined as:

$$Confidence(E_i \xrightarrow{\Delta t} E_j) = \frac{Count(E_i; \omega \Delta t E_j)}{Count(E_i)} \tag{6.1}$$

This value means, from all the time that the first event (E_i) happens, how many times it is followed by the second event (E_j) within Δt.

The confidence value of a size-2 concurrent pattern is defined as:

$$Confidence(E_i \parallel E_j) = \frac{Count(E_i \between E_j)}{\frac{1}{2}\big(Count(E_i) + Count(E_j)\big)} \tag{6.2}$$

As shown in Figure 6.2, x and y axes of the matrix are composed of event types in a specific application domain. Each cell shows the the confidence value of a specific size-2 pattern. Concurrent co-occurrence matrix is symmetrical because the order between events does not matter and practically pattern $E_i \between E_j$ and $E_i \between E_j$ are the same. We create one sequential co-occurrence matrix for each time lag Δt. In the UI, the expert can use a sliding bar to change the value of time lag and visually analyze multiple co-occurrence matrices easily.

6.4 Simulation Results

This section presents the results obtained from synthetic data that generated by embedding specific interval and semi-interval patterns in varying levels of noise. The main objective of the experiments is to empirically demonstrate the ability of the above pattern mining algorithms in detecting co-occurrence patterns with different sizes. By varying the control parameters of synthetic data generation, it is possible to generate qualitatively different kinds of datasets. Each semi-interval pattern to be embedded in the synthetically generated data, consists of a specific ordered sequence of events and time constraints between them.

Data generation process is as follow: There is a timer that specifies the current time instant. Each time an event is generated, this timer specifies event's start time. The duration of each event is picked from a normal distribution with a mean value μ. Event's end time is the sum of its start time and duration. Number of event types defined by $|\gamma|$ and n is the number of events in the stream. Minimum temporal granularity is set to one minute. However, this value is application dependent and can be as small as millisecond. After generating an event, timer is incremented with a small random integer. Each time the next event is to be generated, two decisions should be made:

1. Whether the event is going to have both start and end timestamps, or one of them might be missing randomly. This is controlled by the parameter α, which is the probability that the next event has its both interval boundaries. If α = 1 then event stream contains only complete intervals.

2. Whether the next event is to be generated randomly with uniform distribution over all event types or according to one of temporal patterns to be embedded. This is controlled by the parameter β , which is the probability that the next event is generated randomly. If β = 1 then data contains only noise with no temporal patterns embedded.

If it decides that the next event is to be from one of the temporal patterns to be embedded, then we have a choice of continuing with a pattern that is already embedded partially or starting a new pattern. If time constraints of the partial pattern cannot be satisfied any more, we start a new occurrence of the pattern. Five datasets with varying amount of noise are generated. We embed the following pattern of size six in all datasets:
$\rho = E_A^+ \ ;\omega_{15} \ E_B \ \omega_{10} \ E_c^- \ ;\omega_{20} \ E_G \ ; \ \omega_{60} \ E_H \ ; \ \omega_{90} \ E_D$

For easier understanding, we can write a sequential pattern as:
$\rho = E_A^+ \xrightarrow{15} E_B \xrightarrow{10} E_c^- \xrightarrow{20} E_G \xrightarrow{60} E_H \xrightarrow{90} E_D$

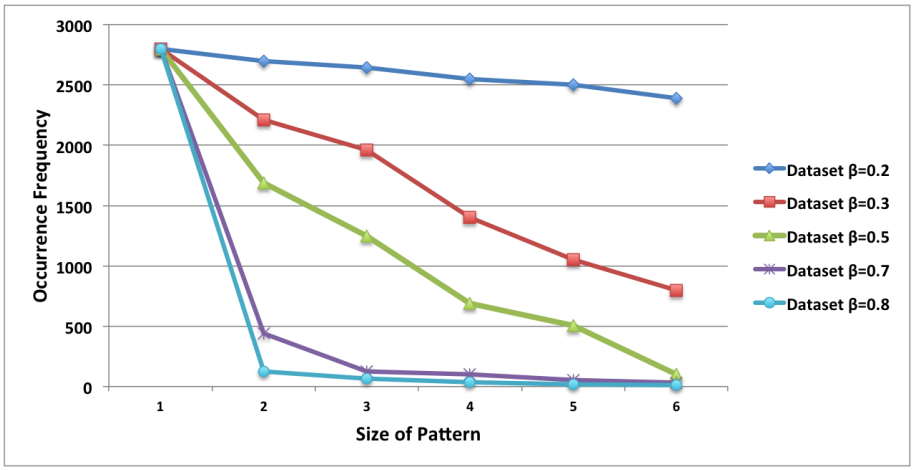


Figure 6.3: Frequencies of pattern ρ with sizes from 1 to 6 in five dataset with $n = 10^6$, $|\gamma| = 22$, $\alpha = 0.3$, and varying amount of noise β. Dataset with $\beta = 0.2$ has the least noise.

Data generation with $\beta = 0.2$ means that with 20% probability the next event is generated randomly and with 80% probability either pattern ρ is continued or a new occurrence of ρ is started. Frequency count of pattern ρ from size one (E_A^+), size two $(E_A^+; \omega_{15} E_B)$, to size six (complete pattern) is plotted in Figure 6.3. Our objective is to see whether this pattern can be detected based on its frequency counts given different amount of noise in the data.

It is apparent that when the noise increases, frequencies of patterns (partial patterns and complete pattern) falls quickly. Looking at the curves corresponding to five datasets we see that decrease of pattern's frequency with increasing size is directly related to how much noise is injected in the dataset. Thus we can say that long patterns with high frequencies cannot come out unless there are strong co-occurrence connection between corresponding event components of the pattern in the underlining data generation model.

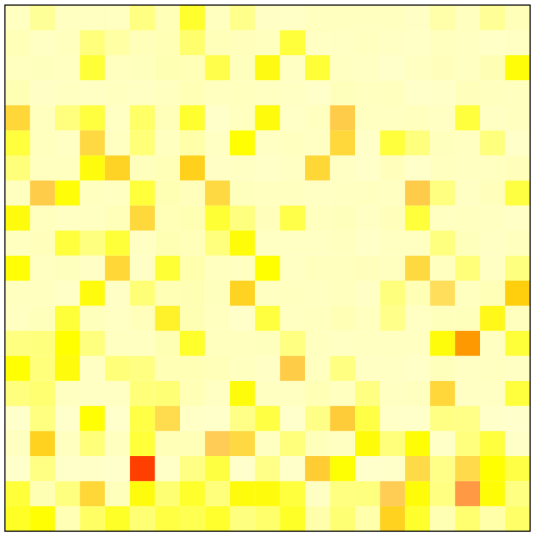
Co-occurrence matrix can only visualize patterns of size two. By changing temporal offset, multiple co-occurrence matrices can be computed. In this experiment we generate a dataset with $n = 10^5$, $|\gamma| = 22$, $\alpha = 0.8$, and three patterns are embedded:

$\rho 1 = E_C; \omega_{15} E_F$

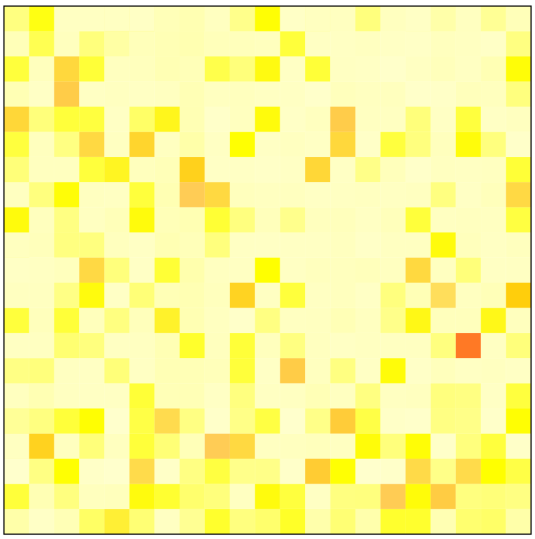
$\rho 2 = E_I; \omega_{30} E_M$

$\rho 3 = E_S; \omega_{60} E_H$

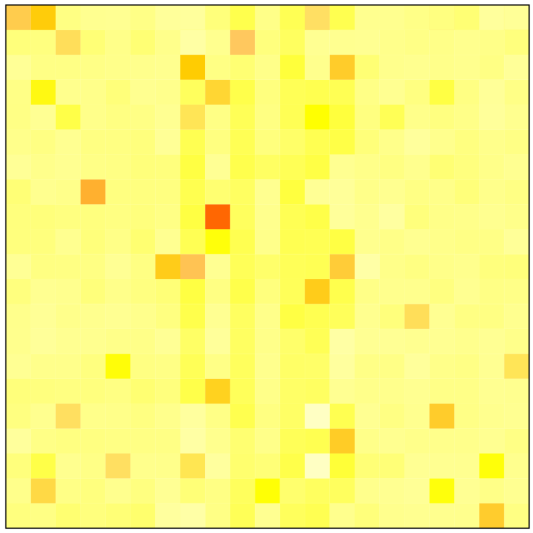
Three co-occurrence matrices are demonstrated in Figure 6.4. Such a visualization facilitates browsing co-occurrence characteristics in event streams, formulating hypothesis regarding those characteristics, and investigating structural and temporal relationships between events.



(a) $\rho 1 = (E_C; \omega_{15} E_F)$ with co-occurrence value=0.92



(b) $\rho 2 = (E_S; \omega_{30} E_H)$ with co-occurrence value=0.82



(c) $\rho 3 = (E_I; \omega_{30} E_M)$ with co-occurrence value=0.84

Figure 6.4: Co-occurrence matrices with different temporal offsets. (a) Δt=15 min, (b) Δt=30 min ,(c) Δt=60 min.

Chapter 7

Objective Self

Self is a fascinating subject. It is what we know intuitively but it is hard to define. When we ask psychologist what self is, they might say it is indefinable; should not be defined; will be limited if we define it; or it is a place holder for something we do not understand. However, we should not stay in dark.

When we define self, it changes the definition of a healthy self and in a practical way, the approaches you can take to strengthen the self. When we try to enhance self, the primary step is to measure it. So we have to measure or monitor the things that we want to influence or control. Monitoring and modifying are two essential components of our way to a healthy self. Once we define self in this way, we can have a better idea how to create a stronger self, or the least how to model self. There are many ways to describe feelings, activities, emotions, etc. However, the ability to observe those feelings and activities and understand the relation between those will take our understanding of self to whole another level.

In this chapter we discuss how to objectively measure self from diverse sources of information, how to build objective self as a representation of an individual's model, and how to use this model in precision health applications.

7.1 Introduction

Humans have always been interested in understanding themselves and their environment. Understanding their relationship with the environment is important to survival as well as thriving in the present situation and planning for the future. This desire has resulted in scientific approaches to understanding the environment and the self. In fact, technology relies on models developed by basic sciences to improve the environment. The popularity of scientific approaches in the last few centuries is due to their success in building models that could be used for prediction in evolving environments. Prediction is essential for shaping and controlling the future but is not possible without some kind of model.

Scientific methods have evolved over centuries. A set of basic assumptions are used to justify the scientific method: First, there is an objective reality shared by all rational observers. Second, this objective reality is governed by natural laws. Third, these laws can be discovered by means of systematic observation and experimentation. Clearly, objectivity is important in science and is accomplished via a data collection process through experimentation. Most scientific methods emphasize the design of experiments to collect data under controlled conditions to discover natural laws. The disciplines where this is possible have significantly increased over time. In other disciplines, technology is needed to record observations. The quest for scientific approaches and development of technology has resulted in the development of novel sensors and approaches to data collection. One such area that many of us are familiar with is medical imaging and other diagnostic techniques. These techniques let us observe an objective reality that goes beyond human experience that relies solely on our natural sensors (senses).

Many scientific fields have evolved and are now well advanced to understand our environment. However, an area that is close to all of us, but remains subjective, is self. We dont have scientific approaches to understand the self. This is true at almost every level, our social

interactions, our body, and our emotional reactions. Surprisingly, although it is usually the most important object of interest to us, the self is possibly the least understood. Historically, we have gone through the following phases of understanding the self: anecdotal, subjective diary, and quantified self. Lately, there has been a lot of talk about the quantified self [141]. We have just we have entered the most scientific stage - yet we are in early infancy: the objective self.

7.2 Toward Objective Self

The interest in the self has been a complex and sustained quest. The issues that complicated this quest and continue to do so remain:

- getting objective data
- storing data
- analyzing data
- protecting privacy

With advances in technology, some of this is changing. Based on the technology trajectory, most of these challenges are now within reach and may make the next stage achievable within the next few years.

7.2.1 Anecdotal Self

At one point, no technology existed for collecting and storing data. People relied on their natural sensors (senses) for data collection, their memories for storage, and their mental

analysis and introspection for processing. Because no techniques were available to collect objective data and memory is a complex system that is at best subjective and volatile, much of the information about the self was anecdotal. This information changed with time because even the original data changed based on later events and their outcome. Although people always used their experiences, these experiences were truly ephemeral.

7.2.2 Diarizing Self

When writing technology became popular and commonly available, the idea of diaries and chronicles started taking shape. People realized that depending on memory to record data was not reliable. They started recording their observations to make them both time invariant and retrievable by themselves and others. Individuals started recording important things about themselves using diaries and started recording their observations in the form of chronicles. Initially, these records were only available in handwriting, drawings, or sketches. As technology advanced, photography and later magnetic recording allowed us to record more objective and experiential data. New sensors have also been developed for observing and recording data.

These techniques yielded better data by transforming simple memory recordings into subjective observations. The techniques still remained subjective because human observers and senses play the primary role in such observations, however, and a persons ability to articulate in recordings was the primary means of saving the data. Also, the data analysis was still performed by humans and hence also subjective.

Some recording techniques have been popular for a long time and have been evolving slowly with advances in technology. Some have decreased in popularity, such as diary writing, while others have increased significantly, such as photos, emails, status updates, and medical information records. Such techniques are increasingly becoming more objective as much of

the recorded data is becoming experiential. On the other hand, the data is sparse. It is usually collected during certain episodes, which make it biased. Also, it lives in silos for many reasons, including privacy issues. All this makes the determination of models of a person difficult, and any model derived from such data is low quality because the data is so poor. An important phrase in computer science can be paraphrased to describe this situation: garbage data, garbage models.

7.2.3 Quantified Self

The 21st century has witnessed significant advances in storage, processing, sensing, and communication technologies. All these have resulted in the popularization of strong data-dependent approaches, leading to the rise in the popularity of *scientism* [113] in almost all disciplines where data can be collected. Because scientific approaches emphasize observation and systematic experimentation, the availability of sensors to observe different aspects of physical reality has encouraged the collection of such data as well as its analysis to develop laws of nature. As the availability of data has become widespread, the desire to understand the physical reality at different levels in different applications has also become possible and desirable. Big data and analytics are now two of the most commonly used terms in scientific circles and in many other fields as well.

Inspired by this growth in data and its applicability in disparate areas, many people have started collecting data about themselves. This popular and rapidly spreading movement is called quantified self. Now that different types of sensors are available and new types of sensors could be developed, people who are sensitive to their health have started recording health-related information using sensors ranging from simple wearable accelerometers that could be classified and recorded in simple activities (such as walking, jogging, and climbing

stairs) to other measurements (such as body temperature, heart rate, perspiration rate, galvanic skin response, and many other deeper parameters).

With the trend of activity recognition, Alan F. Smeaton et al. emphasize the importance of lifelogging as *a phenomenon whereby people can digitally record their own daily lives in varying amounts of detail, for a variety of purposes* [55]. They consider the application of lifelogging in different domains including medical (i.e. memory support), behavioral science (i.e. analysis of quality of life), work-related (i.e. auto-recording of tasks), etc. [116]. They suggest an end to end lifelogging solution with cutting edge components (i.e. gathering, enriching, segmenting, keyframe, annotation and narrative) for extracting meaningful knowledge from one's lifelog data. Lifelogs or eChronicles could be considered the predecessors of the quantified self. With these systems, people record their life activities using wearable and other sensors. Their activities are detected using classification techniques and then visualized or analyzed to extract meaningful information from the data. Much research was done in the context of individuals and for the US Department of Defense. A popular project in this area called MyLifeBits [50] was championed by Gordon Bell at Microsoft.

Wearable computing is now readily available, which will further encourage people to record their physical activities, other life activities, bodily parameters, and social interactions. It is commonly believed that recording such data and having access to it may result in many personal benefits for people. Figure 7.1 shows the evolution of models over time with the type of data that was available. We believe that with the availability of better quality data the quality of the models will facilitate real-time guidance and control, as has become commonplace for mechanical systems.

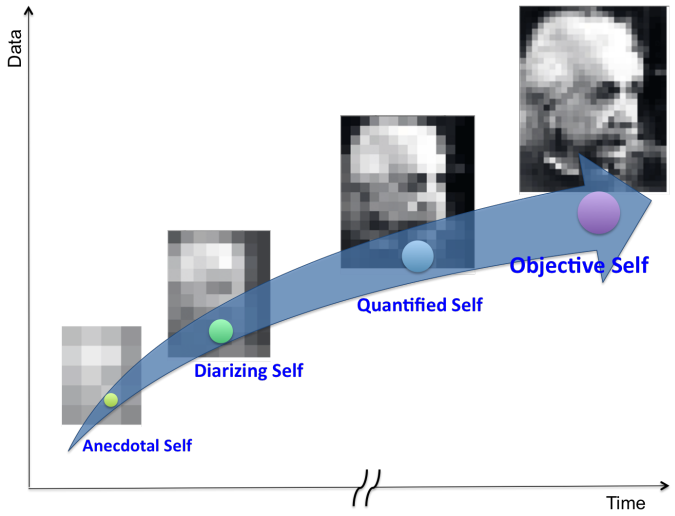


Figure 7.1: Evolution of self models. More data have allowed the progression from poor-quality models to high-quality models.

7.2.4 Objective Self Has Arrived

The quantified self movement is an important step in introducing a scientific framework to help understand an individual based on continuously collected data. This is the first time in the history of humanity that this has become feasible. The early stages of scientific framework based on sustained observations of controlled and natural experiments are being converted to laws related to the physical, social, and spiritual systems representing an individual. This opportunity is revolutionary on many fronts. We define Objective Self as follow:

The process of objectively measure physical, physiological, and mental activities of a human being and understand the associations between these activities.

Sensors are now available that people may use as they go about their regular activities. Also Smartphones can collect considerable context data about the user, ranging from physical activity and apps used to places visited. Considerable research has been devoted to using raw location or physical sensor data to infer and log higher level user activities such as driving and cycling. However, combining these individual multimodal streams to recognize higher level life events (i.e. sleeping, working, or commuting etc.) has remained a challenge.

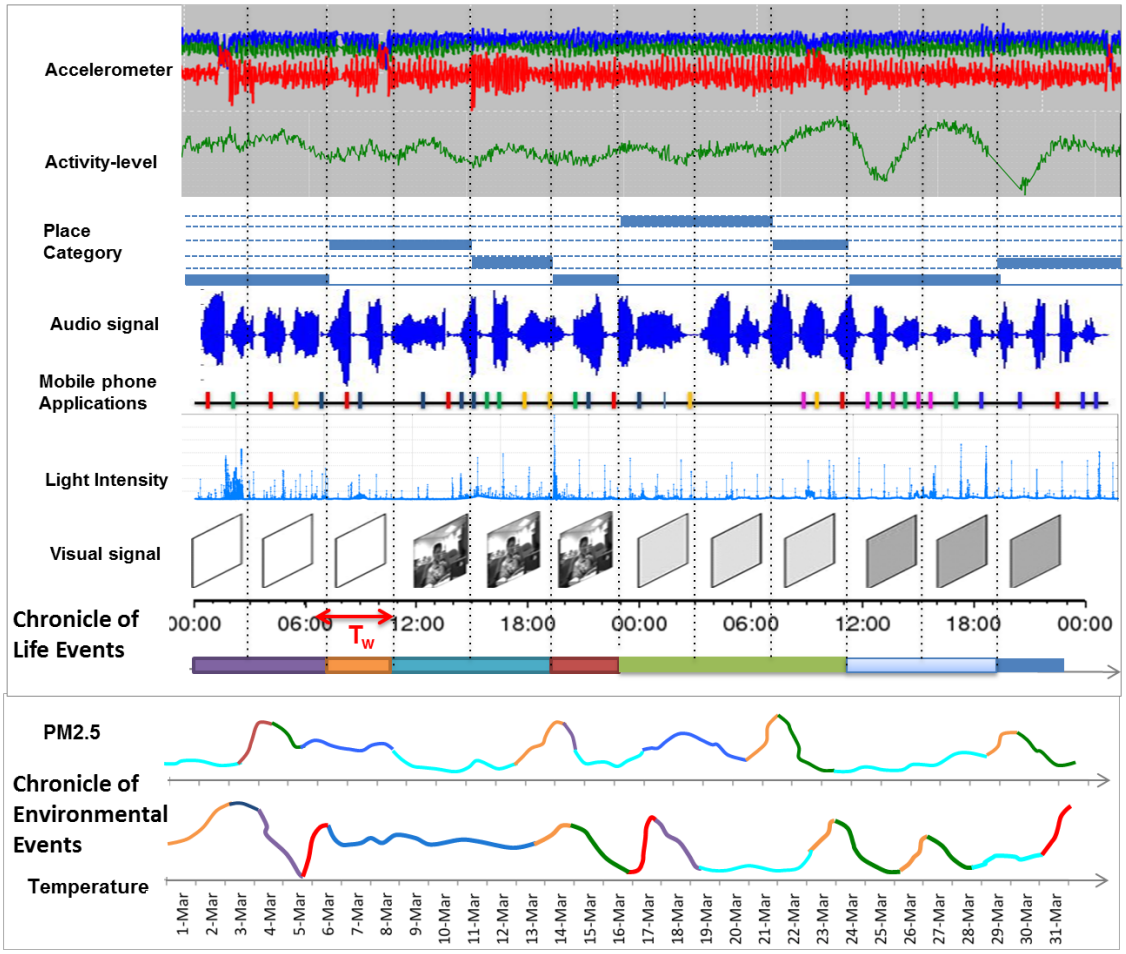


Figure 7.2: Chronicle of life events are derived from heterogeneous mobile multimedia content. Chronicle of environmental events are shown as segmented time series data. Each colored segment corresponds to a specific event.

Figure 7.2 shows two broad categories of data streams. The first category includes sensors that collect heterogeneous personal data such as accelerometer and GPS. These data streams are temporally aligned and divided into equal time windows T_w. Within each time window one or multiple probable life events and relevant attributes of these life events might be recognized. Life events signifies all daily living activities which are part of daily life. Thus, we can effectively obtain a chronicle of the persons life events called **personicle**.

The second category contains environmental sensor information such as pollution and temperature. These time-series data can be converted to time series of discrete labels through discretization, and meaningful events can be defined on top of these data streams (e.g. a

pollution increases suddenly). As data is collected through time, recognized events create a chronicle. Such event chronicle introduce a new structure over time dimension and data analysis and model building can be performed at this abstracted level rather than on low-level time series data. Next we explain an architecture that uses such event chronicle to construct objective self.

7.3 An Architecture for Objective Self

As the number and ubiquity of sensors and mobile devices continue to increase, the need for computational methods to analyze the avalanche of heterogeneous sensor data and derive objective self will grow. In fact, novel information processing architectures and platforms should be developed to enable the handling of multimodal data streams from heterogeneous sources. We have developed an architecture for analyzing objective data from sensors to both recognize life events and identify patterns and interrelationships among them to create the objective self.

Figure 7.3 shows a high-level architecture of an objective self platform. This architecture has three main components: data ingestion, life event recognition, and pattern recognition. Data ingestion uses various sensors and pre-processing modules to extract appropriate attributes from raw sensor measurements. Life event recognition assigns the most appropriate life event, from a set of predefined event classes, based on data stream attributes. Depending on the nature and number of life events, a designer must select sensors that will enable this classification. Thus, life events detected are independent atomic elements of analysis for the objective self. Essentially, the objective self is the model of a person. A co-occurrence based machine learning environment is needed to detect recurring correlated and causal patterns. This may result in insights such as a sensitivity to a particular activity under specific climatic

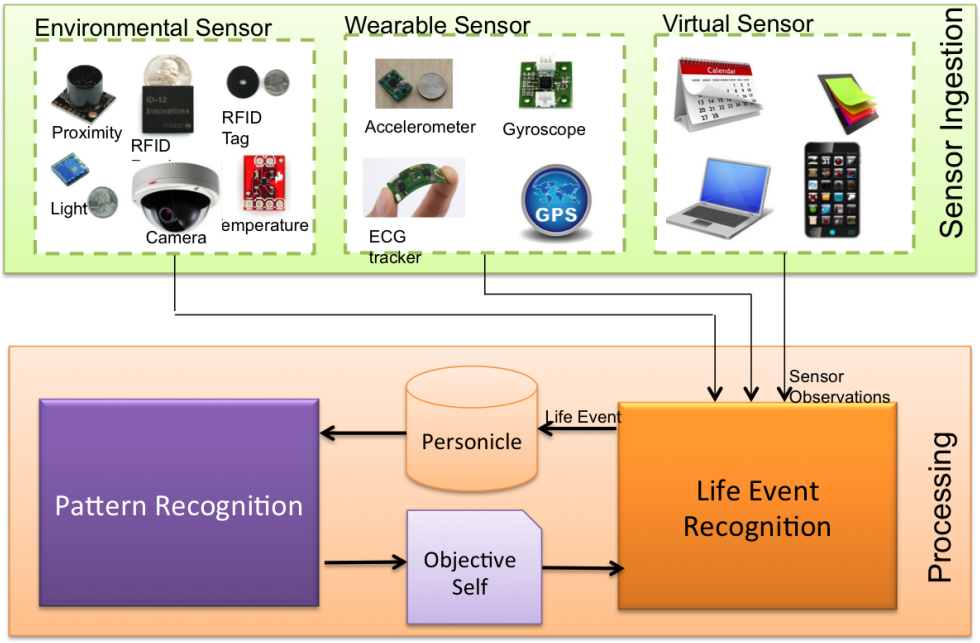


Figure 7.3: High-level architecture of an objective self system. The three main components are data ingestion, life event recognition, and pattern recognition

conditions or the effect of time and duration of exercise on one's sleep patterns and sleep quality.

7.4 Life Events

Most machine algorithms decompose multimedia content to data segments such as shots, scenes, etc., and index them using low-level feature vectors or limited higher-level metadata (e.g. visual concepts), while humans remember real life using past experience structured in form of events. So events are the basic components of how humans perceive the world, and memories are shaped through associations between the perceived events. The necessity of formal event models for the description of real life events has been acknowledged, and a number of such models have been developed [3, 124].

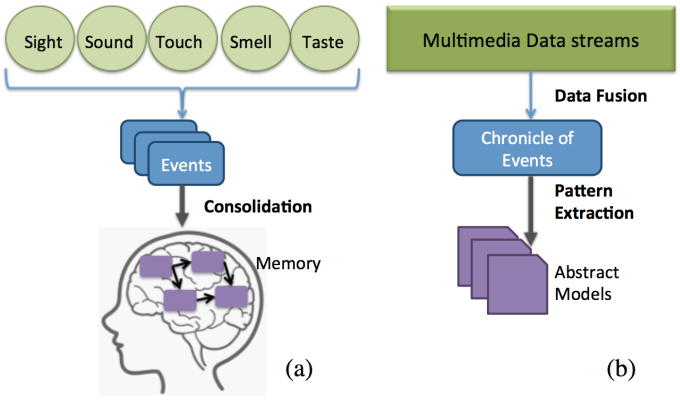


Figure 7.4: Event perception and memory recall in humans resemble multimedia data fusion and model building.

The inputs to hearing, vision, and the other senses are continuous, dynamic, and comprise huge amounts of data each second. However, human conscious experience seems to resolve into a manageable number of more-or-less discrete entities, most prominently *objects* and *events*. The term **event perception** encompasses a range of cognitive techniques involving the processing of temporally extended dynamic information [52]. In this process, our brain picks up intervals of time and distinguishes them from other intervals to form meaningful events. Moreover, Our brain tends to automatically seek patterns (relations among events) [138]. With this capability, our brain consolidate multiple events and their relations as a piece of *memory* or *knowledge*. Cognitive researchers are focused on the formation of what are known as *memory assemblies*. These are networks of neurons, connected via synapses, which can store a particular segment of a memory as an event. When a memory is recalled, its related events are assembled together to produce a whole.

As shown in Figure 7.4(b), our methodology for processing multimedia data and building models of the underlying system is in fact inspired by human's capability in analyzing multimodal data and constructing abstract models of the real world.

7.4.1 Life Log

We use Smartphone as the main source of personal information. The technological and social characteristics of smartphone make it a useful tool in behavioral analysis. The device is willingly carried by a large fraction of people and allows unobtrusive and cost-effective access to previously inaccessible sources of data on everyday activity patterns. We developed an Android-based lifelog app that collected data continuously without user intervention. Table 7.1 demonstrates the type of sensors utilized in this study and the information derived from them. Location data has venue_name and venue_type attributes for a specific latitude and longitude coordinate (e.g. name: Panini, type: restaurant). Venue type information is obtained from Google place API by using latitude and longitude data of Google-play-service API. Media is a boolean attribute that monitors whether the user listens to music or watches video. Transition is a boolean attribute that determines whether user's location has changed from a certain venue type to another between two contiguous 5-min intervals. These attributes are collected for each 5-minute interval. Interval segments are then fed to life event recognition module.

Table 7.2 shows a selected group of life event and their corresponding recognition method. The life event recognition module can either recognize one of the events in the first category with *context fusion* technique, or return an *unknown* event. Contextual information related to application usage on a smartphone can be utilized to determine one of the events in the second category of life events. We assume that life events in the first category are mutually exclusive in a sense that two (or more) of them can not happen simultaneously in a 5-minute interval. However, they can happen in parallel with life events in the second category, e.g. a user might check her email while attending class.

Table 7.1: Data streams from smartphone and list of derived attributes from each stream

Data Stream	**Attribute**
time	time_window, unix_timestamp, weekday/weekend
activity	activity_type ⊂ {standing still, tilting, walking, running, bicycling, and in vehicle}, activity_level ∈[0,4]
location	latitude, longitude venue_name, venue_type
step	step_count
application	app_name
photo	photo_count
light	light_value ∈[0,1000]
phone status	screen_on, screen_off
media	play_time
sound	sound_setting ⊂{(silence, vibration, or bell)}
call	call_type ⊂{(missed, rejected, incoming or outgoing)}
transition	changes in venue type

7.4.2 Life Event Recognition

As the number and ubiquity of sensors have grown phenomenally in recent years, human activity recognition using wearable sensors and mobile phones attracted much attention. There is a rich body of research on activity recognition using Micro-Electro-Mechanical Systems (MEMS) sensors [18]. Lately some multisensor approaches for smartphone-based activity recognition have been developed. A comprehensive survey about activity recognition using mobile phones can be found in [64]. However, these techniques mostly capture low-level

Table 7.2: Selected list of life event. Category (a) shows life events derived from sensor fusion and category (b) shows life events results from raw context data from smartphone.

Category	Method	Life event
(a)	Context Fusion with FCA	Studying, Sleeping, Vehicle Transportation, Dining, Attending Class, Walking, Running, Cycling, Exercise, Leaving Home, Arriving Home
(b)	Raw Context	Interacting with Phone, Surfing Web, Social Networking, Checking Email, Sending SMS, Phone Call, Watching Video, Skype Call

physical motion of a user. There is a significant amount of work that uses location sensors to extract high-level information about person's activities. Routinely visited locations such as home, work, or school can indicate pursued activities such as leisure, working, or picking up someone [80]. However, they do not combine location information with other sensor data from smartphones to detect high-level life events. In an effort to tap the full potential of smartphones for context-aware application, Dey et al. [37] introduce a conceptual context toolkit called *CORTEXT*. They present a mobile data-logging toolkit, AWARE [43], which provides capturing, inferring, and generating context on mobile devices. *CORTEXT* allows researchers to define rules, contexts, and services by integrating sentient objects and an event-based communication protocol [20, 43].

Considering Figure 7.2 again, suppose we have M data streams DS_1, DS_2,... DS_M each coming from a heterogeneous source of information. These data streams are divided into chunks of equal length, called time window T_i. The information obtained from the sensors varies in many respects. Methods to convert data to information and the reliability of information could be entirely different for different sensors. To convert measurements to attributes, complex approaches involving inverse mapping are used. Thus, we can say that $a_i = \mathcal{F}(m_i)$, where a_i is the attribute derived from a measurement m_i using the function

$\mathcal{F}$ for this attribute within a specific time window T_i. The function used to convert the measurement to an attribute could be simple or extremely complex. The followings are some examples of these functions:

- **Mathematical and statistical techniques** can extract basic signal information from raw sensor data, such as mean, variance, standard deviation, median, maximum, minimum, signal correlation and correlation-coefficient, zero-crossing, DC component, spectral energy, entropy, and wavelet analysis.
- **Mapping techniques** can invert geo-coding to extract place categories.
- **Natural-language processing** can help process calendar entries, computer activity logs, and so forth.
- **Machine learning techniques** include a variety of algorithms using classifiers that have been applied to the problem of recognizing physical activities from accelerometer data. Also, it is possible to classify ambient environment by analyzing light sensor readings, Wi-Fi access points, or GPS signals.

Table 7.3 shows some sensor modalities and functions that extract attributes from measurements. This table is indicative and not exhaustive, and more sensors and functions can be added as needed.

Recognizing life events from human physical activity and surrounding contexts is a challenging problem. Learning-based approaches are good at recognizing low-level physical activities (e.g. walking, jogging, etc.) from a limited number of wearable sensors. However, it is difficult to incorporate domain knowledge in their learning process and extract more high-level semantics related to life events. Moreover, collected observational data is noisy and there is not enough **labeled** data available for training purposes since labeling human activities are very tedious. Hence, We propose the use of Formal Concept Analysis (FCA) for data fusion

Table 7.3: Sensor modalities, measurements, and attributes.

Sensor	Measurement	Attribute	Function
Accelerometer, gyroscope	X,Y,Z values	Activity: {sitting, standing, walking, jogging, cycling, ascending stairs, descending stairs, lying, driving, typing on keyboard, etc.} Activity level: {low, medium, high}	Low-level activity classification
Light	Light intensity value	Location: {Indoor, semi-indoor, outdoor} Value: {Light on, light off }	Environment classification
GPS	Latitude, longitude	Location: {home, work, store, restaurant, mall, etc.}	Significant place recognition
RFID reader/ passive RFID tags	Measurements for reader/tag	Interaction with objects :{fork, spoon, mug, appliances, stove, etc.} Indoor location: {kitchen, bedroom, living room, office, etc.}	RFID motion detector
Calendar	Calendar entries	Scheduled events: {meeting, seminar, appointment, party, etc.}	Natural-language processing
Computer logs	Mouse movements, keyboard logs, windows opened/closed	Computer use: {email, Web browsing, document interaction, video watching, gaming}	Natural-language processing
Cell phone logs		Cell phone use: {making/receiving calls, apps running in the background, interaction with specific app}	
Camera	Pixel intensity values	Key frames, objects in the image	Object and event recognition
Microphone	Phonemes	Background sound: {one person talking, multiple people talking, background noise}	Speech recognition, audio scenes

and life event recognition. Concept lattices are very effective when enough labeled data samples are not available for supervised machine learning algorithms, but human knowledge is available to develop classification approaches for recognition. In the next section we explain how Formal Concept Analysis (FCA) can be utilized as an unsupervised learning technique to fuse noisy multimodal data and assign an event tag to each 5-minute interval in time-series data. In the second pass, these 5-minute intervals are merged based on their event label and the resulting event stream is fed as an input to *frequent pattern extraction* module.

7.4.3 Formal Concept Analysis

FCA was introduced by Wille et al. [49] in 1982. FCA is a way of deriving partially ordered set (poset) from pairs of objects and attributes. With all possible poset of pairs, it builds a concept lattice, which is a graphical representation of the partially ordered knowledge, to find the most similar structure to an input pair. Therefore, FCA does not need to use any statistical calculations, but rather uses the structural similarities to produce results, even under uncertainty.

Definitions

The theoretical foundation of concept lattice relies on the mathematical lattice theory [22]. Concept lattice is used to represent the order relation of concepts.

Definition 1: A context is defined by a triple (G, M, R), where G and M are two sets, and R is a relation between them. The elements of G are called objects, while the elements of M are called attributes.

For example, Figure 7.5 represents a context in form of a cross table. $G(o_1,o_2,o_3,o_4,o_5,o_6,o_7)$ is the object set and $M(a_1,a_2,a_3,a_4,a_5,\ a_6,a_7,a_8,a_9)$ is the attribute set. The crosses in the table describe the relation R between G and M, which means an object verifies an attribute.

Definition 2: Given a subset $A \subseteq G$ of objects from a context (G, M, R), we define an operator that produces the set A' of their common attributes for every set $A \subseteq G$ of objects to know which attributes from M are common to all these objects:

$$A' = \{m \in M \quad | \quad gRm \forall g \in A\}$$

Dually, we define B' for subset of attributes $B \subseteq M$, B' denotes the set consisting of those objects in G that have all the attributes from B:

$$B' = \{g \in G \quad | \quad gRm \forall m \in B\}$$

These two operators are called the Galois connection of (G, M, R). These operators are used to determine a formal concept.

Definition 3: A formal concept of the context (G, M, R) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A = B'$, and $B = A'$. A is called extent and B is called intent.

So if B is an attribute set, B' is an object set, and then $(B')'$ is an attribute set. So the following axioms hold:

$$B \subseteq M \Rightarrow B'' \subseteq M$$

$$A \subseteq G \Rightarrow A'' \subseteq G$$

Definition 4: If (A_1, B_1) and (A_2, B_2) are concepts, $A_1 \subseteq A_2$ (or $B_2 \subseteq B_1$), then we say that there is a hierarchical order between (A_1, B_1) and (A_2, B_2). All concepts with the hierarchical order of concepts form a complete lattice called concept lattice.

To map the above definitions to life event recognition problem, we consider life events as objects, and sensor measurements as attributes. The lattice algorithm to build formal concepts and concept lattice plays an essential role in the application of concept lattice. Many algorithms for generating concept lattices are published such as Ganter (NextClosure) [48], Gerhand [107], Norris [105], and Valtchev [137]. Kuznetsov et al. [74] performed a performance analysis of these algorithms and gave preference to Ganter's algorithm with respect to time complexity.

	a1	a2	a3	a4	a5	a6	a7	a8	a9
o1	X		X	X			X		X
o2		X			X				X
o3			X			X			
o4		X				X			
o5				X			X	X	
o6	X		X	X					
o7					X	X			X

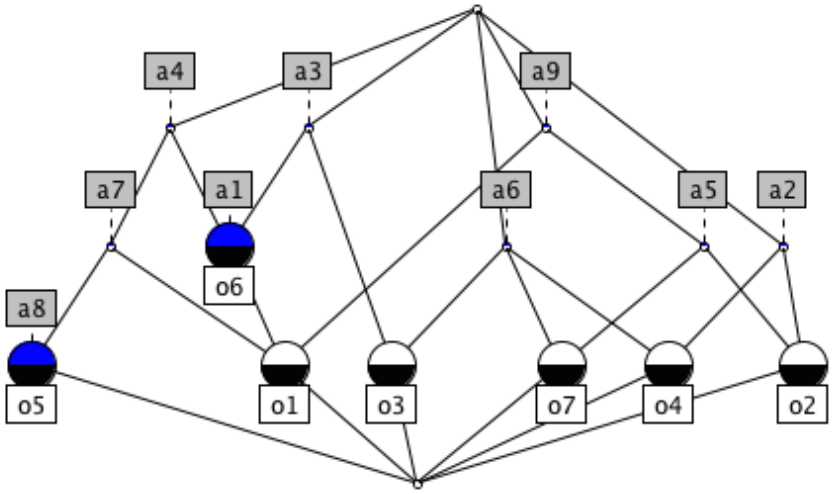


Figure 7.5: An example of context (G, M, R) and its equivalent concept lattice.(a) Sample cross table defining relation between a set of objects and attributes.(b) Concept lattice derived from cross table by applying NextClosure algorithm.

Table 7.4: Cross table of generalized relationship between life events and their attributes

	Time								Activity												Location																											Sensor		Fun																Setting		Other	
	time_band						week		physical_activity						activity_level						venue_type																											light		media		app_usage			photo_taken				sound		Call	
Attr / Life Event	T:B_0	T:B_1	T:B_2	T:B_3	T:B_4	T:B_5	W_0	W_1	P:A_0	P:A_1	P:A_2	P:A_3	P:A_4	P:A_5	A:L_0	A:L_1	A:L_2	A:L_3	A:L_4	home	school	university	acm. buildi	entertainm	gym	nightlife spo	outdoors	store	mall	shop	food	café	restaurant	residential	E:L_0	E:L_1	E:L_2	M_0	M_1	A:U_0	A:U_1	A:U_2	A:U_3	T:P_0	T:P_1	T:P_2	T:P_3	S_0	S_1	C:L_0	C:L_1
Sleeping	x	x	x			x	x	x	x						x					x														x	x														x			x
Studying				x	x	x	x	x		x	x					x	x				x	x	x												x	x														x			x
Vehicle Commute		x	x	x	x	x	x	x		x	x					x	x					x					x								x	x	x	x	x		x	x	x	x				x	x	x
Dining		x	x	x	x	x	x	x		x	x					x	x							x		x				x	x	x			x	x	x	x	x	x	x	x	x	x				x	x	x
Attending Class		x	x	x			x			x	x					x	x				x	x	x												x														x			x
Walking	x	x	x	x	x	x	x	x			x	x					x	x						x	x	x	x	x	x						x	x	x	x	x	x	x	x						x	x	x
Running	x	x	x	x	x	x	x	x	x			x						x	x						x		x								x	x	x		x	x								x			x
Cycling	x	x	x	x	x	x	x	x					x					x	x						x		x								x	x	x		x	x								x	x		x
Exercising			x	x	x		x	x						x					x						x		x								x	x													x			x

Lattice Construction Algorithm

NextClosure algorithm by Ganter [48] is one of the most well known algorithm in FCA. Figure 7.5 shows a lattice constructed from the cross table using this algorithm. The principle of NextClosure algorithm uses the characteristic vector which represents arbitrary subsets A of M, to enumerate all concepts of (G, M, R). Given $A \subseteq M$, M=$\{a_1, a_2, \ldots, a_m\}$, $A \rightarrow A''$ is the closure operator. The NextClouser algorithm proves that if we know an arbitrary attribute subset A, the next concept (the smallest one of all concepts that is larger than A) with respect to the lexicographical order is $A \oplus a_i$, where $\oplus$ is defined by:

$$A \oplus a_i = (A \cap (a_1, ..., a_{i-1}) \cup a_i)''$$

where $A \subseteq M$ and $a_i \in M$, a_i being the largest element of M with $A < A \oplus a_i$ by lexicographical order. In other words, for $a_i \in M \backslash A$, from the largest element to smaller one of $M \backslash A$, we calculate $A \oplus a_i$, until we find the first time $A < A \oplus a_i$, then $A \oplus a_i$ is the next concept.

We apply NextClosure algorithm to build a comprehensive lattice from our pre-defined cross table (as shown in Figure 7.4) that capture the relation between life events and their common attributes. Considering $|L|$= number of life events, and $|A|$= number of attributes, the time complexity of building the lattice is $O(|L|^2 \times |A|)$.

Lattice Navigation Algorithm

Once the lattice is constructed from a pre-defined set of concepts, the next step is to recognize life events using such constructed lattice from observational sensor data. We use backtracking depth first search algorithm for this purpose. To identify a life event, the system collects all the perceptible context information and feed them to the lattice. If the context satisfies the conditions of a life event, then it is identified. For example, an incoming life-log information, $S_{lifelog}$ = {00:00 - 03:59, week, standing_still, sedentary, home, enviornmental_light_low, media_false, app_no_use, photo_no_use, sound bell, call_no_calling}, will navigate the concept lattice by following the backtracking algorithm and *sleeping* event will be predicted.

7.5 Frequent Behavior Pattern Extraction

After fusing multimodal sensor observations into a human understandable set of events, the next step is analyzing these events to understand the co-occurrence relation between them. In this work we aim to infer higher level *behavior patterns* from chronicle of life events, where life events are recognized from longitudinal log of context data collected by smartphones. User behavior patterns can be expressed in a number of different ways. For example, sequence mining algorithms may uncover sequence of user activities that occur frequently [30], while statistical correlation functions may express interesting relationships between numerical values such as activity level and sleep quality [61]. In this work, the bottom-up pattern recognition techniques are employed to discover frequent event co-occurrence patterns as sequential and parallel relations among events, indicating which life events or environmental events frequently occur together. For example:

```
1) Driving is FOLLOWED BY Meeting WITHIN 1 hour
2) Browsing-Web happens WHILE Attending-Class
3) Low-ActivityLevel happens WHILE High-Temperature
```

Formulation and recognition of such sequential (number 1) and parallel patterns (number 2 and 3) are explained in the next section in details. Our long term vision is to use environmental variables and longitudinal user data to infer diverse frequent patterns that capture different aspects of the user's behavior. This framework that doesn't require labeled data, it effectively utilize noisy and incomplete data, and pull together data from multiple modalities to find interesting behavior patterns.

7.5.1 Co-occurrence Behavior Patterns

We define co-occurrence behavior patterns that encode temporal and structural relationships between life events. As mentioned in chapter 3, there are two types of co-occurrence between events:

Sequential co-occurrence: For a pair of events E_i and E_j, if E_j usually occurs after E_i (within a specific time lag Δt), these two events are considered to be co-occurring. For Objective Self application and for the ease of understanding, we represent sequential co-occurrence behavior as an association rule in form of $E_i \xrightarrow{\Delta t} E_j$ and reads E_i is *followed by* E_j *within* Δt.

For example $VehicleCommuting \xrightarrow{1hour} Meeting$ specifies that within an hour of commuting with car, user attends a meeting.

As before, the **confidence** of this size-2 pattern is defined as:

$$Confidence(E_i \xrightarrow{\Delta t} E_j) = \frac{Count(E_i \xrightarrow{\Delta t} E_j)}{Count(E_i)}$$

The confidence of sequential pattern essentially means, from all the time that the first event (E_i) happens, how many times it is followed by the second event (E_j) within Δt.

Concurrent co-occurrence: Two events are considered co-occurring if a non-empty time overlap exists between them. Concurrent co-occurrence behavior pattern is represented as

$E_i \Vdash E_j$ and reads E_i happens *while* E_j.

For example $CheckingEmail \Vdash Meeting$ specifies that user is checking email while attends a meeting. As before, the **confidence** of this pattern is defined as:

$$Confidence(E_i \Vdash E_j) = \frac{Count(E_i \Vdash E_j)}{\frac{1}{2}(Count(E_i) + Count(E_j))}$$

Two processing algorithms introduced in chapter 5 are used for counting occurrence number of sequential and concurrent patterns. Time constraint information is critical in co-occurrence definition and detection. In co-occurrence pattern detection, these constraints will be handled and evaluated in *time states* within a finite state automaton.

7.5.2 Processing Co-occurrence Patterns

For data-driven co-occurrence analysis we designed an algorithm that process multiple patterns with only one pass through the data. Life events create an event stream $ES^{(1)} = \{e_1^{(1)}, e_2^{(1)}, ..., e_N^{(1)}\}$ where each life event $e_k^{(1)} \in \{pE, iE, sE\}(1 \leq k \leq N)$, and N is the number of life event types in the application. We also have an environmental event stream $ES^{(2)} = \{e_1^{(2)}, e_2^{(2)}, ..., e_M^{(2)}\}$ where each environmental event $e_k^{(2)} \in \{pE, iE, sE\}(1 \leq k \leq M)$, can be a point event or interval event to capture state and state transition in environmental data. M is the number of environmental event types in the application. Hence we create a multi-event stream $ES = \{ES^{(1)}, ES^{(2)}\}$ to encode or serialize all event types in the system.

With N life events, N^2 patterns need to be analyzed to compute co-occurrence value between each pair of life events. Considering different time lags, the total number of possible patterns increase exponentially. Each candidate pattern is derived as follow:
$\forall e_i^{(1)}, e_j^{(1)}$ such that $e_i^{(1)}, e_j^{(1)} \in ES^{(1)}$, sequential candidate pattern $\rho = e_i^{(1)} \xrightarrow{\Delta t} e_i^{(1)}$, and concurrent candidate pattern $\rho' = e_i^{(1)} \Vdash e_i^{(1)}$

Table 7.5: Sample sequential and concurrent candidate patterns

Sequential Candidate Pattern	**Concurrent Candidate Pattern**
Studying $\xrightarrow{\Delta t}$ Vehicle Transportation	SMS $\amalg$ Attending Class
Arriving Home $\xrightarrow{\Delta t}$ Sleeping	Web Surfing $\amalg$ Attending Class
Vehicle Transportation $\xrightarrow{\Delta t}$ Attending Class	Interacting with Phone $\amalg$ Studying
Vehicle Transportation $\xrightarrow{\Delta t}$ Dinning	Dinning $\amalg$ Social Networking
Dinning $\xrightarrow{\Delta t}$ Arriving Home	Interacting with Phone $\amalg$ Exercise
Leaving Home $\xrightarrow{\Delta t}$ Exercise	Social Networking $\amalg$ Attending Class
Leaving Home $\xrightarrow{\Delta t}$ Walking	Walking $\amalg$ Phone Call
Leaving Home $\xrightarrow{\Delta t}$ Cycling	Vehicle Transportation $\amalg$ SMS
Cycling $\xrightarrow{\Delta t}$ Attending Class	Low_activity_level $\amalg$ Temperature_stays_high
Vehicle Transportation $\xrightarrow{\Delta t}$ Temperature_stays_high	Vehicle Transportation $\amalg$ Pollution_stays_high

Given M environmental events, $N \times M$ patterns need to be analyzed to compute co-occurrence between a pair of life event and environmental event. In this case, the candidate patterns are derived as:
$\forall e_i^{(1)}, e_j^{(2)}$ such that $e_i^{(1)} \in ES^{(1)}, \quad e_j^{(2)} \in ES^{(2)}$, sequential candidate pattern $\rho = e_i^{(1)} \xrightarrow{\Delta t} e_i^{(2)}$, sequential candidate pattern $\rho' = e_j^{(2)} \xrightarrow{\Delta t} e_i^{(1)}$, and concurrent candidate pattern $\rho'' = e_i^{(1)} \amalg e_j^{(2)}$.

Table 7.5 shows some samples of the sequential and concurrent candidate patterns. We analyzed different time lags with multiple temporal resolutions from *minutes*, *hours*, and *days* to *weeks*. As mentioned in Table 7.2, life events of category (a) can only happen in parallel with life events of category (b).

7.6 Evaluation

In previous sections we explained how multiple sources of information (e.g. location, motion, etc.) can be semantically fused to recognize activities of a person in form of life events and how effective models can be built by finding significant sequential or concurrent co-occurrence patterns. In this section we evaluate the applicability of our framework in understanding behavior of individuals using Smartphones as the main source of personal information. The technological and social characteristics of smartphone make it a useful tool in behavioral analysis. The device is willingly carried by a large fraction of people and allows unobtrusive and cost-effective access to previously inaccessible sources of data on everyday activity patterns. Also the changes of these patterns through time and the effect of environmental conditions on such patterns can be investigated.

7.6.1 Data Collection

We developed an Android-based lifelog app that collected data continuously without user intervention. Table 7.1 demonstrates the type of sensors utilized in this study and the information derived from them. All participants in the study were voluntarily recruited from a mobile Programming class, a computer science programming class offered to both undergraduate and graduate students during spring quarter in 2014. 65 students enrolled in the class and 31 joined the study. As the quarter progressed, 8 users dropped out of the study. From 23 remaining users, 12 were undergraduate and 11 were graduate students. Users used their own Android devices to run our lifelog app and carried the phones with them throughout the day. Data was collected without any user interaction and uploaded to the cloud daily. The data collection phase lasted for 4 weeks for the class. However, 4 users continued to collect data for another 4 weeks.

GPS tracking generates large sets of geographic data that need to be transformed to be useful for life event recognition and behavior analysis. We are interested in the type of a location/venue that a user visits rather than its coordinates. The reverse geo-coding is done by using Google place API. In other words, our behavior analysis system distinguishes between locations only if it helps determine what the user is doing. The venue categories we incorporated in this study are:*arts and entertainment, collage and university, academic building, gym, nightlife spot, outdoors and recreation, store, mall, shop, food, cafe, restaurant.* Also we asked users for their *home* location once the app was launched. Figure 7.6 shows location distribution for 7 users based on the percentage of time they spent at each location. The amount of time spent at different places reveals a lot of information about a user. For instance Figure 7.6 suggests that user 1 spent the majority of her time at home while user 3 spent considerable time at school. Also user 2 visits gym frequently while food places and malls are interest for user 7 and user 4 respectively.

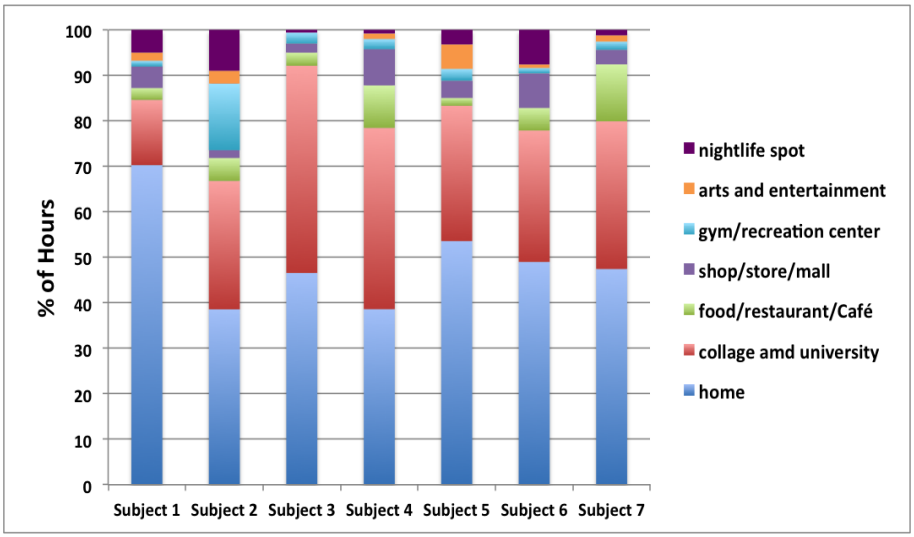


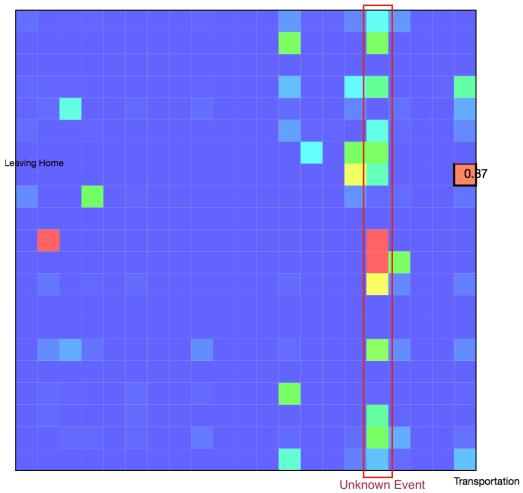
Figure 7.6: Percentage of hours multiple subjects spent at different locations.

Next we visualize sample co-occurrence patterns generated from our lifelog over 23 users with the goal of finding interesting behavior patterns at individual as well as public/group level.

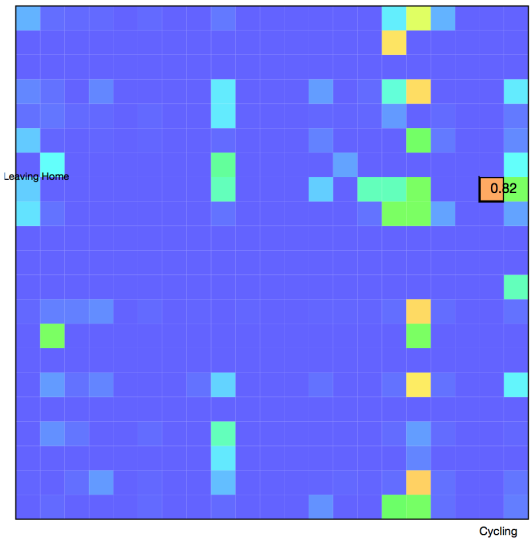
7.6.2 Sequential Co-occurrence: Commute Behavior and Activity Trends

The objective of this experiment is to find patterns of commute behavior from life event data. By generating sequential co-occurrence matrices with different temporal offsets, co-occurrences between *leaving home* and *arriving home* and commute types such as *walking*, *vehicle commute*, and *cycling* are studied. Figure 7.7 demonstrates the results for three different users. Figure 8.3a indicates commute pattern: $LeavingHome \xrightarrow{[15mins]} VehicleCommute$ with confidence value of 0.87, which means with 87% probability user used a vehicle within 15 minutes after leaving home. By increasing the temporal offset to one hour, the probability reaches 98%. Leaving home followed by cycling and cycling followed by attending class is a commute pattern observable for another user in Figures 8.3b and 7.7c. Finally, Figure 7.7d shows walking commute pattern to/from school for the third user.

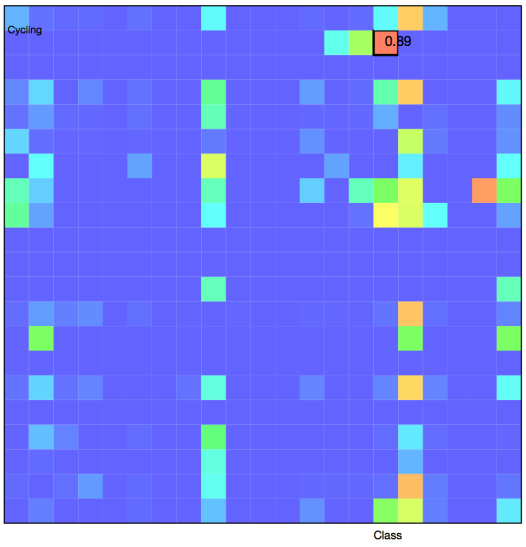
Since the study was conducted for four to eight weeks, we could analyze the change of commute behavior and activity level in some participants. Figure 7.8 and 7.9 shows sequential co-occurrence frequency of multiple patterns for two different users. The occurrence frequency of a pattern gets accumulated through time. So a sharp slope within an interval in the graph indicates the pattern was repeated often during that time, while a flat line suggests the pattern did not occur. Figure 7.8 implies that for one participant, activity level trend and vehicle transportation did not show any changes between subsequent weeks. However, a clear decrease in using bicycle for commute purposes is visible in Figure 7.9 during third and forth week, which in fact corresponds to a decrease in activity level.



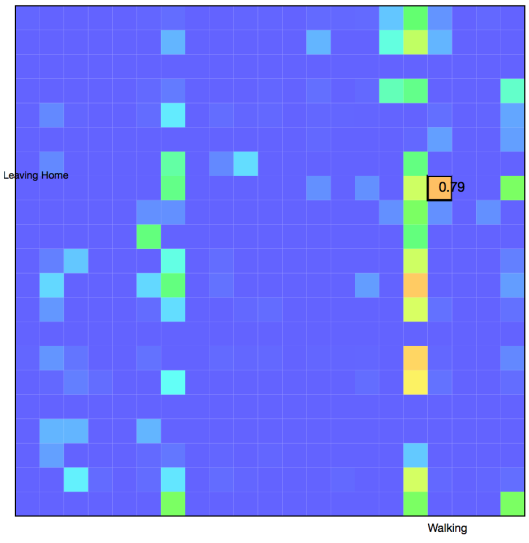
(a) Leaving home followed by Vehicle commute within 15 minutes, Confidence=0.87



(b) Leaving home followed by cycling within 15 minutes. Confidence=0.82



(c) Cycling followed by attending class within 1 hour. Confidence=0.89



(d) Leaving home followed by walking within 15 minutes. Confidence=0.79

Figure 7.7: Sequential Co-occurrence matrices. Unknown event is shown with a red box surrounding it. Each cell shows the confidence value of pattern $E_i \xrightarrow{\Delta t} E_j$.

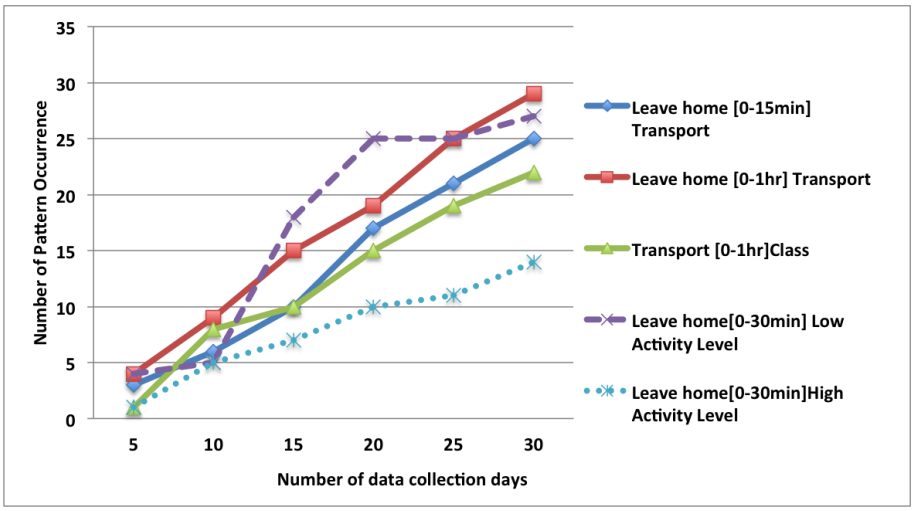


Figure 7.8: Vehicle commute and activity-level patterns. No major change is observed in commute behavior

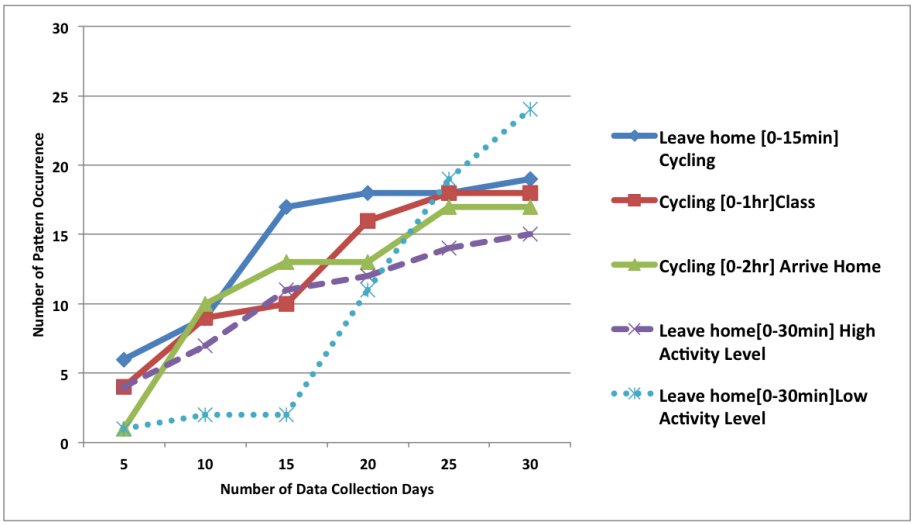


Figure 7.9: Vehicle commute and activity-level patterns. Commute behavior has changed during the second half of study.

7.6.3 Concurrent Co-occurrence: Multitasking Behavior

Concurrent co-occurrence examines the co-occurrence relation between events that are temporally overlapped. Some of the life events explained before might happen in parallel. For instance dining might be concurrent with sending text or making a phone call. In this ex-

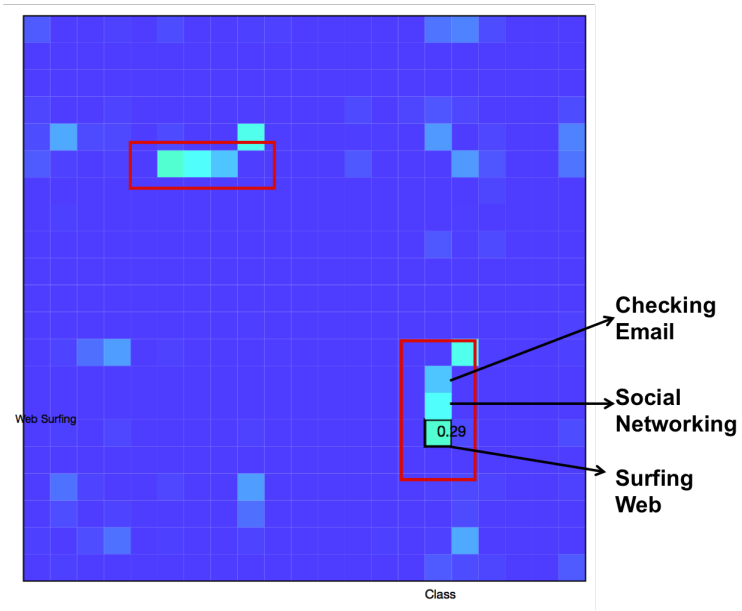


Figure 7.10: Concurrent Co-occurrence matrix visualizing co-occurrence of life events. For a pair of events, each cell shows the confidence value of pattern $E_i \parallel E_j$.

periment we used concurrent co-occurrence matrix to investigate user's interaction with the phone **while** they engage in other activities. Figure 7.10 displays an interesting result for one of the users. As shown, there is a clear co-occurrence between surfing web, checking email, and using social networking apps while the person is attending a class. This analysis reflects multitasking in the classroom and indicates the user is bored or distracted during classes.

7.6.4 Patterns Across a Group of Users

We analyze commonly occurring co-occurrence patterns across all users. Figure 7.11 shows common sequential co-occurrence patterns across multiple users using the same matrix visualization as before. However, the main difference is that in Figure 7.11 each cell shows the percentage of users that the pattern occurs in. We only show the patterns that have at least 60% confidence. Figure 7.11(a) shows sequential co-occurrence patterns (event types in x axes follow event types in y axes within an hour). The most frequent pattern occurring

in 78% users is *phone call followed by SMS within an hour*. Other common patterns are *attending class followed by studying*, *using phone followed by sleeping*, and *vehicle commute followed by arriving home*. Figure 7.11(b) shows concurrent co-occurrence patterns. The most frequent pattern in this case is *using social networking apps while dining*. The next common patterns are checking email and sending text while studying. Common pattern could be a useful way to express and use common sense knowledge about user context for a particular community of user.

7.6.5 The Effect of Environmental Factors on Behavior

It is now apparent that exposure to some air pollutant (Particulate Matter PM2.5) has consequences for human health and life expectancy [121]. Exposure to fine particulate matter is particularly dangerous since these small particles penetrate deep into the lungs and may also affect other aspects of individual's life. This experiment is devoted to examine the associations between certain environmental conditions and human behavior. The main question is whether short term exposure to PM2.5, or certain climate change conditions has any bearing on an individual's physical activity, and does it cause any deviation in routine activities? By GPS tracking, the closest pollution and weather stations to user's current location is found and ambient temperature and PM2.5 data is collected. Unified event streams are then constructed by abstracting trends of time-series data using Symbolic Aggregate approXimation (SAX) algorithm [82] with alphabet of size 3 (a, b, c symbols). For each data stream, 7 event types (ab/bc= increase, ac=suddenly increase, ca = suddenly decrease, cb/ba=decrease, aaa = stay low, bbb= medium, ccc= stay high) are defined as color coded in Figure 7.2. After processing sequential co-occurrences and concurrent co-occurrences between life event stream and PM2.5/temperature event stream, we found a few occurrences of the following patterns for multiple users:

$ActivityLevel - stays - low \Perp Temperature - increasing$

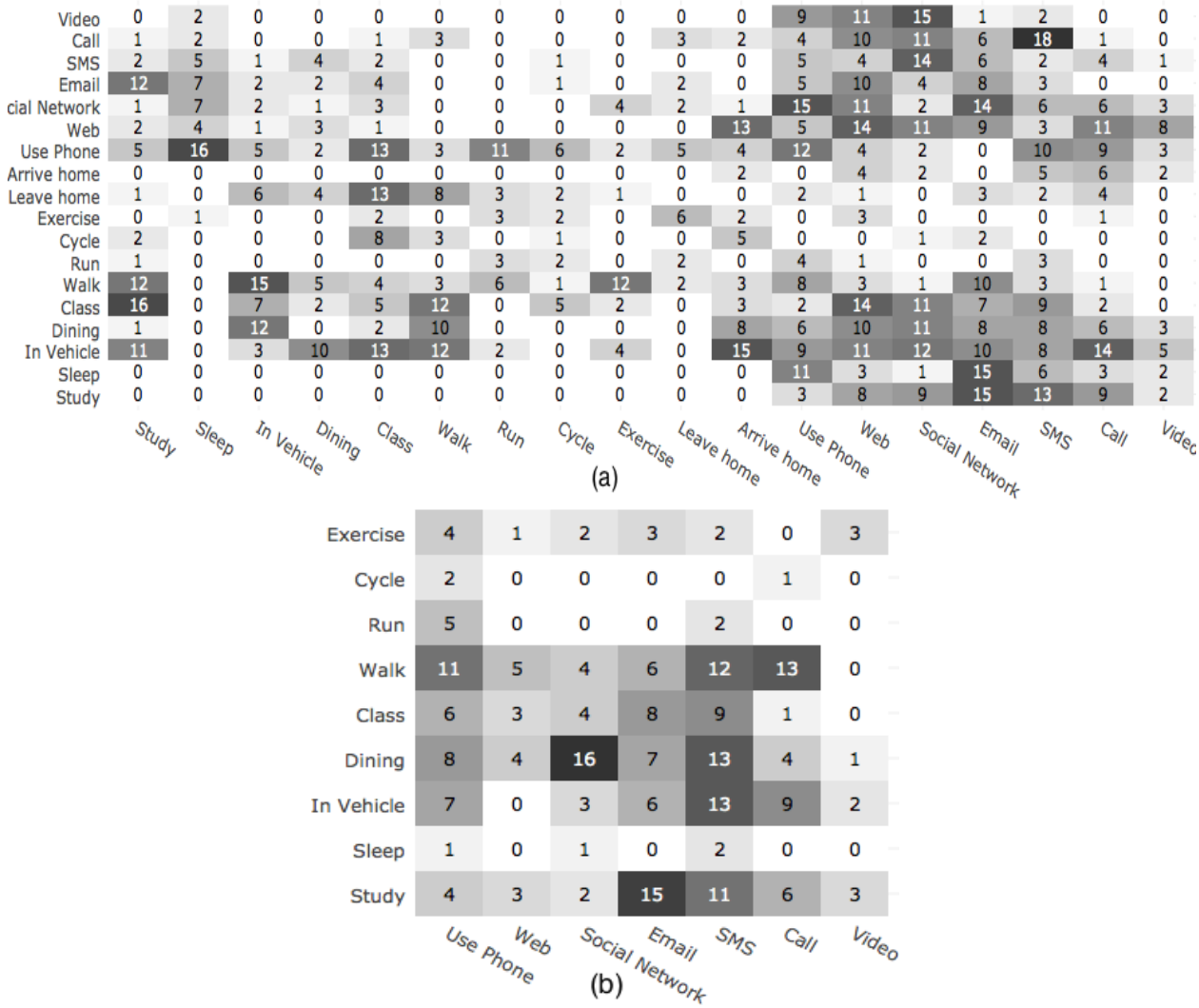


Figure 7.11: Common co-occurrence patterns. For each pattern, we show the percentage of users the pattern occurs in. (a) sequential co-occurrence patterns $E_i \xrightarrow{\Delta t} E_j$ across all users, where Δt=1 hour. (b) concurrent co-occurrence patterns $E_i \Subset E_j$.

$$PM2.5 - suddenly - increase \xrightarrow{3hrs} Vehichle - commute$$

$$Walking \perp\!\!\!\perp PM2.5 - stay - low$$

Although the above patterns might be an indication of how users commute or how active they are in different environmental situations, a longitudinal study with longer duration shall be performed to asses more reliable patterns.

Chapter 8

Asthma Risk Management

8.1 Introduction

Nowadays, multimedia sensors and technologies play an important role in seamlessly interpreting world events and to react to them in the most suitable way. Physical sensors generate massive amount of data related to our environment, from traffic and air pollution to climate change. Human sensors create vast streams of social messages related to various phenomenon in the world. Once we combine these heterogeneous sources of information new knowledge can be extracted. Novel information processing architectures and platforms should be developed to analyze and extract insight from such multimodal data. This knowledge has a high potential to transform different aspects of our lives. One aspect that can greatly benefit from this paradigm is healthcare.

Targeting care for those at highest risk of asthma attack is an attractive concept. Asthma attacks are at best unpleasant, and at worst catastrophic and even fatal. Asthma risk prediction is growing in importance both for the individual and at the public health level. Exposure to air pollution specifically Particular Matter (PM2.5) is linked with asthma exac-

erbation; however, the role played by meteorological factors such as temperature, humidity, rainfall, wind etc. and the complicated interrelations between these factors and asthma attacks are not well understood. Researchers in medical science and public health are trying to understand such relations by applying traditional statistical techniques. Regression models are the most widely used methods to conclude a positive or negative correlation between asthma attacks and individual risk factors. For example [71][47] study the relationships of asthma end points with pollutant exposures. However, results are narrow, in form of correlation coefficients and p-values, and not expressive enough beyond assessing a strong or weak positive/negative correlations. In other words traditional statistical analysis cannot fulfill the requirements of complex health care applications such as asthma management since there is neither a simple nor a linear relationship between multiple asthma risk factors.

This leaves simple yet important questions such as the followings unanswered.

- During fall season, how pollution effects asthma in rainy or windy days?
- Within how many days after a sudden increase in PM2.5 asthma outbreak happens? How the presence of different climate conditions effects this situation?

In most applications the value of heterogeneous data streams are either numerical or multivariate and categorical. For example, the value of environmental data are of a numerical type and social messages are in text format. In order to reduce problem complexity and facilitate data analysis, we should unify data representation first. Humans think in terms of events and entities. Events provide a natural abstraction of happenings in the real world. So we use an event model as an abstraction unification technique on top of raw data streams to assess meaning and information of each value on its related studies. For example, the concentration of environmental factor PM2.5 is a numerical attribute. We can define events by symbolizing their value into different levels of air pollution (interval events) or when the

value increases/decreases from one level to another (point events). Abstracting data streams to event streams makes the result of any analysis more comprehensive.

8.2 Motivation

Asthma is a lifelong disease that causes wheezing, breathlessness, chest tightness, and coughing. It can limit a person's quality of life. While we don't know why asthma rates are rising, we do know that most people with asthma can control their symptoms and prevent asthma attacks by avoiding asthma triggers and correctly using prescribed medicines, such as inhaled corticosteroids. An estimated 300 million people worldwide suffer from asthma, with 250,000 annual deaths attributed to the disease. It is estimated that the number of people with asthma will grow by more than 100 million by 2025 [32]. In United States, one in 12 people (about 25 million, or 8% of the population) had asthma in 2009, compared with 1 in 14 (about 20 million, or 7%) in 2001. More than half (53%) of people with asthma had an asthma attack in 2008.

Asthma is a chronic disease, so managing this disease effectively is best way to cope with it. To manage asthma, it's important to understand the many asthma triggers. Some major asthma triggers are: pollution, pollen, cold air, exercise, dust, smoke, pets, stress, chemical fumes, and bugs. Once one identifies and reduces exposure to the specific triggers or causes of asthma, she can take an active role in controlling her asthma and reducing the frequency of asthma attacks. Being aware of environmental, food, and inhaled allergens and avoiding them can significantly help in asthma prevention by reducing the frequency or severity of asthma attacks. If environmental pollution seems to cause one's asthma, it's important to stay indoors during periods of heavy air pollution. Overall, the best management plan for each asthmatic patient is finding the specific triggers or causes of asthma, and then plan to avoid these triggers and have better asthma control.

8.3 Related Work in Asthma Risk Factor Prediction

A great body of research in medical science apply correlation detection techniques and regression models to conclude a positive or negative correlation between asthma attacks and individual risk factors. For example [71][47] [26] study the relationships of asthma attack with pollutant exposures. Also [92][76] study the influence of weather fluctuation and asthma risk. However, reported results are in form of correlation coefficients and p-values, which are not expressive enough beyond assessing strong or weak positive/negative correlations and certainly not adequate for designing complex predictive models.

Bae et al. [16], introduce a framework that can monitor and analyze individual exposure to environmental triggers of asthma attack. The system emphasizes on using patient's locations, environmental pollution, temperature, and humidity to find correlations between the patients' health condition and the negative impact of environmental factors. Although the concept is interesting, this research has been introduced at the conceptual level, and there are no detailed experimental results to see how well an application built on this framework works in reality. In [66], an auto-regressive model applied on weather factors (e.g. temperature, atmospheric pressure, humidity) coming from physical sensors, is used to predict asthma attacks. Authors define three rules based on knowledge of asthma experts. These rules reflect the vulnerability of asthma patients to the fluctuation of meteorological factors. This research does not recognize any rules from weather or air pollution, but apply predefined rules to predict asthma attacks. Lee et al. [77] apply decision tree and association rule mining techniques to build a classification model for asthma disease. These classifiers are used in an alarm system to notify patients about asthma attack risks. In their approach, an integrated patient dataset is created by combining patient's biological data and environmental factors. This design has some serious flaws since environmental data is being repeated within each individual patient's data. Also, authors use a discretization method to assign symbols to numerical environmental values. In our approach however, we consider analyzing trends of

data as opposed to raw data and assign meaningful events to states and state transitions in trend data.

8.4 Approach

In this section we apply our interactive knowledge discovery framework to identify meaningful relations in the form of complex patterns between environmental factors and asthma exacerbation. We use messages from social networks as indicator of a real world phenomenon (asthma outbreaks); then we try to find digital footprints of potential triggers of the phenomenon in physical sensory data. Temporal structure and order relation between weather, air pollution and their effect on asthma exacerbation form complex patterns called *asthma risk factors*. By extracting such patterns we are able to create a risk prediction model that is important both for the individual and at the public health level and this can be used for environmental health decision support. For experimental evaluations, we've collected pollution and meteorological data in Tokyo and Osaka cities and analyzed complex risk factor patterns that might have resulted in asthma outbreaks.

Figure 8.1 shows the overall framework for analyzing and modeling asthma risk factors. Our goal is to detect the prevalence of asthma by finding complex interrelations between air pollution, weather, and asthma exacerbation. Once a burst in asthma related messages is detected, we'll then analyze historical physical sensory data, to find a set of complex risk factor patterns that might have resulted in the burst.

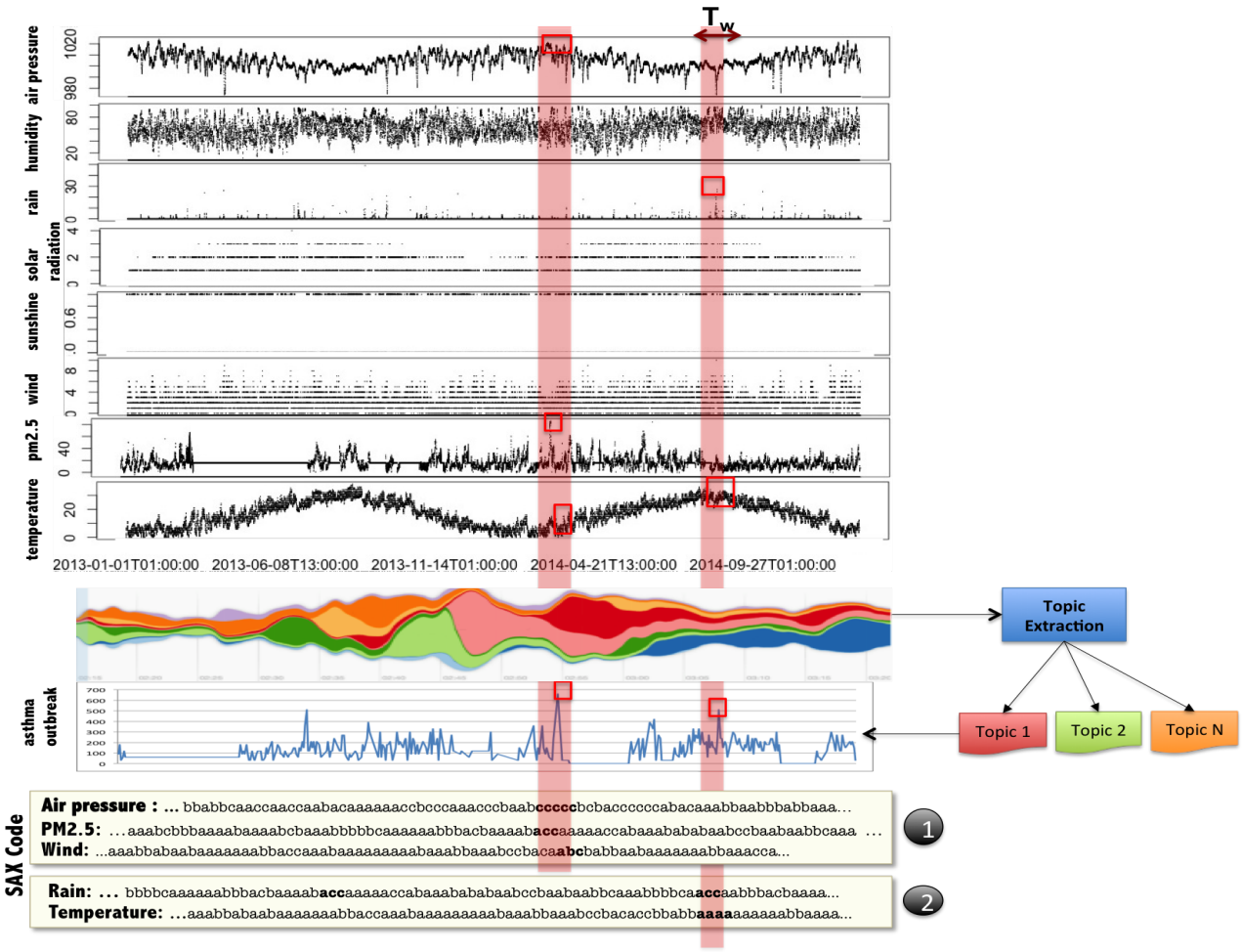


Figure 8.1: Analyzing and modeling asthma risk factors from social network data and meteorological sensor data

8.5 Data Pre-processing

Mizunuma et al. [93] shows that there is a relationship between real world events and what people tweet about. So, tweets can be leveraged for recognizing a situation in the real world. Once we have recognized a real world phenomenon from social messages then we can look for digital footprints of potential triggers of the phenomenon in physical sensory data (e.g. air pollution and meteorological factors). We use data collected for 2 years, from 2013-Jan-01 to 2015-Jan-01, in Osaka and Tokyo. This data contains continuous hourly PM2.5, temperature, rain, wind, humidity, solar radiation, sunshine, and air pressure.

8.5.1 Topic Modeling

The target of this component is to build a topic detector E_v that filters messages from social sensors that probably talk about the topic we are interested in (e.g. topic detector filters only topic 1 from social streams in Figure 8.1). In order to build the topic detector, we used Keygraph [122], a topic detection approach that considers keyword co-occurrence to detect topics from large and noisy data collections such as social media. This component is used to filter out all suitable social messages related to asthma topic and build the topic histogram as shown in Figure 8.1. The KeyGraph is the set of non-overlap sub-graphs, call communities; each community is considered as one topic or event. The significance of this methodology is that there is no need to define the number of topics beforehand. In order to build E_v of a given topic X, say "asthma", we first collect a set of X-related documents from the Internet (e.g. wikipedia, medical organizations), and scientific journal archives (e.g. ojphi.org). Then, we use KeyGraph to build a set of communities. Finally, we pick a community that contains keyword X (or at least contains one keyword that has a high semantic coherence to topic X), and collect all keywords that connect directly to keyword X as the E_v. We assume that an incoming tweet contains at least one keyword in E_v that may talk about something related to X.

8.5.2 Environmental Event Stream Modeling

The target of this component is to extract meaningful events from data streams and generate corresponding event streams. We use an event model to create an abstraction level on top of sensory data streams to mask the heterogeneity of the underlying data. For human sensory data, a burst detector [70] is applied to find bursting points on the topic histogram. For physical sensory data, first trend data is extracted from data streams. Then trend data streams are abstracted to SAX format [82].

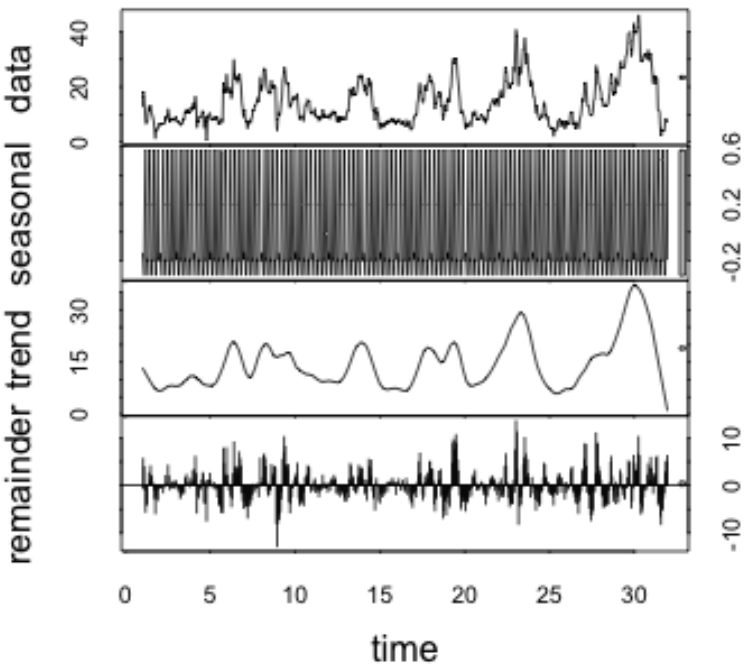


Figure 8.2: Season-Trend Decomposition by Loess on PM2.5 data stream.

In time-series data streams, trend analysis plays a major role. It not only facilitates prediction of a new value of data within a certain interval of time, but also gives a holistic view of changing values. Therefore, instead of raw data, trend data is used to extract information. The Season-Trend Decomposition by Loess (STL) method, introduced in [82], is taken into account to decompose original time-series data into Trend, Seasonal, and Remainder streams. Figure 8.2 shows such decomposition on PM2.5 observations for a duration of one month in December 2014 in Tokyo.

SAX algorithm with alphabet size equal to 3 (a,b,c symbols) is applied on trend data in order to create SAX-code streams. Given the specifications of our application, we are interested in capturing not only state transitions (e.g. pollution level increases or decreases) but also maintaining a specific state value for a duration of time (e.g. pollution level stays high). To do so, we define 6 event types for each stream. Table 8.1 shows a list of SAX codes and the

Table 8.1: Definition of events assigned to each SAX-code

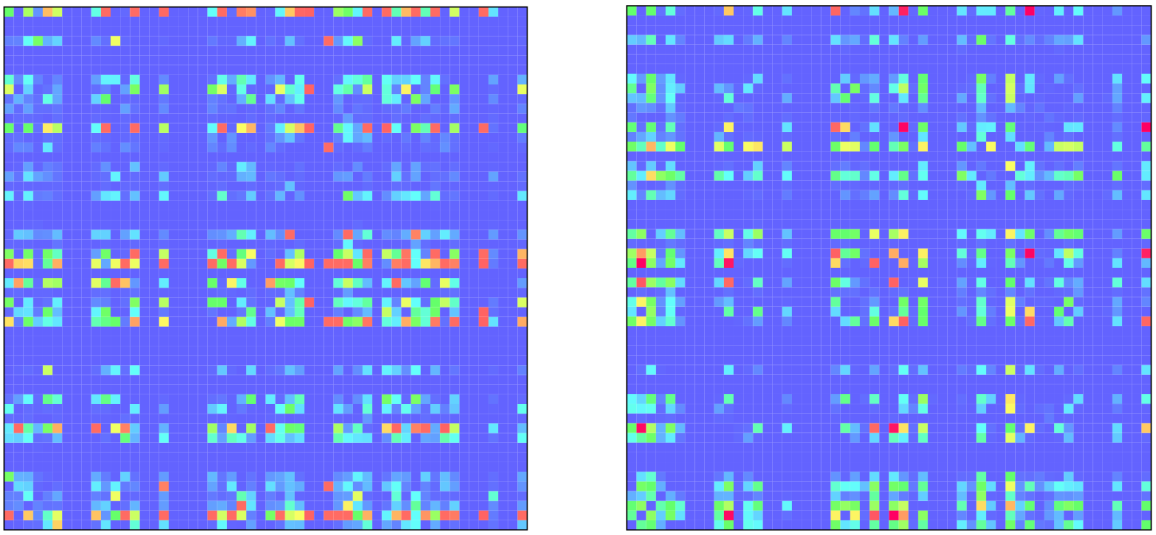
Symbol	SAX-code	Event Definition
Low Value Level=a	ab / bc	increase
	ac	suddenly increase
Medium Value Level=b	ca	suddenly decrease
	cb / ba	decrease
High Value Level=c	aaa	stay low
	ccc	stay high

corresponding event definition assigned to them. Using this encoding, each SAX-code trend data stream is converted to an event stream. These event streams are used as the input to the pattern recognition component.

8.6 Experiments

8.6.1 Data-driven Risk Factor Recognition

We are interested in finding the relation between multiple asthma triggers, mainly PM2.5 and meteorological factors. Asthma risk factor is a pattern characterizing by structural order and temporal constraints between multiple events that have resulted in an asthma outbreak. An outbreak is an increase in the frequency of a disease above what is expected in a given population. For example the pattern $\rho = ((rain\_stayHigh \between temperature\_stayHigh); asthma\_outberak)$ is considered to be a risk factor if ρ occurs frequently. Figure 8.3a displays sequential co-occurrence matrix with 5 hour time lag between all event types from all event streams



(a) (b)

Figure 8.3: (a) Shows Sequential Co-occurrence Matrix. (b) Shows Concurrent Co-occurrence Matrix.

(meteorological, pollution, and asthma outbreak event streams) in Osaka city. Concurrent co-occurrence matrix, showing which events are happening in parallel, is depicted in Figure 8.3b. For this experiment, using the framework's visual and interactive UI, we examine significant complex patterns that have resulted in asthma outbreak.

Figures 8.4 to 8.9 displays 6 dominant risk factor based on their frequency count in *Osaka*. Since risk factors are quite different between different seasons, total occurrences of the patterns for each season are displayed. RF1 shows that in summer, when solar radiation is high, the combination of low wind and high pollution cause asthma outbreak. Although in RF2 without the solar radiation cause, high wind and high pollution affect the outbreak. Low humidity in presence of pollution plays an important role in outbreak during spring while temperature fluctuation is a major factor during fall and winter. Also lack of wind while air pressure and humidity are high cause major outbreak during winter. Overall we can see that although PM2.5 has the most impact on asthma outbreak in general, its effect is not noteworthy in the fall and winter seasons. Also, temperature fluctuation has the most impact in fall and winter and it is not a risk factor during spring and summer.

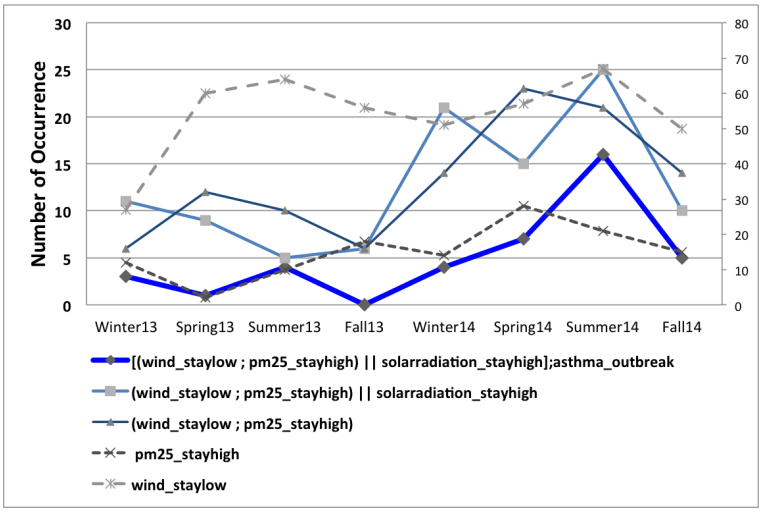


Figure 8.4: RF1=((Wind_stays_low ; PM2.5_stays_high) ⫫ SolarRadiation_stays_low)

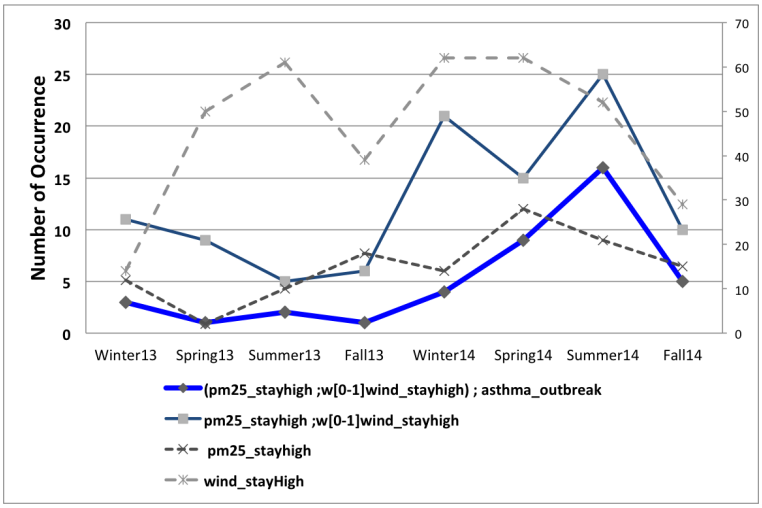


Figure 8.5: RF2= (PM2.5_stays_high ;ω [0-1 days] Wind_stays_high)

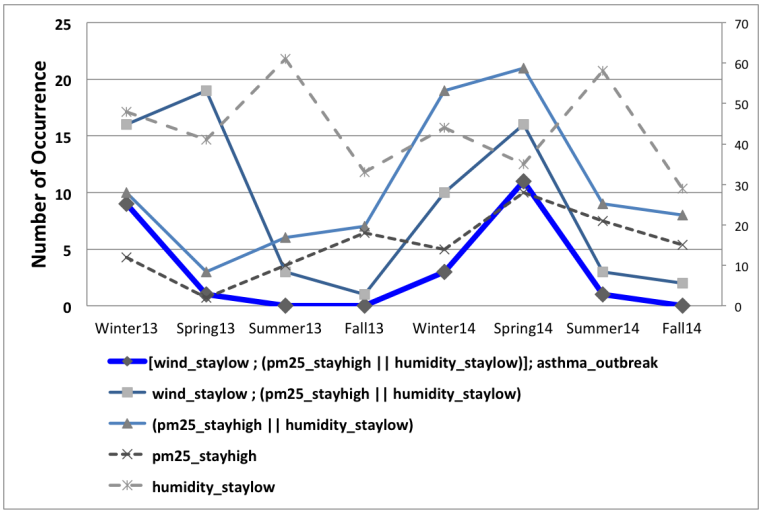


Figure 8.6: RF3= (Wind_stays_low ; (PM2.5_stays_high ⫫ Humidity_stays_low))

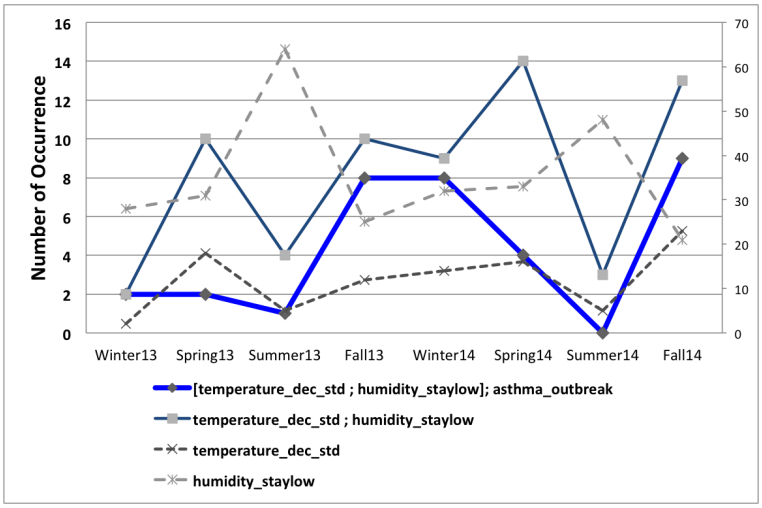


Figure 8.7: RF4= (Temperature_dec_steadily ; Humidity_stays_low)

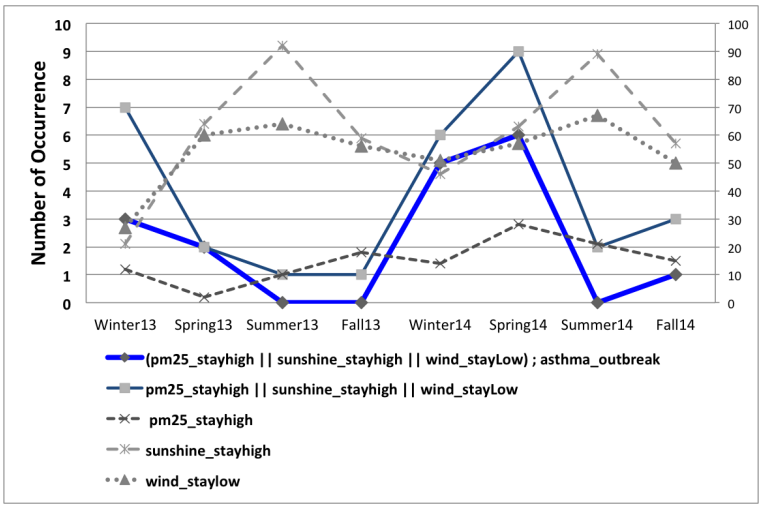


Figure 8.8: RF5= (PM2.5_stays_high ⫫ Sunshine_stays_high ⫫ Wind_stays_low)

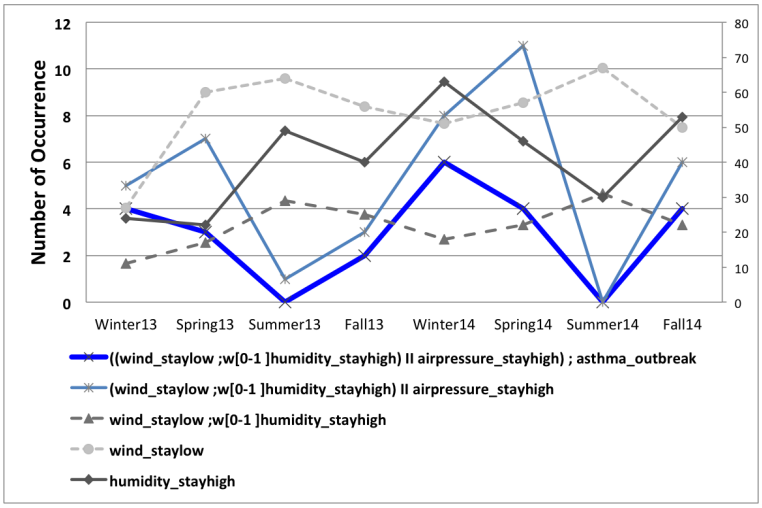


Figure 8.9: RF6= ((Wind_stays_low ; ω[0-1 days] Humidity_stays_low) ⫫ Airpressure_stays_high)

Figure 8.10 demonstrates the most significant asthma risk factors of size 2 with their frequency in total and for each season in *Tokyo*. Description of each pattern is shown in table 8.2. From this analysis we have found that although PM2.5 has the most impact on asthma outbreak in general, its effect is not noteworthy in the fall and winter seasons. Also, temperature fluctuation has the most impact in the fall and winter seasons and it is not a risk factor during spring or summer. Another interesting pattern is that during spring and summer,

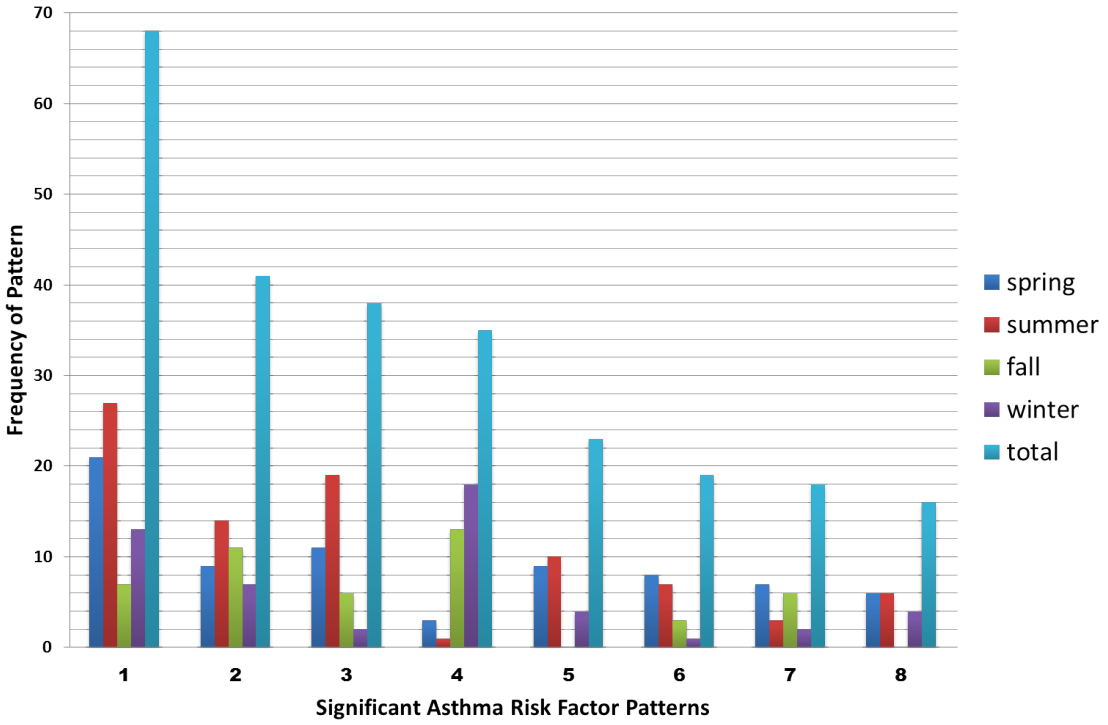


Figure 8.10: Frequency of asthma risk factors of size 2.

when rain suddenly increases to a very high level, an asthma outbreak is more probable. This might sound counter intuitive, since rain by itself should not affect asthma, unless it is combined with some other factors that we have not been included in our model. In this regard, consulting with asthma experts revealed that thunderstorms are associated with an increase in asthma exacerbation. This is because high concentrations of allergens, specially pollen, are found to coincide with thunderstorms. Our experiments do not account for the effect of allergens such as pollen on asthma outbreaks since reliable data was not available. This impacts the quality of some recognized patterns and reflects its effect on the accuracy of our model.

Table 8.2: Risk factor patterns and their corresponding pattern number from figure 8.10. (e.g. $(PM2.5\_inc \;\; ;\omega_{[0-4days]} Asthma\_outbreak)$ reads: once PM2.5 increases, asthma outbreak happens within 4 days)

Pattern number	Risk factor pattern
1	$PM2.5\_inc$;$\omega_{[0-4days]}$ Asthma_outbreak
2	$Temp\_stays\_high$;$\omega_{[0-4days]}$ Asthma_outbreak
3	$Rain\_inc\_suddenly$;$\omega_{[0-5days]}$ Asthma_outbreak
4	$Temperature\_stay\_low$;$\omega_{[0-3days]}$ Asthma_outbreak
5	$PM2.5\_inc\_suddenly$;$\omega_{[0-2days]}$ Asthma_outbreak
6	$Wind\_dec$;$\omega_{[0-3days]}$ Asthma_outbreak
7	$Wind\_inc$;$\omega_{[0-4days]}$ Asthma_outbreak
8	$PM2.5\_stays\_high$;$\omega_{[0-5days]}$ Asthma_outbreak

Fig. 8.11 shows 16 significant patterns of size 3 and 5 significant patterns of size 4 again in *Tokyo* city. These patterns encode relations between two or three events as well as the time lag between them and asthma exacerbation (e.g. when rain decreases followed by PM2.5 increase within 4 days, then asthma outbreak is probable.) The total number of occurrences of size-3-patterns declined compared with size-2-patterns. The same thing is

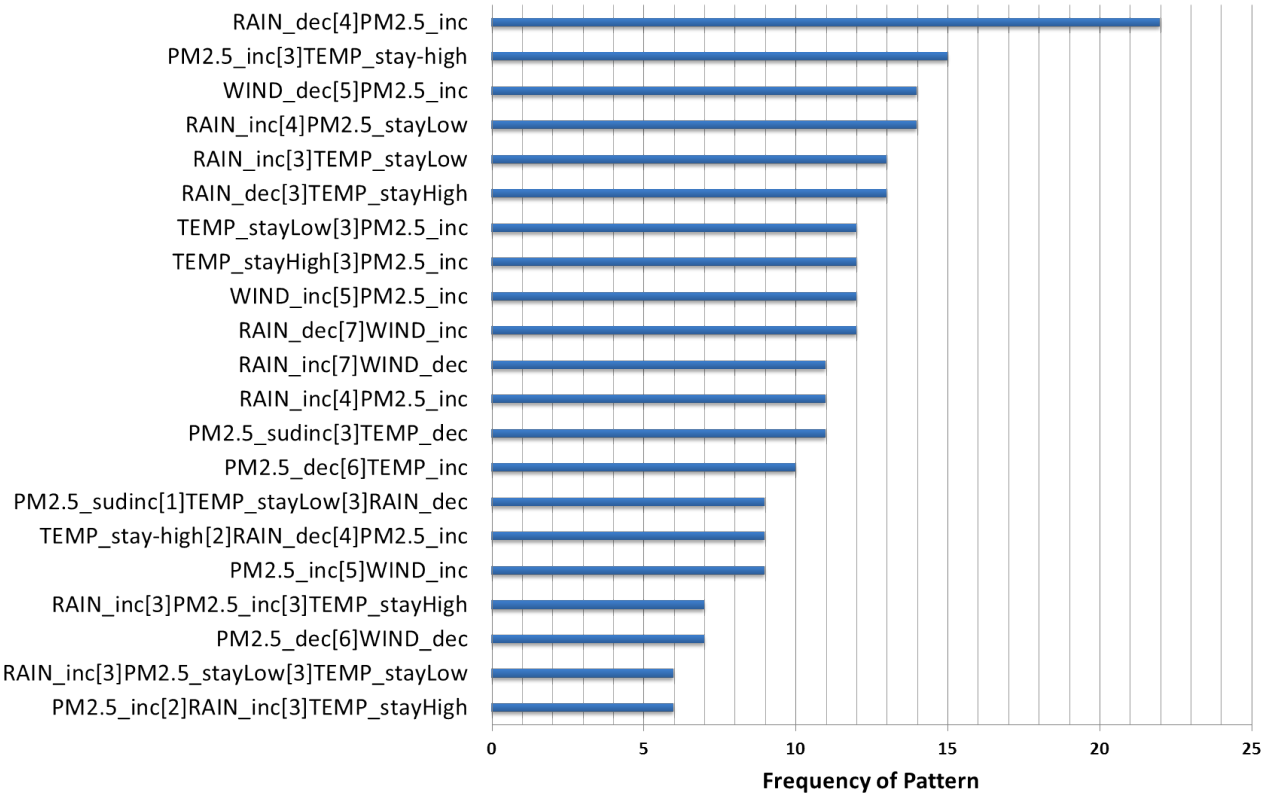


Figure 8.11: Frequency of asthma risk factors of size 3 and 4

true for size-4-patterns compared with size-3 ones. By increasing the size of a pattern, it becomes more specific, and the more specific a pattern is, the less frequent it will be in our dataset. Hence in the processing algorithm we have used different thresholds to assess the significance of patterns with different sizes. Our results suggest some interesting patterns: when PM2.5 increases followed by temperature stay high within 3 days, then asthma outbreak is probable; when wind decreases followed by PM2.5 increases within 5 days, then asthma outbreak is probable. Recognizing the time lags between events is one of the most important contributions of our approach. To the best of our knowledge this aspect of correlation between different risk factors has never been studied systematically before in healthcare applications.

8.6.2 Interactive Asthma Risk Factor Assessment

In this experiment, we investigate the effect of meteorological factors combined with other environmental conditions (specially air pollution) in asthma outbreak in *Osaka*. This process is called hypothesis refinement. Although studies have shown a relation between exposure to PM2.5 and outbreak[26][34], we found that *PM2.5_inc_steadily* event cause asthma outbreak with 54% probability. However, when air pollution increases *while* air pressure is high, the possibility of outbreak reaches 59%. So, increased air pressure is shown to be a surrogate for accumulation of PM2.5, and is more strongly associated with asthma outbreak than any other combination between meteorological factors and air pollution. Some of these interesting findings are shown in Table 8.3. For example $P21$ is not a risk factor by itself because the chances of outbreak is only 9% but when it is combined with high temperature, outbreak possibility increases significantly. Another interesting example is the probability of asthma outbreak when rain suddenly increases to a very high level while temperature is high ($P34$). This might sound counter intuitive, since rain by itself should not effect asthma, unless it is combined with some other factors that have not been included in our causal model. As we investigate it further, a pulmonologist suggested that thunderstorms are associated with an increase in asthma exacerbation specially in summer. This is because high concentrations of pollen are found to coincide with thunderstorms. Our experiments do not account for the effect of allergens such as pollen on asthma outbreaks since reliable data was not available. This impacts the quality of some causal patterns and reflects its effect on the accuracy of our model. A demo of the working framework is available at `https://youtu.be/ABx859OcFx8`.

Table 8.3: Potential risk factors and the probability of asthma outbreak before and after hypothesis refinement.

Pattern Description	Count	Co-occurrence Value
P11 = Wind_dec_suddenly ;ω[0-1day] PM2.5_stays_high	20	**(P12/P11)=0.8**
P12 = (P11) ; Asthma_outbreak	16	
P13 = (P11) ⇊ Humidity_stays_low	14	**(P14/P13)=0.92**
P14= (P13) ; Asthma_outbreak	13	
P21 = Wind_dec_steadily ;ω[0-1day] PM2.5_stays_high	171	(P22/P21)=0.09
P22 = (P21) ; Asthma_outbreak	16	
P23 = (P21) ⇊ Temperature_stays_high	12	**(P24/P23)=0.83**
P24= (P23) ; Asthma_outbreak	10	
P31 = Rain_stays_high	192	(P32/P31)=0.18
P32 = Rain_stays_high ; Asthma_outbreak	35	
P33 = Rain_stays_high ⇊ Temperature_stays_high	10	**(P34/P33)=0.7**
P34 = (P33) ; Asthma_outbreak	7	
P41 = PM2.5_inc_streadily	120	(P42/P41)=0.54
P42 = PM2.5_inc_streadily ; Asthma_outbreak	54	
P43 = PM2.5_inc_streadily ⇊ Airpressure_stays_high	32	**(P44/P43)=0.59**
P44 = (P43) ; Asthma_outbreak	19	

Chapter 9

Conclusion and Future Work

A large volume of diverse, noisy, complex, unstructured, and longitudinal data are continuously generated from diverse sensors. This pose a serious challenge to knowledge extraction from these heterogeneous temporal data. KDD field is mostly focused on extracting hidden patterns from diverse data sources and much research has been devoted to designing efficient data mining algorithms. Recently, there is much attention to more user-friendly solutions for knowledge extraction especially in complex domains such as healthcare and bioinformatics. This results in the emergence of HCI-KDD that combines Human-Computer Interaction (HCI) and Knowledge Discovery (KDD). The focus of this field is to design visualizations that use human expert in the *result interpretation phase* of KDD process. However, the main problem is how and when to incorporate human expert in the data analysis phase of knowledge discovery given the heterogeneity of input data streams and the complexity of processing tasks?

This thesis introduces a framework for knowledge discovery in complex domains by emphasizing on human-in-the-loop for pattern mining and data analysis tasks. The framework is called interactive Knowledge Discovery from Data Streams (interactive KDDS), and has

the important effect of coupling both data-driven and hypothesis-driven modeling, keeping human in the loop while taking advantage of data-driven analysis when needed (Chapter 2). The framework is based on a high-level declarative language which allows hypothesis formulation, complex pattern query, and pattern mining using the combination of a unique set of well-defined operators. To bridge the gap between the flow of information in real world and how it is captured, managed, and processed in the cyber world, we define an event model that encapsulate multiple aspects of an event. Semi-intervals are used to represent temporal facet of an event (Chapter 4). So events are *structuralization* of the timeline using semantics rather than the uniform structuralization as imposed by calendars.

New pattern mining algorithms are developed to process multiple sequential, conditional sequential, and concurrent patterns very fast with only one pass through event steams (Chapter6). Also a comprehensive GUI is developed to facilitate interaction of a domain expert with the system. The framework is evaluated for two applications: objective self (Chapter 7) and asthma risk management (Chapter 8).

Correspondingly, there are multiple opportunities for improvement and future work.

- **User Interface and User Experience**

  In the current GUI of the framework, the result of pattern mining is shown as co-occurrence matrix visualization. The interpretation of the matrix is not straightforward. More interactive visualizations can be added not only to display results of mining operators but also to display results of pattern queries. Also, the current GUI is primarily based on icon clicking for choosing proper operators and formulating a hypothesis. In future, proper wrappers can be developed to convert textual pattern descriptions to their corresponding automata for processing.

- **Distributed Processing**

  In the system design aspect, so far we assumed that the amount of input data will not

overwhelm the processing capacity of the framework. So the operators are implemented in a single processing machine. The patterns that are formulated using the high-level pattern language are translated into their corresponding processing automata. With the amount of temporal data generated from different sensors and devices, data can potentially become so large that it goes beyond the system processing limits. In future, processing automata can be implemented in different machines. This allows detection to be distributed throughout multiple system.

- **Additional Operators**

  Currently the framework supports four operators for building sequential, conditional sequential, concurrent, and alternation patterns. The first operator that can be added for a comprehensive pattern formulation is *negation* operator. The fact that specific events are not allowed to occur in a given interval, might sometimes be necessary in pattern formulation and query.

- **Additional Applications**

  As explained in chapter 2, by combining top-down and bottom-up analysis in a single framework, complex applications especially in healthcare domain can benefit from engaging an expert in the analysis process. In future, the framework need to be tested for multiple applications in helathcare domain to improve the functionalities that are needed in this domain.

- **Cloud Storage**

  In the system design aspect, so far we assumed that data is stored on a local storage. However, given that the temporal data streams are now being generated by massive amounts of devices, the overall data could potentially be so large that it goes beyond the storage capacities of a local machine. In future, cloud storage services such as Amazon Simple Storage Services (S3) can be utilized to manage the the ever growing volume of data.

Bibliography

[1] Challenges of deep learning howpublished = https://www.coursera.org/learn/ml-foundations/lecture/e8hqz/challenges-of-deep-learning.

[2] Cmsr data miner, data mining & predictive modeling software howpublished = http://www.roselladb.com/starprobe.htm.

[3] A data model and format for collecting and distributing event information. https://iptc.org/standards/eventsml-g2/.

[4] Elki: Environment for developing kdd-applications supported by index-structures howpublished = http://elki.dbs.ifi.lmu.de/.

[5] Unstructured information management application, the apache software foundation howpublished = https://uima.apache.org.

[6] J. Abello, F. Van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):669–676, 2006.

[7] R. Agrawal, C. Faloutsos, and A. Swami. *Efficient similarity search in sequence databases*. Springer, 1993.

[8] R. Agrawal and R. Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[9] W. Aigner and S. Miksch. Carevis: integrated visualization of computerized protocols and temporal patient data. *Artificial intelligence in medicine*, 37(3):203–218, 2006.

[10] C. B. Akgül, D. L. Rubin, S. Napel, C. F. Beaulieu, H. Greenspan, and B. Acar. Content-based image retrieval in radiology: current status and future directions. *Journal of Digital Imaging*, 24(2):208–222, 2011.

[11] J. Alcala-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, et al. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[12] J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *Proceedings of the 2000 ACM symposium on Applied computing-Volume 1*, pages 294–300. ACM, 2000.

[13] J. F. Allen. An interval-based representation of temporal knowledge. In *IJCAI*, volume 81, pages 221–226, 1981.

[14] C. M. Antunes and A. L. Oliveira. Temporal data mining: An overview. In *KDD workshop on temporal data mining*, volume 1, page 13, 2001.

[15] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435. ACM, 2002.

[16] W. D. Bae, S. Alkobaisi, S. Narayanappa, and C. C. Liu. A mobile data analysis framework for environmental health decision support. In *Information Technology: New Generations (ITNG), 2012 Ninth International Conference on*, pages 155–161. IEEE, 2012.

[17] P. Bak, I. Omer, and T. Schreck. *Visual analytics of urban environments using high-resolution geographic data*. Springer, 2010.

[18] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.

[19] R. Bellazzi, R. Guglielmann, L. Ironi, and C. Patrini. A hybrid input-output approach to model metabolic systems: An application to intracellular thiamine kinetics. *Journal of Biomedical Informatics*, 34(4):221–248, 2001.

[20] G. Biegel and V. Cahill. A framework for developing mobile, context-aware applications. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 361–365. IEEE, 2004.

[21] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[22] G. Birkhoff, G. Birkhoff, G. Birkhoff, and G. Birkhoff. *Lattice theory*, volume 25. American Mathematical Society New York, 1948.

[23] I. Bowman, S. H. Joshi, and J. Van Horn. Visual systems for interactive exploration and mining of large-scale neuroimaging data archives. *Frontiers in neuroinformatics*, 6(11):1143–1150, 2012.

[24] B. Bredeweg and K. D. Forbus. Qualitative modeling in education. *AI magazine*, 24(4):35, 2003.

[25] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[26] S. Cakmak, R. E. Dales, and F. Coates. Does air pollution increase the effect of aeroallergens on hospitalization for asthma? *Journal of Allergy and Clinical Immunology*, 129(1):228–231, 2012.

[27] G. Casas-Garriga. *Discovering unbounded episodes in sequential data*. Springer, 2003.

[28] Z. Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information processing letters*, 40(5):269–276, 1991.

[29] X. Chen, I. Petrounias, and H. Heathfield. Discovering temporal association rules in temporal databases. In *IADT*, pages 312–319, 1998.

[30] D.-Y. Chiu, Y.-H. Wu, and A. L. Chen. An efficient algorithm for mining frequent sequences by a new strategy without support counting. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 375–386. IEEE, 2004.

[31] P. R. Cohen. Fluent learning: Elucidating the structure of episodes. In *Advances in Intelligent Data Analysis*, pages 268–277. Springer, 2001.

[32] A. A. Cruz, J. Bousquet, and N. Khaltaev. *Global surveillance, prevention and control of chronic respiratory diseases: a comprehensive approach*. World Health Organization, 2007.

[33] A. Cuzzocrea and M. M. Gaber. Data science and distributed intelligence: Recent developments and future insights. In *Intelligent Distributed Computing VI*, pages 139–147. Springer, 2013.

[34] G. D'Amato, G. Liccardi, M. D'amato, and M. Cazzola. Outdoor air pollution, climatic changes and allergic bronchial asthma. *European Respiratory Journal*, 20(3):763–776, 2002.

[35] J. de Kleer. Qualitative and quantitative knowledge in classical mechanics. 1975.

[36] J. de Kleer and J. Brown. A qualitative physics based on confluences. *Artificial Intelligence - Special volume on qualitative reasoning about physical systems*, 24(1-3):7–84, 1984.

[37] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction*, 16(2):97–166, Dec. 2001.

[38] M. H. Dunham. *Data mining: Introductory and advanced topics*. Pearson Education India, 2006.

[39] S. Dzeroski and L. Todorovski. Discovering dynamics: from inductive logic programming to machine discovery. *Journal of Intelligent Information Systems*, 4(1):89–108, 1995.

[40] A. Endert, P. Fiaux, and C. North. Semantic interaction for sensemaking: inferring analytical reasoning for model steering. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2879–2888, 2012.

[41] J. A. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, pages 167–174. IEEE, 2006.

[42] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[43] D. Ferreira, V. Kostakos, and A. K. Dey. AWARE: mobile context instrumentation framework. *Frontiers in ICT*, 2(6), 2015.

[44] K. Forbus. Qualitative process theory. *Artificial Intelligence - Special volume on qualitative reasoning about physical systems*, 24(1-3):85–168, 1984.

[45] K. D. Forbus. Qualitative reasoning., 1997.

[46] C. Freksa. Temporal reasoning based on semi-intervals. *Artificial intelligence*, 54(1):199–227, 1992.

[47] I. Galan, A. Tobias, J. Banegas, and E. Aranguez. Short-term effects of air pollution on daily asthma emergency room admissions. *European Respiratory Journal*, 22(5):802–808, 2003.

[48] B. Ganter. *Two basic algorithms in concept analysis*. Springer, 2010.

[49] B. Ganter and R. Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[50] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. Mylifebits: fulfilling the memex vision. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 235–238. ACM, 2002.

[51] S. Ghosh. Challenges in deep learning for multimodal applications. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 611–615. ACM, 2015.

[52] E. Goldstein. *Sensation and perception*. Cengage Learning, 2013.

[53] D. Gotz, F. Wang, and A. Perer. A methodology for interactive mining and visual analysis of clinical event patterns using electronic health record data. *Journal of biomedical informatics*, 48:148–159, 2014.

[54] D. Gotz and K. Wongsuphasawat. Interactive intervention analysis. In *AMIA annual symposium proceedings*, volume 2012, page 274. American Medical Informatics Association, 2012.

[55] C. Gurrin, A. F. Smeaton, and A. R. Doherty. Lifelogging: Personal big data. *Foundations and trends in information retrieval*, 8(1):1–125, 2014.

[56] J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM (JACM)*, 38(4):935–962, 1991.

[57] W. Hamscher, M. Y. Kiang, and R. Lang. Qualitative reasoning in business, finance, and economics: Introduction. *Decision Support Systems*, 15(2):99–103, 1995.

[58] J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 106–115. IEEE, 1999.

[59] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359. ACM, 2000.

[60] D. J. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. MIT press, 2001.

[61] T. Hao, G. Xing, and G. Zhou. isleep: unobtrusive sleep quality monitoring using smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 4. ACM, 2013.

[62] M. Hirao, S. Inenaga, A. Shinohara, M. Takeda, and S. Arikawa. A practical algorithm to find the best episode patterns. In *Discovery science*, pages 435–440. Springer, 2001.

[63] K.-Y. Huang, C.-H. Chang, and K.-Z. Lin. Cocoa: Compressed continuity analysis for temporal databases. *Lecture notes in computer science*, pages 509–511, 2004.

[64] O. D. Incel, M. Kose, and C. Ersoy. A review and taxonomy of activity recognition on mobile phones. *BioNanoScience*, 3(2):145–171, 2013.

[65] M. W. Kadous. *Temporal classification: Extending the classification paradigm to multivariate time series*. PhD thesis, The University of New South Wales, 2002.

[66] Y. Kaku, K. Kuramoto, S. Kobashi, and Y. Hata. Asthmatic attacks prediction considering weather factors based on fuzzy-ar model. In *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pages 1–4. IEEE, 2012.

[67] P.-s. Kam and A. W.-C. Fu. *Discovering temporal patterns for interval-based events*. Springer, 2000.

[68] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, volume 98, pages 239–243, 1998.

[69] A. Ketterlin. Clustering sequences of complex objects. In *KDD*, pages 215–218, 1997.

[70] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[71] J. Q. Koenig. Air pollution and asthma. *Journal of allergy and clinical immunology*, 104(4):717–722, 1999.

[72] J. Kölling, D. Langenkämper, S. Abouna, M. Khan, and T. W. Nattkemper. Whidea web tool for visual data mining colocation patterns in multivariate bioimages. *Bioinformatics*, 28(8):1143–1150, 2012.

[73] B. Kuipers. *Qualitative reasoning: modeling and simulation with incomplete knowledge*. MIT press, 1994.

[74] S. O. Kuznetsov and S. A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):189–216, 2002.

[75] S. Laxman, P. Sastry, and K. Unnikrishnan. Discovering frequent episodes and learning hidden markov models: A formal connection. *Knowledge and Data Engineering, IEEE Transactions on*, 17(11):1505–1517, 2005.

[76] C. C. Lee, S. C. Sheridan, and S. Lin. Relating weather types to asthma-related hospital admissions in new york state. *EcoHealth*, 9(4):427–439, 2012.

[77] C.-H. Lee, J. C.-Y. Chen, and V. S. Tseng. A novel data mining mechanism considering bio-signal and environmental data with applications on asthma monitoring. *Computer methods and programs in biomedicine*, 101(1):44–61, 2011.

[78] N. Lesh, M. J. Zaki, and M. Ogihara. Mining features for sequence classification. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 342–346. ACM, 1999.

[79] C.-S. Li, P. S. Yu, and V. Castelli. Hierarchyscan: A hierarchical similarity search algorithm for databases of long sequences. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, pages 546–553. IEEE, 1996.

[80] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *The International Journal of Robotics Research*, 26(1):119–134, 2007.

[81] F. Lin. Embracing causality in specifying the indeterminate effects of actions. In *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 1*, pages 670–676. AAAI Press, 1996.

[82] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[83] M.-Y. Lin and S.-Y. Lee. Improving the efficiency of interactive sequential pattern mining by incremental pattern discovery. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 8–pp. IEEE, 2003.

[84] W. Lin, M. A. Orgun, and G. J. Williams. Temporal data mining using hidden markov-local polynomial models. In *Advances in Knowledge Discovery and Data Mining*, pages 324–335. Springer, 2001.

[85] D. C. Logan. Known knowns, known unknowns, unknown unknowns and the propagation of scientific enquiry. *Journal of experimental botany*, 60(3):712–714, 2009.

[86] S. Ma and J. L. Hellerstein. Mining partially periodic event patterns with unknown periods. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 205–214. IEEE, 2001.

[87] M. S. Magnusson. Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods, Instruments, & Computers*, 32(1):93–110, 2000.

[88] O. Maimon and L. Rokach. Introduction to knowledge discovery in databases. In *Data Mining and Knowledge Discovery Handbook*, pages 1–17. Springer, 2005.

[89] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[90] N. McCain, H. Turner, et al. Causal theories of action and change. In *AAAI/IAAI*, pages 460–465. Citeseer, 1997.

[91] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Readings in artificial intelligence*, pages 431–450, 1969.

[92] N. Mireku, Y. Wang, J. Ager, R. C. Reddy, and A. P. Baptist. Changes in weather and the effects on pediatric asthma exacerbations. *Annals of Allergy, Asthma & Immunology*, 103(3):220–224, 2009.

[93] Y. Mizunuma, S. Yamamoto, Y. Yamaguchi, A. Ikeuchi, T. Satoh, and S. Shimada. Twitter bursts: Analysis of their occurrences and classifications. In *ICDS 2014, The Eighth International Conference on Digital Society*, pages 182–187, 2014.

[94] F. Moerchen. Algorithms for time series knowledge mining. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 668–673. ACM, 2006.

[95] F. Moerchen and D. Fradkin. Robust mining of time intervals with semi-interval partial order patterns. In *SDM*, pages 315–326, 2010.

[96] C. Mooney and J. F. Roddick. Mining relationships between interacting episodes. In *SDM*, pages 1–10. SIAM, 2004.

[97] F. Mörchen and A. Ultsch. Efficient mining of understandable patterns from multivariate interval time series. *Data Mining and Knowledge Discovery*, 15(2):181–215, 2007.

[98] K. Morik. The representation racepreprocessing for handling time phenomena. In *Machine Learning: ECML 2000*, pages 4–19. Springer, 2000.

[99] T. Morzy, M. Wojciechowski, and M. Zakrzewicz. Efficient constraint-based sequential pattern mining using dataset filtering techniques. In *Databases and Information Systems II*, pages 297–309. Springer, 2002.

[100] B. C. Moszkowski. Reasoning about digital circuits. Technical report, DTIC Document, 1983.

[101] E. T. Mueller. Event calculus reasoning through satisfiability. *Journal of Logic and Computation*, 14(5):703–730, 2004.

[102] E. T. Mueller. A tool for satisfiability-based commonsense reasoning in the event calculus. In *FLAIRS Conference*, volume 4, 2004.

[103] E. T. Mueller and G. Sutcliffe. Reasoning in the event calculus using first-order automated theorem proving. In *FLAIRS Conference*, pages 840–841, 2005.

[104] N. J. Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

[105] L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information processing letters*, 71(5):199–204, 1999.

[106] D. E. O'Leary. Artificial intelligence and big data. *IEEE Intelligent Systems*, (2):96–99, 2013.

[107] G. D. Oosthuizen. The use of a lattice in knowledge processing. 1992.

[108] B. Özden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Data Engineering, 1998. Proceedings., 14th International Conference on*, pages 412–421. IEEE, 1998.

[109] C. Pastrello, E. Pasini, M. Kotlyar, D. Otasek, S. Wong, W. Sangrar, S. Rahmati, and I. Jurisica. Integration, visualization and analysis of human interactome. *Biochemical and biophysical research communications*, 445(4):757–773, 2014.

[110] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *icccn*, page 0215. IEEE, 2001.

[111] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. Mining access patterns efficiently from web logs. In *Knowledge discovery and data mining. Current issues and new applications*, pages 396–407. Springer, 2000.

[112] J. Pei, J. Han, and W. Wang. Mining sequential patterns with constraints in large databases. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 18–25. ACM, 2002.

[113] G. R. Peterson. Demarcation and the scientistic fallacy. *Zygon®*, 38(4):751–761, 2003.

[114] C. Plaisant, R. Mushlin, A. Snyder, J. Li, D. Heller, and B. Shneiderman. Lifelines: using visualization to enhance navigation and analysis of patient records. In *Proceedings of the AMIA Symposium*, page 76. American Medical Informatics Association, 1998.

[115] R. A. G. Psaila and E. L. Wimmers Mohamed &It. Querying shapes of histories. 1995.

[116] Z. Qiu, C. Gurrin, and A. F. Smeaton. Evaluating access mechanisms for multimodal representations of lifelogs. In *MultiMedia Modeling*, pages 574–585. Springer, 2016.

[117] Y. Qu, C. Wang, and X. S. Wang. Supporting fast search in time series for movement patterns in multiple scales. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 251–258. ACM, 1998.

[118] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[119] S. Ramachandran, R. J. Mooney, and B. J. Kuipers. Learning qualitative models for systems with multiple operating regions. In *Proceedings of the 8th International Workshop on Qualitative Reasoning about Physical Systems*, pages 212–223, 1994.

[120] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, 27:359–380, 1991.

[121] A. Russell and B. Brunekreef. A focus on particulate matter and health. *Environmental science & technology*, 43(13):4620–4625, 2009.

[122] H. Sayyadi and L. Raschid. A graph analytical approach for topic detection. *ACM Transactions on Internet Technology (TOIT)*, 13(2):4, 2013.

[123] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. Pybrain. *The Journal of Machine Learning Research*, 11:743–746, 2010.

[124] A. Scherp and V. Mezaris. Survey on modeling and indexing events in multimedia. *Multimedia Tools and Applications*, 70(1):7–23, 2014.

[125] M. Shanahan. An abductive event calculus planner. *The Journal of Logic Programming*, 44(1):207–240, 2000.

[126] M. Shanahan and M. Witkowski. Event calculus planning through satisfiability. *Journal of Logic and Computation*, 14(5):731–745, 2004.

[127] P. Struss and C. Price. Model-based systems in the automotive industry. *AI magazine*, 24(4):17, 2003.

[128] D. Šuc and I. Bratko. Induction of qualitative trees. In *ECML*, volume 2167, pages 442–453. Springer, 2001.

[129] D. Šuc, D. Vladušič, and I. Bratko. Qualitatively faithful quantitative prediction. *Artificial Intelligence*, 158(2):189–214, 2004.

[130] G. Sutcliffe and C. Suttner. The tptp problem library for automated theorem proving. *Web site at: http://www. cs. miami. edu/˜ tptp*, 2004.

[131] M. Thielscher. Ramification and causality. *Artificial intelligence*, 89(1):317–364, 1997.

[132] C. Tominski, J. Abello, and H. Schumann. Cgvan interactive graph visualization system. *Computers & Graphics*, 33(6):660–678, 2009.

[133] L. Travé-Massuyès, L. Ironi, and P. Dague. Mathematical foundations of qualitative reasoning. *AI magazine*, 24(4):91–106, 2003.

[134] Z. Troníček. Episode matching. In *Combinatorial pattern matching*, pages 143–146. Springer, 2001.

[135] A. K. Tung, H. Lu, J. Han, and L. Feng. Breaking the barrier of transactions: Mining inter-transaction association rules. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 297–301. ACM, 1999.

[136] P. Tzvetkov, X. Yan, and J. Han. Tsp: Mining top-k closed sequential patterns. *Knowledge and Information Systems*, 7(4):438–457, 2005.

[137] P. Valtchev, R. Missaoui, and P. Lebrun. A partition-based approach towards constructing galois (concept) lattices. *Discrete Mathematics*, 256(3):801–829, 2002.

[138] P. S. V.M. Hudson and R. Whitmer. A new kind of social science?: Moving ahead with reverse wolfram models applied to event data. In *Proceedings of the 46th annual International Studies Association Convention, Honolulu, Hawaii*, 2005.

[139] G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *KDD*, pages 359–363, 1998.

[140] M. P. Wellman. Qualitative simulation with multivariate constraints. In *2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 547–557, 1991.

[141] J. Wilson. Quantified self: Bring science into everyday life, one measurement at a time. 2014.

[142] M. Wojciechowski. Interactive constraint-based sequential pattern mining. In *Advances in Databases and Information Systems*, pages 169–181. Springer, 2001.

[143] K. Wongsuphasawat and D. Gotz. Outflow: Visualizing patient flow by symptoms and outcome. In *IEEE VisWeek Workshop on Visual Analytics in Healthcare, Providence, Rhode Island, USA*, pages 25–28. American Medical Informatics Association, 2011.

[144] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2659–2668, 2012.

[145] K. Wongsuphasawat and B. Shneiderman. Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pages 27–34. IEEE, 2009.

[146] S.-Y. Wu and Y.-L. Chen. Mining nonambiguous temporal patterns for interval-based events. *Knowledge and Data Engineering, IEEE Transactions on*, 19(6):742–758, 2007.

[147] X. Yan, J. Han, and R. Afshar. Clospan: Mining closed sequential patterns in large datasets. In *In SDM*, pages 166–177. SIAM, 2003.

[148] C.-C. Yu and Y.-L. Chen. Mining sequential patterns from multidimensional sequence data. *Knowledge and Data Engineering, IEEE Transactions on*, 17(1):136–140, 2005.

[149] J. Žabkar, I. Bratko, and J. Demšar. *Learning qualitative models through partial derivatives by Padé*. 2007.

[150] J. Žabkar, R. Žabkar, D. Vladušič, D. Čemas, D. Šuc, and I. Bratko. Q2 prediction of ozone concentrations. *Ecological modelling*, 191(1):68–82, 2006.