Extracting Knowledge from Raw IoT Data Streams

Adnan Akbar

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Institute for Communication Systems
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

December 2017

Summary

As sensors are adopted in almost every field of life, the Internet of Things (IoT) is triggering a massive influx of data. This large amount of data is of little value until it is processed intelligently to extract high-level knowledge which can be used to make decisions. The process of knowledge extraction from data streams is complex predominantly due to heterogeneous data sources, unreliable networks and real-time processing requirements. Different recent studies have showed that solutions based on complex event processing (CEP) have the potential to extract high-level knowledge from these data streams. However, the use of CEP for IoT applications is still in early phase and faces many challenges.

First, CEP applications are intended to provide reactive solutions by correlating data streams using predefined rules as the events happen. As the notion of many IoT applications is changing from reactive to proactive where complex events can be predicted before they actually happen, solutions based on CEP required an extension to address this issue. To this end, this work proposes a proactive method based on CEP and machine learning (ML) where historical data is exploited using ML part and combined with real-time flow of CEP to provide the basis for predictive event processing. Second, systems based on CEP deploy static rules and there is no means to update the rules according to the current context automatically. In order to address this issue, this thesis proposed a novel method based on ML to find CEP rules automatically and update them according to the current context.

Third, in state-of-the-art CEP systems, events are correlated using absolute rules where a complex event detected is either true or false. Given the sporadic nature of IoT, missing and uncertain data is a common phenomenon and CEP systems of today are unable to take this inherent uncertainty of real-world events into account while taking decisions. This thesis addressed this issue by proposing a probabilistic event processing approach by extending state-of-the-art CEP by combining it with Bayesian networks (BNs). Finally, the size and complexity of the IoT data presents a generic challenge which is being addressed throughout in this thesis.

The above mentioned contributions were evaluated using real-world data collected from heterogeneous sources to prove the accuracy and reliability of the proposed methods. The feasibility and applicability of the proposed solutions were demonstrated by implementing it for real-world applications. The work presented in this thesis significantly improves state-of-the-art methods and provides a fundamental building block towards extracting knowledge from raw IoT data streams.

Key words: Big data, Complex event processing, Context-aware, Internet of Things, Machine learning, Predictive analytics.

Email: adnan.akbar@surrey.ac.uk; adnan.akbar@outlook.com
WWW: http://www.epubs.surrey.ac.uk/

Acknowledgements

Foremost, I would like to express my deepest gratitude to my supervisors Professor Klaus Moessner and Dr. Francois Carrez for their guidance and continuous support of my PhD study and research. Their extensive expertise, as well as enthusiasm, passion, devotion and optimism have played a pivotal role in shaping the direction of my research.

I would also like to give my appreciation to my colleagues and friends, Waqas Rafique, Sameed Husain, Saad Anjum, Shahzaib Nabi, Hasan Malik, Junaid Mir, Atif Mir, Junaid Ali and Adnan Zafar for their constant support and help during the duration of my PhD.

Last but not the least, I would also like to acknowledge the unconditional love and support of my parents Dr Akbar Saeed and Rizwana Akbar, my wife Saleha Sarfraz and rest of the family which they have shown on me during the course of my PhD.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Sensors are by no means a new phenomenon: the first thermostat was invented in the 19th century and space exploration would not have been possible without them. What is revolutionary today about the Internet of Things (IoT) lies in connecting sensors to the internet followed by their recent adoption on an unprecedented scale, fuelled by several economic factors such as dramatic drop in the cost of sensors, network bandwidth and processing power. Moreover, unlike the Internet (of humans), the IoT allows data to be captured and ingested autonomously, avoiding the human data entry bottleneck. IoT data will arguably become the biggest big data (as projected by Gartner [1]), possibly overtaking enterprise, media and entertainment data.

The characteristics of the IoT data differ from conventional data sources in many ways. IoT networks are formed by heterogeneous devices including different sensors, smart phones and everyday connected objects. The type and structure of data transmitted by each device is different, making IoT data heterogeneous in a true manner. Most of these networks are deployed outdoor and are connected using less reliable wireless links in order to save energy. As a result, data provided by these devices can be sporadic, less reliable and even incomplete.

[1] http://www.gartner.com/newsroom/id/3598917

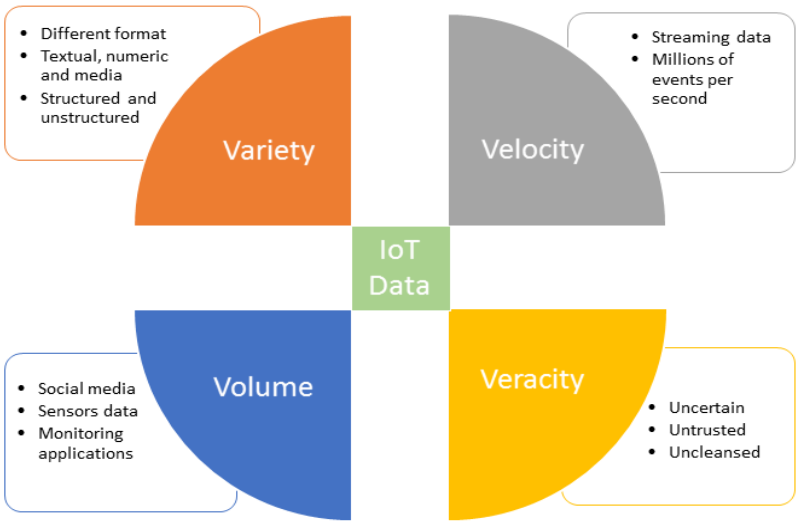


Figure 1.1: Characteristics of IoT data [75]

Continuous data streams are ubiquitous in IoT. Supply chain, health care, energy management and transportation are among the few examples. Timely analysis of these streams can not only turn into profit (supply chain), but also result in increased citizen's comfort (transportation), cost reduction (energy management) and can even save lives (healthcare). Furthermore, often the data volume is so high that it cannot be stored on disk or sent over slow network links before being processed. Figure 1.1 highlights the characteristics and challenges of the IoT data in the form of famous 4 V's of big data [75].

These applications require methods which are capable of interpreting patterns, applying them to current situations and taking accurate decisions with minimal time latency. Data is in the form of real-time events requiring a paradigm shift in the methodology for analysing and inferring high-level knowledge. In order to extract high-level knowledge from such streaming data in real-time, an event driven architecture (EDA) [22] called complex event processing (CEP) has been proposed in recent years. The research area of CEP includes processing, analysing and correlating event streams from different data sources using distributed message-based systems to extract high-level or actionable knowledge in near real-time [59].

The core of the CEP is a rule-based engine which can extract causal and temporal

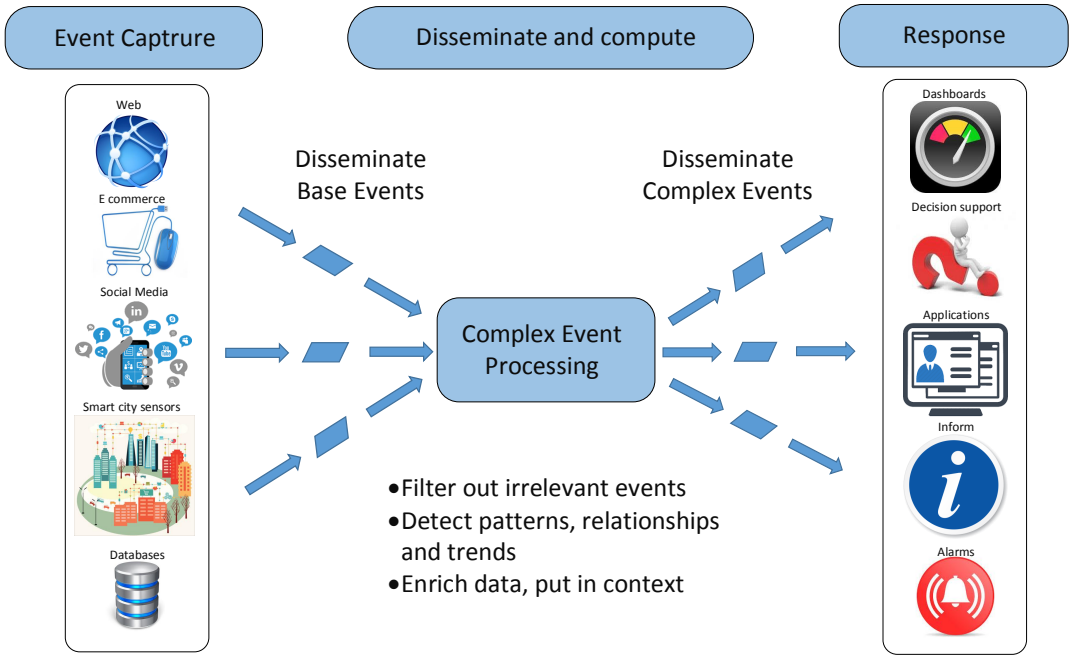


Figure 1.2: Complex event processing for real-world applications

patterns using pre-defined rules. The inherent distributed and scalable nature of CEP [16] fulfils many requirements for real-time IoT applications as evident by numerous examples; for instance analysing real-time traffic data for intelligent transportation system (ITS) [98] or providing automatic managing systems for smart buildings [12] [84]. Figure 1.2 shows some examples of real-world applications of CEP where data is generated in the form of events from heterogeneous sources and CEP provides the capability to analyse and detect complex events from these raw events in real-time. These complex events can be disseminated to other applications for either decision making or serve as an input for other systems.

1.2 Definitions

There are several terms used in this thesis which have different meanings in different contexts. In order to have a common understanding, these terms are defined here in the context it has been used in this thesis;

Knowledge: Any high-level event which is extracted from raw IoT data with the

potential to contribute towards making decisions is treated as knowledge in this thesis. For example a hot weather extracted from temperature sensor readings represents knowledge. Similarly, bad traffic extracted from traffic sensors is another example of knowledge.

latency: Latency is defined as the amount of time, data takes from being ingested into the system to being analysed to extract knowledge from it.

Large-scale/Big data: Large-scale data or Big data is referred in this thesis as the data sets which are voluminous and complex enough that it cannot be analysed using conventional methods.

Complexity: Complexity in this thesis is used in two different contexts; first, the complexity of data being produced and second, the complexity of the algorithm. The former relates to the heterogeneity of IoT data where different IoT data streams being generated at different rates need to be analysed and correlated whereas the latter is referred to an algorithm which is unable to analyse the input data in near real-time.

Proactive: Proactive is referred as looking into future where actions are taken in order to avoid the situation at first place.

Heterogeneous event processing: When data from different type of sensors or devices are combined to extract a complex event, it is referred as heterogeneous event processing.

Scalability: Scalability is referred in this thesis as a solution which can be made scalable to address the requirements of large-scale applications. It should be noted that this thesis addresses scalability in terms of horizontal scaling where processing requirements are distributed across different machines rather than using a single machine with high processing power.

1.3 Challenges

Although CEP provides solutions to analyse complex data streams in real-time, it was not initially designed for IoT applications. CEP finds its background in financial applications where the major focus is on processing high velocity data with minimal time latency. The real-time processing capability of CEP is a main factor which encouraged researches to explore it for IoT applications. However, the characteristics of IoT data pose different set of challenges, requiring extension of current CEP technology.

Figure 1.3: Challenges for the use of CEP for IoT applications

There are several challenges which CEP faces for its successful application to extract knowledge from IoT data streams [24]. Figure 1.3 presents an overview of these challenges. There are few challenges which are related to the development of technology such as the processing power of CEP and the development of generic query language for writing CEP rules. The former involves increasing the processing power of CEP to analyse the ever increasing IoT data in real-time whereas the latter challenge is towards the development of generic query language so that CEP can be used by wider community in IoT for extracting knowledge. These challenges do not lie in the scope of this

thesis as the focus of this thesis is on the application of existing CEP technology for extracting high-level knowledge using methods from machine learning and probabilistic theory. A summary of identified challenges which are addressed in this thesis is below;

1) CEP has the potential to analyse data streams in real-time, however it lacks the predictive power provided by machine learning (ML) and statistical data analysis methods [30]. Most of the CEP applications are intended to provide reactive solutions by correlating data streams using predefined rules as the events occur, and do not exploit historical data [59]. But in many applications, early prediction of an event is more useful than detecting it after it has already occurred. For example, it is more useful to predict the traffic congestion as compared to detecting it when it has already occurred. Such information can help to take preventive actions by traffic administrators to avoid a congestion in the first place. The advantage of predicting an event is more obvious if we imagine the gain of predicting Tsunami disaster as compared to detecting it after it has happened. TIBCO CEO Vivek Ranadive explained the importance of predicting events in his book "The Two-Second Advantage" [66] by stressing that a little information about the event before it happens is more valuable as compared to all the information after the event has happened.

2) In order to detect complex events, systems based on CEP require rules which have to be defined manually by the system administrators[49]. Based on this, there is an assumption that administrators have the required background knowledge which sometimes is neither available nor so precise. The manual setting of rules and patterns limits the use of CEP only to domain experts and poses a major drawback. And even though with prior expertise and knowledge, experts are prone to make errors in choosing optimized parameters for dynamic systems. Systems based on CEP deploy static rules and there is no means to update the rules automatically. In real-time dynamic IoT applications, the context of the application is always changing and the performance of CEP will deteriorate in such scenarios.

3) In state-of-the-art CEP systems, events are correlated using absolute rules where

a complex event detected is either true or false [2] [86]. Even if a single event or condition is missing, a complex event will not be generated. Given the sporadic nature of IoT, missing and uncertain data is a common phenomenon and CEP systems of today are unable to take this inherent uncertainty of real-world events into account while taking decisions. This represents a major drawback considering the random and probabilistic nature of real-world events.

4) The largeness and the complexity of IoT data represents a generic challenge which should be considered while addressing all of the above mentioned challenges.

1.4 Research Problem

This research work addresses the following research question:

"How can high-level knowledge be extracted from raw IoT data?"

The research focus can be further sub-divided into following sub-problems.

- How can predictive analytics be applied to real-time data streams in IoT for predicting complex events with minimum time latency?
- How to provide adaptive context-aware methods for extracting complex events from real-time streaming data in IoT?
- How to provide fast and scalable data analysis solutions for extracting high-level knowledge from large-scale IoT data?
- How to handle uncertainty associated with real-world events in order to provide a reliable solution for complex IoT applications?

1.5 Research Methodology

The methodology for this research consist of several steps as shown in the Figure 1.4. The first step involves identifying and formulating the research problem. After formulating the research problem, background study was carried on where complex event

Figure 1.4: Research Methodology

processing (CEP) was identified as the potential solution for extracting knowledge from IoT data streams. CEP was chosen because of its ability to extract temporal, causal and spatial patterns from heterogeneous data streams in near real-time. Literature was reviewed on the existing solutions based on CEP to identify four major research challenges in this direction. In order to demonstrate the application of this research, a

use-case from intelligent transportation system (ITS) was chosen.

There were several reasons for choosing ITS use-case scenario for this research. First, it provides a big data problem in true meaning due to the amount of data being generated at city level; second, the availability of traffic data from Madrid city council constitutes an important factor and third, as ITS has the direct impact on the social and economical development of smart cities, it provides the perfect use-case for the demonstration of this research.

In the context of identified research challenges, three different methods were proposed, implemented and evaluated using the ITS use-case. After evaluation of proposed solutions, an architecture was developed for large-scale applications based on it to address the fourth challenge of managing the big data. Finally, the conclusions were drawn from this research and potential future directions were highlighted.

1.6 Research Contributions

The main objective of this research work is to propose and validate novel methods for extracting high-level knowledge from raw IoT data. The complexity, heterogeneity, largeness and real-time processing requirements of IoT data make it a challenging task. The complex process of extracting high-level knowledge from raw IoT data is divided into following three main contributions. The three contributions directly mapped against the first three challenges whereas the fourth challenge is generic and addressed in all contributions.

1) Predictive Analytics for Complex IoT data streams

As a first contribution, this thesis proposes a novel architecture which exploits historical data using machine learning (ML) in conjunction with CEP in order to provide proactive solutions for IoT applications. An adaptive prediction algorithm called adaptive moving window regression (AMWR) is proposed for dynamic IoT data and evaluated using a real-world use case with an accuracy ranging from 87 -96 %. The proposed algorithm can perform accurate predictions in near real-time due to reduced complexity

and can work along CEP in the proposed architecture. Furthermore, this work implemented the proposed architecture using open source components which are optimized for big data applications and validated on a use-case from intelligent transportation system (ITS). The proposed architecture is reliable and can be used across different fields in order to predict complex events.

Contribution 1 can be divided into following sub-contributions:

- Proposed and implemented a generic framework based on open source components for combining ML with CEP in order to predict complex events for proactive IoT applications;
- Proposed an adaptive prediction algorithm called adaptive moving window regression (AMWR) for dynamic IoT data streams and implemented it on a real-world use-case of ITS achieving accuracy ranging from 87 -96 %. AMWR is based on a novel method for finding optimum size for training window by exploiting spectral components of time series data;
- Modelled the error introduced by the prediction algorithm using a parametric distribution and derived expressions for the overall error of the system, as the error propagates through the CEP.

2) Context-Aware Event processing for distributed IoT applications

The second contribution of this thesis provides context-aware event processing for large-scale IoT applications. In this regard, a novel approach is proposed for finding optimized parameters for CEP rules using machine learning methods which is adaptive with respect to current context. The proposed solution is demonstrated using real-world traffic data and was implemented in a scalable manner for large-scale IoT applications.

Contribution 2 can be divided into following sub-contributions:

- Proposed and implemented a novel method based on machine learning for calculating optimized parameters for CEP rules and update it according to the current context;

- Demonstration of proposed method using real-world use-case of intelligent transportation system (ITS) with an accuracy ranging between 79 - 85 %;
- Proposed the *Hut* architecture for large-scale IoT applications based on the novel method and implemented it using open-source components optimized for big data applications.

3) Handling uncertainty in Event Processing

Final contribution of this thesis addresses the challenge of handling uncertainty inherently present in all real-world events. An hierarchical event processing framework based on two levels of analytics was proposed where first layer provides a generic interface using a service oriented gateway to ingest data from multiple interfaces in IoT systems, store it in a scalable manner and analyse it in real-time to extract high-level events whereas second layer is responsible for probabilistic processing of these high-level events. This work extends state-of-the-art CEP using Bayesian networks (BNs) in order to take uncertainty into account while detecting complex events.

Contribution 3 can be divided into following sub-contributions:

- Proposed a solution based on probabilistic graphical models in order to take uncertainty into account while detecting complex events using CEP. Explored the use of Bayesian networks and extended it with CEP for large-scale systems;
- Validation of proposed solution on a real world scenario of intelligent transportation where multiple data streams including traffic, weather and social media data were used. Furthermore, developed a complete end-to-end solution and demonstrated the application of proposed solution for real-time management of urban traffic control.

1.7 Publications

The research work carried out during the course of this PhD has resulted in the following publications:

Journals

J.1 **A. Akbar**, A. Khan, F. Carrez and K. Moessner, "Predictive Analytics for Complex IoT Data Streams," *IEEE IoT Journal,* PP(99):1-1, 2017. [In Press, Accepted on 27 May, 2017, impact factor = 7.6]

J.2 P. Ta-Shma, **A. Akbar**, G. Gerson-Golan, G. Hadash, F. Carrez and K. Moessner "An Ingestion and Analytics Architecture for IoT applied to Smart City Use Cases," *IEEE IoT Journal,* PP(99):1-1, 2017. [In Press, Accepted on 22 June, 2017, impact factor = 7.6]

J.3 G. Kousiouris, **A. Akbar**, J. Sancho, P. Ta-Shma, A. Psychas, D. Kyriazis and T. Varvarigou "An Integrated Information Lifecycle Management Framework for Exploiting Social Network Data to identify Dynamic Large Crowd Concentration Events in Smart Cities Applications," *Future Generation Computer Systems Journal, 2017.* [In Press, Accepted on 10 July, 2017, impact factor = 3.9]

J.4 **A. Akbar**, G. Kousiouris, J. Sancho, P. Ta-Shma, F. Carrez and K. Moessner "Real-time Probabilistic Data Fusion for Large-Scale IoT Applications," *IEEE Access* (under review)

J.5 **A. Akbar**, W. Rafique, F. Carrez and K. Moessner "Complex Event Processing for Internet of Things - An overview and Survey," *Sensors Journal* (under review)

Conferences

C.1 **A. Akbar**, M. Nati, F. Carrez and K. Moessner, "Contextual occupancy detection for smart office by pattern recognition of electricty consumption data," in *2015 IEEE Internation Conference on Communications (ICC),* pages 561-566, June 2015.

C.2 **A. Akbar**, F. Carrez, K. Moessner and A. Zoha "Predicting complex events for pro-active iot applications," in *2015 IEEE World Forum on Internet of Things (WF-IoT),* pages 327-332, Dec 2015.

C.3 **A. Akbar**, F. Carrez, K. Moessner, J. Sancho and J. Rico "Context-aware stream processing for distributed iot applications," in *2015 IEEE World Forum on Internet of Things (WF-IoT),* pages 663-668, Dec 2015.

1.8 Structure of the thesis

The rest of this thesis is organised as follows.

Chapter 2: Literature Review - The chapter starts with the brief introduction of the use of CEP for IoT data analytics followed by the background description of CEP technology and how it evolved to its current form. It then discusses the different concepts of CEP technology and how it fits to IoT world. Furthermore, it describes the use of CEP as a potential solution for real-time analytics for different IoT applications. Finally, it highlights the major challenges for the application of the CEP for IoT data streams and discusses potential future research directions.

Chapter 3: Overall System Framework - This chapter presents an overview of the system framework in the context of major functionalities which were developed in this research. A high-level view of the contributions is also explained in this section.

Chapter 4: Predicting Complex Events - This chapter explains this thesis's approach towards predicting complex events using ML and CEP. A novel time series prediction algorithm called adaptive moving window regression (AMWR) is proposed for dynamic IoT data streams and its performance is evaluated using real-world traffic dataset provided by city of Madrid council. A qualitative comparison with other state-of-the-art regression algorithms is provided where AMWR outperforms other algorithms for different scenarios. Furthermore, an hybrid solution is proposed based on the prediction algorithm and CEP where output of the prediction algorithm is fed as the input to CEP engine and implemented the proposed solution using different open source components. Moreover, error introduced by the prediction algorithm is modelled using a parametric

distribution and expressions for the overall error of the system were derived, as the error propagates through the CEP.

Chapter 5: Context-aware Event Processing - This chapter presents our work towards context-aware event processing for large-scale IoT applications. A novel method is proposed for event processing based on adaptive clustering and CEP which updates according to the current context. The performance of the proposed method is validated using traffic data provided by city of Madrid[1]. An ingestion and analytics architecture for large-scale IoT applications is proposed by extending the novel method. The proposed architecture provides a generic interface using a service oriented gateway to ingest data from multiple interfaces and IoT systems, store it in a scalable manner and analyse it in real-time to extract high-level events with respect to current context. Finally, the chapter is concluded by highlighting how the proposed approach solves the problem of context-aware event processing and provides an edge on existing technologies.

Chapter 6: Handling uncertainty in Event Processing - This chapter explains the proposed solution for handling uncertainty while detecting complex events using CEP. An hierarchical event processing framework is proposed based on two levels of analytics where layer 1 provides a generic interface using a service oriented gateway to ingest data from multiple interfaces and IoT systems, store it in a scalable manner and analyse it in real-time to extract high-level events whereas layer 2 is responsible for probabilistic processing of these high-level events using Bayesian inference in order to take inherent uncertainty of events into account. A description is provided on the use of the *Hut* architecture from previous contribution to ingest and analyse heterogeneous data streams including twitter and weather followed by the description of probabilistic event processing. Furthermore, a novel method based on the integration of Bayesian network with CEP is explained. Finally, the chapter is concluded by highlighting how the proposed approach solves the problem of probabilistic event processing and provides an edge on existing technologies.

[1] http://informo.munimadrid.es/informo/tmadrid/pm.xml

Chapter 7: Future Work - This chapter summarises the major results achieved in this thesis and highlights important lessons learned from the research. It also discusses possible research directions for the future work.

Chapter 2

Literature Review

Complex event processing (CEP) is an emerging technology with its concepts originating from interdisciplinary research. In this regard, it is important to describe its background and components before highlighting the research challenges associated with the CEP. This chapter starts with a brief description of technologies that lead to the development of the CEP in section 2.1, followed by the explanation of different components and concepts related to the CEP in section 2.2. Section 2.3 describes the use of CEP for IoT applications with the help of different examples found in literature. Section 2.4 discusses the future research direction of the CEP for IoT applications and highlights the major challenges in this direction. Finally, the chapter is concluded by explaining the contributions of this thesis in the context of highlighted research challenges in section 2.5.

2.1 Background

Conceptualization of complex events is interdisciplinary research and find its roots in a wide range of systems operating under real-time constraints. The development of the CEP can be credited to following four technologies ;

1) Event-Driven Simulation

2) Networks

3) Active Databases

4) Middleware

2.1.1 Event Driven Simulation

Event-driven simulation represents one of the earliest exploitation of events and use of event-driven computing. Generally, it refers to the use of events to drive the steps in computing with models to predict the behaviour of systems; where the system can be anything from a simple design of traffic lights controller to a more complex model of autonomous cars.

The development of event-driven simulation started in 1950s with the invention of computers. People started to use computers in order to model the design of their devices and predict their behaviour before actual manufacturing. They used computer programs to mimic the behaviour of actual systems and record their response with different input events. By the 1960s, most large manufacturing and aerospace companies had their own internal event driven simulation languages. A number of the early simulation languages are shown in Figure 2.1. Simula67 [28] is one of the most popular among them due to its flexibility based on object-oriented design of modules.

The early development of CEP can be credited to the work carried by event driven simulations team at Stanford university including the use of event patterns and event abstraction [43]. At that time, the output of event driven simulators used to produce a single stream of events ordered in time as they were computed by the simulator. Most of the simulators did not have the capability to show sequentiality and parallelism of events in the system. Instead, it was the observer's task to figure it out based on their own understanding. In contrast to it, a team of researchers at Stanford university while working on Rapide project led by David Luckham [43] applied CEP techniques for the first time in order to analyse the patterns of low-level events and extract more meaningful higher-level events. The higher-level events facilitate the observer and made their task easier. Furthermore, it reduces the human error by making the process automated. The number of events produced by simulations decrease exponentially as

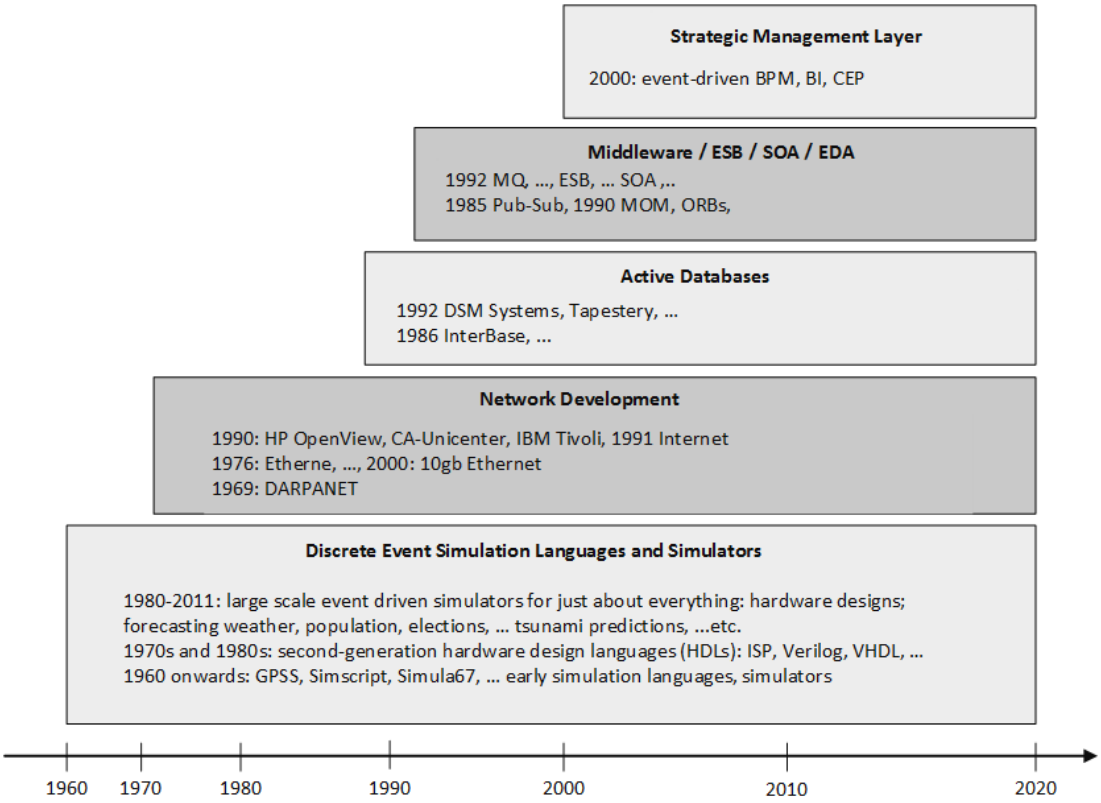


Figure 2.1: Background of event processing [23]

the level of output events increase, which in turn magnifies the errors in the higher-level events making it easier to detect them.

Today event driven simulation is a big business. The design of almost any product is subjected to some degree of simulation before it is produced. For instance, designs for a new computer chips are subjected to extensive simulation at every design level before they reach the production phase. In fact, it is more costly to fix the error if it is detected further down the production line. The use of large-scale event driven simulations can be found in many other technology sectors such as communication networks or smart city simulations.

2.1.2 Networks

The development of event processing has also been a fundamental part of networking. The core idea is to use simple types of events in the form of network protocols to build

communication between computers. The communication is in the form of packets, where a packet represents a sequence of bits encoding a small amount of data, with packet origin and destination in the network. In order to ensure that a packet arrives at the destination, computers use protocols to acknowledge the receipt of each packet or resend if an acknowledgement is not received within certain time limit.

At the lowest level, event processing in a communication network is simple message processing. Consider an example of sending an email through a network, first it is broken down into a set of packets where each packet is transmitted using the network protocols. At the destination, all the packets are reassembled into the original email. It involves a continuous activity within the network, which consist of various low-level events involved in sending and receiving the network packets.

In recent times, event processing is ubiquitous in the form of internet and mobile networks. The processing and understanding of events at such a large scale has become an extremely complicated problem with several concerns, including quality of service, security concerns such as hijacking of events or cyber attacks and monitoring applications

2.1.3 Active Databases

Event processing finds its roots in the development of active databases as well. Traditional databases are passive by nature which means they are only responsive to the called queries. In contrast, active databases are capable of invoking other applications without waiting for queries [90]. For instance, an active employees database might initiate an action of generating a warning message to the employee after detecting that the number of absentees for the employee is greater than the threshold level set in the database rules. Active databases extend traditional databases with a layer of event processing rules called event-condition-action (E-C-A) rules together with event detection mechanisms [64].

The core idea behind the active databases is to extend the capabilities of traditional databases to make them more reactive and fulfil the increasing requirements of real-time processing. The amount of event processing in active databases is not as significant as

in networking or in simulations, still a major part of their functionality is being driven by events.

2.1.4 Middleware

The development of middleware technologies started in late 1980s and came into limelight in 1990s. It exploded in different directions with the start of 21st century. The main purpose of the development of middleware is to conceal the complex details of the events and their transmission protocols from the users to provide a generic high-level communication interface. The development of middleware can be credited as the first step towards the hierarchical event processing [42].

The development of middleware has been in two major directions; synchronous communication based on remote procedure call (RPC) [56], and asynchronous communication based on messages, known as message-oriented middleware (MOM) [19]. In synchronous communication, a request for a service (the remote call) has to be made and wait until the reply has been received (the service answer). New services could be added in the system through a registy component. Such RPC paradigms served as the main building blocks across different enterprises. The common object request broker architecture (CORBA) [62] is the classic example of synchronous communication for middleware. Although the service call and the service response can be viewed as the event communication, the major drawback associated with the synchronous communication is that it wastes time in requests and responses, making it unsuitable for time sensitive applications.

In contrast to RPC, MOM middleware is based on asynchronous communication. The simplest example of MOM middleware is the publish/subscribe paradigm, most commonly known as the pub-sub. The pub-sub architecture provides an API which allows applications to publish messages under specific topic. Similarly, applications can receive the messages by subscribing to the topics using APIs provided. This is an example of the one-to-many communication architecture. It is intrinsically asynchronous, because a publisher does not wait to get responses from subscribers, and subscribers can be engaged in other activities while messages on their subscribed topics list are be-

ing published. The pub-sub architecture today serves as the backbone for many CEP networks. Other common middleware paradigms for transporting events include point-to-point event transmission and message queuing. Although the concepts of middleware did not contribute directly to the development of the CEP, it provides a building block for the reliable communication of events expediting the process of implementation of the CEP applications.

2.2 CEP concepts

2.2.1 Events

An event is defined as an occurrence within a particular system or domain; it is something that has happened or is contemplated to happen in that particular domain. For example, in an intelligent transportation system (ITS), the driver applying brakes or a passenger boarding the bus are examples of different events. Similarly, in a smart office space, switching on any equipment represents an event. Few more examples of events are given below.

- A sensor generating a new recording,
- A message that reports an RFID reading,
- A new comment or a like on social media picture,
- A stock ticker message that reports a stock trade.

2.2.2 Complex Events

A complex event can be defined as any high-level event detected by the specific pattern of low-level events. It can be a temporal pattern of events, spatial patterns of events or a correlation of different events. For example, a complex event of fire in a building is detected by the correlation of low-level events of smoke and high temperature; similarly a temporal pattern of stock prices continuously falling indicates a complex event of a market crash.

2.2.3 Event Driven Architecture

Event driven architecture (EDA) is an architecture which supports design and implementation of applications in which events are transmitted from decoupled services or components [22]. In EDA, node publish the events on a messaging service as the events happen and does not depend on the availability of requests. It uses messaging to communicate between two or more applications. As the events happen, they are published so that the relevant subscribers can be notified and activated. CEP is an extension of such architectures.

2.2.4 Generic CEP Architecture

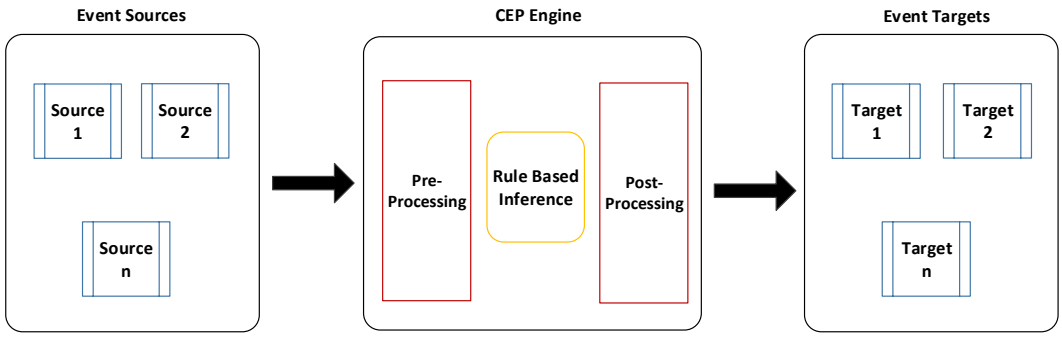


Figure 2.2: Generic CEP architecture

A generic architecture of a CEP system is shown in Figure 2.2. The event sources are responsible for producing or sending the events to the CEP engine. They can be low-level events in the form of a raw sensor data, can be extracted high-level events after performing pattern matching mechanisms or even the output of any complex machine learning unit. The hierarchical event processing is built on this concept where the output of one operator can serve as the input for another. In this manner, a system can handle complex scenarios by building on the top of other low-level systems. Pre-processing is an important step as the CEP engine provides generic interface in order to take events from multiple event sources. Event sources can provide data in different formats and it needs to be converted into format understandable by the

CEP engine. After the pre-processing step, events from different sources are combined, aggregated and correlated using temporal and spatial relations to detect meaningful patterns. CEPs perform these tasks with the help of different components which forms the basis of every CEP engine. These components are explained as follows;

2.2.4.1 Filters An event filtering is the basic functionality which supports other more complex patterns. Not every event is of interest for the consumers and a user might be interested in only specific events. Lets consider a simple example of a temperature sensor which generates a reading every second; if a user is only interested in the temperature greater than a specified threshold which is defined in the filter. Then if only the conditions becomes true, the event would be published to the observer.

2.2.4.2 Windows Windows provide a tool to extract limited number of events to analyse and infer a complex event from the stream at any given time. The two most basic types of windows are:

Time Window: The temporal relation between different events plays an important role in evaluating a complex event. For example 5 °C temperature change in a room in 1 hour will have different meaning as compared to the same temperature change in 1 minute. The former observation might result from the heater being switched on and the latter might be caused by a fire. The time window can be a fixed time window or a sliding time window. Basic arithmetic tasks like finding the maximum/minimum or the aggregated value also requires the definition of the time window.

Tuple Window: In contrast to the time window, the tuple window acts on the number of events defined. Aggregation of every five samples is a typical example of the tuple window operation.

2.2.4.3 Joins The functionality of joins is to correlate events from different streams using basic logical operations including boolean operators. For example if we have the data from a temperature sensor and a smoke sensor, a more complex event fire can be derived as

$$\text{If (temp} > \text{Threshold \&\& Smoke} == 1) \implies \textbf{Fire}.$$

2.2.4.4 Aggregation Most of the CEPs have built in aggregation functions such as sum, min, max and count to assist for simple analysis. More advanced platforms also offer statistical parameters such as standard deviation and variance. The aggregation functions always require a window to assign. Few versions of CEP offer extensions to write user defined aggregation functions. But care should be taken when writing a user defined function because computationally heavy functions can effect the streaming pipeline and hence the performance of the CEP.

The post-processing step is generally opposite of the pre-processing step where the CEP format is undone and target-specific format is created. The events could then be made available to the target as a push or a pull. Similarly as with the event sources, event targets can also vary from high-level decision making system to another CEP process. They can also be the input to more sophisticated machine learning processes.

2.3 How CEP fits in IoT world ?

IoT is significantly increasing the amount of data produced in every field of life. Intelligent transportation system (ITS), smart homes and automatic health monitoring are few of the IoT initiatives producing terabytes of data every day. The amount of data produced by social media applications is unparalleled to any other existing applications. The processing of such large data streams with the constraint of near real-time not only changes the use of traditional data mining tools but also affects the whole process of knowledge extraction and interpretation.

Figure 2.3 shows few IoT applications where event processing can be employed to extract high-level knowledge. These applications produce millions of low-level events

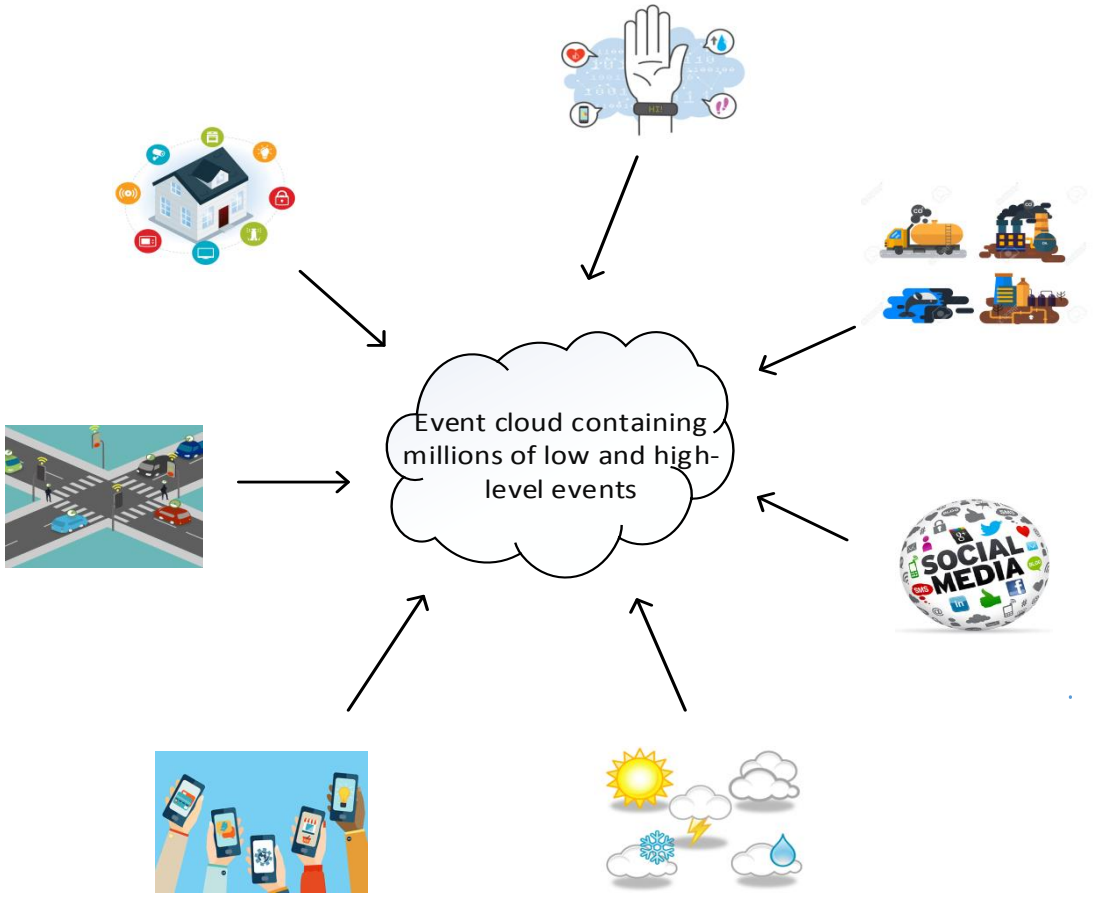


Figure 2.3: Event processing in the IoT world

which when analysed individually do not add significant amount of knowledge. CEP provides a perfect fit for analysing spatial and temporal patterns of low-level events to extract high-level events and hence reducing the number of events exponentially. High-level events can be used as input for hierarchical CEP or other sophisticated tools such as machine learning and artificial intelligence (AI) etc.

In this section, the use of CEP for IoT applications is briefly explained in the context of IoT data processing requirements. The development of technology in most of the cases is generic and can easily be applied to other IoT applications.

2.3.1 Intelligent Transportation Systems

The use of the CEP is increasing in many IoT applications due to its ability to process large volumes of events in near real-time. Intelligent transportation system is one of the most integral part of smart cities due to its social and economic impact on the standards of living. The increasingly high number of traffic sensors in cities along with the recent concept of crowd sensing [68] has resulted into amount of data being generated to a level where conventional data analysis methods are proving to be a limiting factor. This large amount of data generated has to be analysed with minimum time latency to manage traffic. The ability of the CEP to process data streams in near real-time makes it ideal candidate to provide solutions in this scenario.

One of the early work in this direction was mentioned by Dunken et al [21] where the authors proposed an event driven architecture (EDA) based on the CEP for analysing large traffic data to provide automatic solutions for decision support systems for the traffic management. They introduced the concept of event processing agents (EPAs) connecting streams of increasingly abstract types of events. They proposed a distributed hierarchical architecture where several instances of EPAs were deployed in every layer. The proposed solution was implemented by using Esper CEP [1] which is an open-source java based CEP system. In order to realize the distributed architecture, the authors have used message oriented middleware (MOM) as an event distribution service. Although they discussed the requirements of real-time data processing and correlating different data streams, but they did not support their solution with sufficient qualitative and quantitative evaluations. Another early work for using the CEP for traffic is shown in [77]. They used the data from different traffic sensors including loop detectors, radars and cameras and configured CEP rules with the help of domain experts to detect different complex events including congestion. Both of these research efforts demonstrate the application of the CEP as a potential solution for handling real-time fast data streams.

The number of sensors deployed in modern vehicles has also seen a significant rise and led to many research efforts intended to provide more contextual solutions for users. It

[1] http://www.espertech.com/esper/

involves processing and fusion of data from multiple sensors in real-time which makes the CEP an ideal candidate. The authors in [78] propose an analytics solution for smart vehicles using Esper CEP and introduced the concept of Intra-vehicular context awareness (IvCA). The approach uses different on-board sensors to detect the on-board passengers and their activities. Such information can be used to provide more context-aware solutions like playing the favourite track of passengers or locking the door next to a minor. This work indicates a future trend where CEP capabilities are being implemented towards the edge near to the data source.

2.3.2 Smart Homes

Smart homes is another application area which has seen the significant rise in the use of sensor based solutions. The deployment of smart energy meters, occupancy sensors and environmental sensors is becoming common in many domestic and commercial buildings [57]. A common goal for all these deployments is to monitor and correlate data coming from different sensors in real-time and take actions accordingly. For instance, if heating unit is detected in ON state when there is no one at home, it should be switched off in order to save energy. CEP provides the capability to monitor and correlate this data in real-time using simple rule-based approach. In this regard, many examples are found in the literature where researchers have proposed solutions based on CEP for smart homes.

One of the early work in this direction is mentioned in [83], where authors highlighted the drawbacks of current semantic web technologies for real-time applications and discussed the gains of event-driven architecture in contrast to semantic web request-response architectures. Following the concepts of event processing network introduced by David Luckham [42], they provided the basics for a CEP framework with inference and reasoning capabilities to work with linked datasets and discussed its use for the real-time energy optimising service for the smart homes. They presented their solution at abstract level as technology was not matured enough at that time.

Another work mentioned in [82], where authors proposed a framework for energy monitoring and management in the factory in order to assist decision support systems.

They highlighted the difficulties to apply CEP in an industrial environment such as collecting data from different processes running in noisy environments, problems with synchronization of data, unnecessary data or redundant data and high volume of network traffic. They deployed distributed CEP with event processing agents running in an embedded hardware responsible for cleaning, filtering and extracting abstract events before transmitting it to the server. CEP running on server gathers data from different processes and calculates the manufacturing system performance according to defined energy KPIs in real-time. They demonstrated their solution by deploying the prototype in machining line of a major European automotive manufacturer and showing how it increases energy efficiency and helped them in diagnosing failures reactively by analysing and correlating data streams in real-time.

Another work which proposed a distributed architecture for the deployment of CEP is discussed in [13] for smart buildings. Authors proposed a client server based architecture and demonstrated that it can be used to separate pre-processing and reasoning parts which will contribute towards less amount of data flow across the network, thereby, reducing the burden on the central server. They showed that CEP based architecture enables users to inject simple rules in order to contribute for applications such as managing lighting system using occupancy schedule and monitoring devices for malfunctioning. Environmental monitoring in buildings represent another application area where CEP has the potential to provide real-time solution for extracting knowledge as shown in [53]. They captured data from different environmental sensors measuring temperature, smoke and carbon monoxide and correlated it with occupancy sensors data using a CEP running in middleware to detect crowded places in real-time in the event of a Fire. Such information can be used by fire fighters to reduce damage and save human lives.

2.3.3 Social Media

Social networks, blogs, mobile services and applications generate large volumes of data that can be seen as ‘sensors’ of people’s moods, interests, relations, etc. The analysis on social media data have significant potential on different applications like poll analysis,

stock market predictions, public opinion analysis. Data is generated in the form of events where every new status or picture upload, every tweet, or every like/comment can be seen as an event. Event processing tools provide the capability to monitor, aggregate or correlate the data coming in the form of events with different sources.

One of the goals of the analysis of social media data is to capture the hidden relationships between people and concept. One such application is mentioned in [48], where authors used CEP to find community relationships. They highlighted the increasing demands of processing data for social network graphs in near real-time for social applications and proposed an architecture based on parallel CEP and graph processing called GraphCEP. They demonstrated their solution using two different use cases. First use case involved detecting top posts based on the likes and comments where second use case was targeted to find community relationships. Both use-cases involve analysing large and dynamic data in near real-time. In [73], authors have discussed that not many real-time public mood tracking frameworks over social media streams are available. To address this issue, they proposed a hierarchical framework for real-time public mood time series tracking over Chinese microblog streams using CEP. They used text sentiment analysis to transform microblogs into emotional microblog events and later applied CEP operators like sliding windows and pattern matching to derive higher level events indicating public moods.

2.3.4 Remote Health Monitoring

Remote health monitoring is another potential application of CEP which has gained significant importance with the advent of IoT. It started with the use of discrete event simulations for healthcare applications [32] and as the technology evolved, it has found its applications for more real-world problems [85]. It involves deploying different sensors in the form of wearable devices collecting patients information to gain insights about the context and situation of patients. These devices generate fast data in near real-time which needs to be analysed as soon as possible in order to ensure safety of the patients. Every reading indicating heart rate, blood pressure or current blood sugar level represents an event which can be correlated with other events such as physical

activities of the patient, pollution level of their surroundings or dietary information to extract high-level knowledge.

2.3.5 Supply Chain Management

Supply chain management is also one of the major area which has experienced a significant impact with IoT. The monitoring of food and pharmaceutical products for ensuring their quality during the complete cycle of supply chain is an interesting application of IoT. However, the increasing number of connected devices and real-time monitoring requirements is one of the challenging factor in this regard. The ability of CEP to process data streams in near real-time makes it ideal candidate for such situations as the authors proposed autonomous monitoring of sensitive goods in [72] using Esper CEP[1]. There solution was based on simple rules for continuously monitoring if the surrounding temperature for products lies within the prescribed range. It involved manual setting of threshold values for normal working range.

[1] http://www.espertech.com/

Table 2.1: Applications of CEP in IoT

literature	application	real-time processing	heterogeneous event processing	Scalability
Nicholas et al [68]	ITS	yes	-	yes
Jurgen et al [21]	ITS	yes	yes	yes
Bogdan et al [77]	ITS	yes	yes	-
Fernando et al [78]	ITS	yes	yes	-
Andreas et al [83]	smart homes	yes	yes	-
Konstantin et al [82]	smart homes	yes	yes	-
Ching et al [13]	smart homes	yes	-	-
Marina et al [53]	smart homes	yes	yes	
Ruben et al [48]	social media	yes	yes	yes
Si Shi et al [73]	social media	yes	-	-
Jun et al [32]	Health	-	yes	-
Di Wang et al [85]	Health	yes	yes	-
Septimiu et al [72]	Supply chain	yes	-	yes
Bogdan et al [81]	Geofencing	yes	-	yes

Table 2.1 summarises the application of CEP for extracting knowledge from IoT data streams. It is evident from the Table 2.1 that CEP is mostly deployed to meet the requirements of real-time data processing. Heterogeneous event processing is another important factor which makes CEP as a potential solution for different IoT applications. However, only few research efforts addressed the issue of scalability while deploying CEP based solutions.

2.3.6 Geofencing

CEP is a generic tool to analyse and correlate events which makes it suitable for variety of applications as evident by its use for providing a geofencing service in order to monitor physically disabled individuals [81]. As geofencing service involves continuous streams

of mobility data, CEP provides a perfect fit to analyse this data and continuously monitor the journey. The use of CEP also enables it to further integrate the service with event rich environments, such as those resulting from smart city infrastructure.

2.4 Challenges and Research Gaps

In the previous sections we have discussed how CEP addresses the requirements of data processing for different IoT applications. In this section, we highlight the challenges in CEP based solution while pointing out the gaps this thesis aims to close.

2.4.1 Challenge 1: Predictive Event Processing

Most of the CEP applications found in the literature are intended to provide reactive solutions by correlating data streams using predefined rules as the events happen and does not exploit historical data due to its limited memory [24]. However, in many applications, prediction of a forthcoming event is more useful than detecting it after it has already occurred. For example, it will be more useful to predict traffic congestion as compared to detecting it, so that traffic administrators can take preventive measures to avoid it. The advantages of predicting an event are more obvious if we imagine the gain of predicting natural disasters and disease epidemics.

On the other hand there are several methods found in literature based on ML and statistics which have the ability to provide innovative and predictive solutions in different domains. For instance, predicting passengers travel time for ITS [91] and energy demand for buildings [100] are two of the many examples found in literature. However, these methods are unsuitable to analyse and correlate different data streams in near real-time due to the requirement of historical data for training. ML methods exploit historical data and apply diverse approaches such as probability, statistics and linear algebra to train the models in order to make predictions about the future. They have the potential to provide the basis for proactive solutions for IoT applications but lack the power of scalability and processing multiple data streams in real-time which is provided by CEP [31] .

In literature, CEP and ML have been explored extensively as separate research fields and were mostly targeted for different types of applications. CEP has been designed for processing and correlating high speed data on the fly without storing it [16]. Whereas ML methods are targeted for applications which are based on the historical data for knowledge extraction[79].

In recent years, the diverse requirements for processing data in IoT led to hybrid approaches where predictive analytics (PA) methods based on ML and statistics are combined with CEP to provide proactive solutions. Initially, it was proposed in [30], where authors presented a conceptual framework for combining PA with CEP to get more value from the data. The approach of combining both methods lead to encouraging results; however, they did not support their idea with any practical application. Another example is given in [93] where authors used probabilistic event processing network to detect complex events and then used these complex events to train multi-layered Bayesian model to predict future events. Their proposed prediction model employs expectation maximization (EM) algorithm [33]. EM being an iterative optimization algorithm has high computational cost. Its complexity increases exponentially as the training dataset increases, therefore making it unsuitable for large scale IoT applications. They demonstrated their solution on the simulated traffic data with the assumption of availability of statistical data of vehicles which is unlikely to be available in a real-world use-case.

In [55], authors provide a basic framework for combining time series prediction with CEP to monitor food and pharmaceutical products in order to ensure their quality during the complete cycle of supply chain. The authors highlighted the open issues related to prediction component such as model selection and model update as new data arrives but they did not address these issues and left it for their future work. Another example of using time series prediction of data for CEP in order to provide predictive IoT solutions is mentioned in [5], where authors implemented neural network for prediction. They demonstrated their approach on the traffic data and used 60 days of data to train the neural network. Once trained, model parameters are static which is a major drawback. The statistics and behaviour of underlying data may change over time due to concept drift [89] which can effect the accuracy and performance of the neural network. Furthermore, the model is unable to update and adapt to

changes. In case of erroneous readings, errors will propagate and potentially will keep on increasing eventually effecting the reliability of the system. This thesis addresses the above mentioned challenges as a first contribution and explained the proposed approach in chapter 4.

2.4.2 Challenge 2: Automatic Rule Generation for Context-Aware Event Processing

In order to detect complex events, systems based on CEP require rules which have to be provided manually by the system administrators. Based on this, there is an assumption that administrators have the required background knowledge which sometimes is neither available nor so precise. The manual setting of rules and patterns limit the use of CEP only for domain experts and poses a weak point. And even with prior expertise and knowledge, experts are prone to make errors in choosing optimized parameters for dynamic systems. Current systems based on CEP deploy static rules and there is no means to update the rules automatically. In real-time dynamic IoT applications, the context of the application is always changing and the performance of CEP will deteriorate in such scenarios.

In literature, there are quite a few examples where the researchers have applied the concepts of machine learning domain for optimizing the rules of CEP. One such example is given in [46], where authors discuss the possibility of applying machine learning techniques for finding the rules. They applied information gain ratio (IGR) to optimize the detection of already recorded events. Their system was application dependent and easily prone to fail in situations where prior knowledge about the events is unavailable.

In [80], authors presented a mechanism for automating both the initial definition of rules and their update over time. Their proposed solution was based on iterating over two stages; rule parameter prediction and rule parameter correction. The first stage is based on updating parameters using an expert knowledge and the updated history of already materialized events while the second stage is based on expert feedback of actual occurences of complex events in order to fine-tune the rule parameters for the next prediction stage. They highlighted the issues of finding optimal rule parameters

or threshold values and proposed a solution based on the optimal statistical estimators exploiting principles of kalman filter. They used MIT Lincoln Laboratory DARPA On-Line intrusion detection evaluation data set [41] to validate their approach. Although the performance in terms of recall and precision were not as good as tailor-made solutions but nevertheless it provides a generic and automatic approach towards rule definition.

Another approach to find CEP rules is mentioned in [54] where authors have explored time series classification to learn patterns from historically data in the form of shapelets and then algorithmically transform those shapelets into CEP rules. The output of the experiment is dependent on the length of shapelets or patterns which are chosen by domain experts. Their work is still in its early stages with not much implementation details available. Also, their proposed solution does not support multivariate time series which is often the case in IoT applications.

All the methods discussed above are based on frequent event pattern mining approach based on the frequency of events. As a result, event patterns consist many primitive events which does not contribute to finding optimized rules and increases the complexity of the algorithm and can lead to ambiguous rules. In contrast to it, authors in [39] proposed a sequence clustering-based automated rule generation (SCARG) algorithm for automatically generating CEP rules by mining decision making history of domain experts. Their approach is composed of four parts. First, selection of land mark windows consisting of primitive and complex events based on domain experts reactions. In the second step, the sequence clustering algorithm is applied on the land mark windows to find similar patterns. Every cluster in the model represents a particular complex event pattern. Third, every cluster is graphically modelled using Markov Probability Transition Model and finally in the fourth step, node centrality measures are used to prune the nodes and get rid of primitive events which does not contribute towards complex pattern and thus making the model simpler. They demonstrated their solution using NASDAQ 100 stock data provided by Yahoo finance and showed significant gain in terms of return as compared to approach mentioned in [80]. Although they have shown significant gain, however sequence clustering algorithms are prone to data sparsity issues and their performance can deteriorate if sufficient data is not

provided for training. Their proposed solution was unable to provide updates in real-time due to complexity of the underlying algorithms. In contrast to existing approaches, the second contribution of this thesis (chapter 5) proposes a context-aware method for event processing where CEP rules are found using machine learning and are updated automatically with respect to the current context.

2.4.3 Challenge 3: Uncertainty in Event Processing

In conventional CEP systems, events are correlated using absolute rules where complex event detected is either true or false. Even if a single condition or rule is violated, complex event will not be generated. It is a major drawback given the random and probabilistic nature of real-world events. In contrast to conventional CEP systems, researchers have proposed to use the principles of probability and statistics to take the inherent uncertainty of events into account.

One of the early work to introduce the uncertainty and probabilistic processing of events was presented in [36] where authors build their system PEEX on the top of traditional relational database management system (RDBMS) in order to detect probabilistic events from RFID data. They validated the system on a real-world deployment of RFID tags for recognizing activities. From an implementation point of view, PEEX uses a RDBMS where it stores all information received from sources and information about confidence. Rules are then translated into SQL queries and run periodically. For this reason, the detection time of events may be different from real occurrence time with a small delay.

Another initial approach for handling uncertainty was proposed by authors in [87], where authors discussed two types of inherent uncertainty in CEP systems; uncertainty in data and uncertainty in relation between events. They presented theoretical framework for embedding uncertainty in events and later same authors extended their work in [88]. They proposed a mechanism for constructing probability space that captures the semantics of defined rules and used an abstraction based on Bayesian networks to define the probabilities of possible worlds. Bayesian model was automatically created based on defined rules and explicit events. Experiments were performed on simulated

data with the assumption of conditional probabilities given for Bayesian network. The authors highlighted the difficulties in obtaining conditional probabilities for real-world scenarios which limits the practical implementation of their work. Another recent work [17] also proposed to construct a Bayesian network from a rule by extending existing CEP language TESLA [14]. They implemented the proposed system on the top of T-rex cep [15] adding an overhead for CEP. It models every rule using a BN. The uncertainty about the events is modeled using theory of probability and assuming that the probability density function of measurement error of the events is given which is highly unlikely to be available in many real-world problems.

All the existing work on using Bayesian network with CEP assumed that conditional probabilities are given which are vital for constructing Bayesian network. It is not true for many real-world scenarios which makes this assumption invalid. The process of calculating conditional probabilities can be tricky for real-world applications due to sparsity and scale of data involved. As the data size and number of data sources increases, the process of Bayesian inference becomes more complex and requires suitable methods. In this regard, the third contribution of this thesis (chapter 6) addressed these challenges and proposed a novel method based on BN and CEP for handling uncertainty for large-scale IoT applications.

2.4.4 Challenge 4: CEP and Big Data Management

Analytics frameworks for Big Data can often be categorized as either batch or real-time processing frameworks. Batch processing frameworks are suitable for efficiently processing large amounts of data with high throughput but also high latency - it can take hours or days to complete a batch job. Real-time processing typically involves time sensitive computations on a continuous stream of data.

One of the most common and widely used techniques for batch processing on Big Data is called MapReduce [20]. MapReduce is a programming model for carrying out computations on large amounts of data in an efficient and distributed manner. It is also an execution framework for processing data distributed among large numbers of machines. It was originally developed by Google as a generic but proprietary framework

for analytics on Google's own Big Data, and later was widely adopted and embodied in open source tools. MapReduce was intended to provide a unified solution for large scale batch analytics and address challenges like parallel computation, distribution of data and handling of failures.

Hadoop [1], an open source embodiment of MapReduce, was first released in 2007, and later adopted by hundreds of companies for a variety of use cases. Hadoop provides generic and scalable solutions for big data, but was not designed for iterative algorithms like machine learning, which repeatedly run batch jobs and save intermediate results to disk. In such scenarios, disk access can become a major bottleneck hence degrading performance. In order to overcome the limitations of Hadoop, a new cluster computing framework called Spark [95] was developed. Spark provides the ability to run computations in memory using resilient distributed datasets (RDDs) [94] which enables it to provide faster computation times for iterative applications as compared to Hadoop. Spark not only supports large-scale batch processing, it also offers a streaming module known as Spark streaming [96] for real-time analytics. Spark streaming processes data streams in micro-batches, where each batch contains a collection of events that arrived over the batch period (regardless of when the data was created). It works well for simple applications but the lack of true record-by-record processing makes time series and event processing difficult for complex IoT applications.

The need for real-time processing of events in data streams on a record-by-record basis led to the development of CEP. The CEP is specifically designed for latency sensitive applications involving large volumes of streaming data with timestamps such as trading systems, fraud detection and monitoring applications. In contrast to batch processing techniques which store the data and later run queries on it, CEP instead stores queries and runs data through these queries. The inbuilt capability of CEP to handle multiple seemingly unrelated events and correlate them to infer complex events make it suitable for many IoT applications. A drawback of CEP is that the authoring of these rules requires system administrators or application developers to have prior knowledge about the system which is not always available.

[1] http://hadoop.apache.org/

Table 2.2: Challenges and research gaps

literature	predictive event processing	context-aware event processing	uncertainty in event processing	big data management
Fulop et al [30]	✓			
Wang et al [93]	✓			
Nechifor et al [55]	✓			✓
Thomas et al [5]	✓			✓
Margara et al [46]		✓		
Turchin et al [80]	✓	✓		
Mousheimish et al [54]	✓	✓		
Lee et al [39]		✓		
Peex [36]			✓	
autoCEP [88] [87]			✓	
Cugola et al [17]			✓	

Big Data analytics systems have the challenge of processing massive amounts of historical data while at the same time ingesting and analysing real-time data at a high rate. The dichotomy of event processing frameworks for real-time data, and batch processing frameworks for historical data, led to the prevalence of multiple independent systems analysing the same data. The Lambda architecture was proposed by Nathan Marz [47] to address this, and provides a scalable and fault tolerant architecture for processing both real-time and historical data in an integrated fashion. The purpose of this architecture was to analyse vast amounts of data as it arrives in an efficient, timely and fault tolerant fashion. Its focus was on speeding up online analytical processing (OLAP) style computations, for example web page view and click stream analysis. It was not designed to make per-event decisions or respond to events as they arrive. The dichotomy between batch and event processing poses a major challenge for a unified architecture which is suitable for both type of applications. In addition, the architecture should be able to manage and analyse large data in a scalable manner. This represents a general challenge which was considered in all three contributions.

2.5 Discussion

This chapter provides an overview on the use of CEP for IoT applications and identify open research challenges and future research directions. CEP fulfils many requirements for IoT data processing including real-time processing of data, correlating heterogeneous data streams and providing scalable solutions which makes it a potential solution for extracting knowledge. However, the diverse nature of IoT data poses many challenges in this regard.

Four major challenges were identified and explained in this chapter. First, the nature of CEP enables it to provide reactive solutions by correlating data streams using predefined rules as the events happen. The trend of many IoT applications is changing from reactive to proactive where complex events can be predicted before they actually happen. It formed the basis of different research efforts in this direction. Second, solutions based on CEP require manual setting of rules which limits its use only to domain experts. In addition, these rules are static and unable to update it according to current context. This represents a second major challenge towards the successful application of CEP for many IoT applications. Third, CEP provides an absolute solution where a complex event detected is either true or false. Even if a single event or condition is missing, a complex event will not be generated. Given the sporadic nature of IoT, missing and uncertain data is a common phenomenon and CEP systems of today are unable to take this inherent uncertainty of real-world events into account while taking decisions. Fourth, any proposed solution based on CEP should be able to manage and fulfil the ever increasing data being generated by IoT devices.

The following chapters of this thesis demonstrate the proposed approach of this research work towards addressing these challenges with the help of suitable evaluations methods.

Chapter 3

Overall System Framework

This chapter presents an overview of the system framework with key functionalities identified which were developed during this research work. This work demonstrates the functionalities with the help of Intelligent Transportation System (ITS) scenario where inputs are taken in the form of different traffic parameters, weather data and social media data. In this thesis, first contribution was developed independently whereas contribution 2 and contribution 3 were developed as part of the same framework. However, all three contributions have the potential to deploy as part of the same framework as shown in Figure 3.1.

3.1 Data Ingestion

Data Ingestion serves as the first step of the proposed framework. IoT has provided the researchers with a global view enabling access to truly heterogeneous data sources for the very first time. These data sources can be RESTful web service [67], MQTT data feed[1] or any other external data source. Data format is not limited to any specific format in IoT. XML[2] and JSON[3] are two most commonly used formats which are used extensively for transmitting IoT data. Also, different data feeds from different sources

[1] http://mqtt.org/

[2] https://www.w3.org/XML/

[3] https://www.json.org/

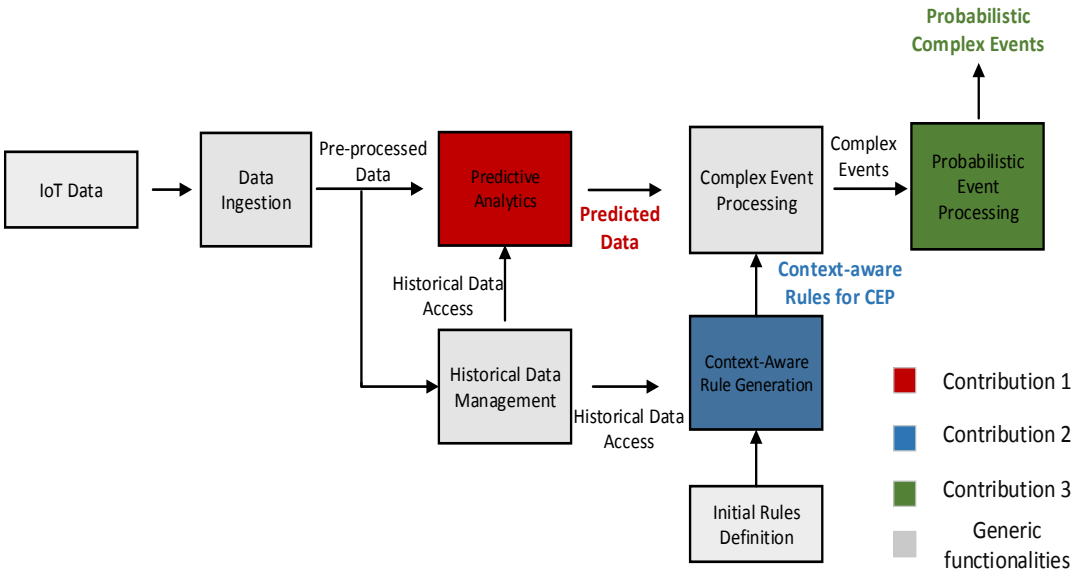


Figure 3.1: Overall System Framework

may contain data which might be redundant for a specific application and needs to be filtered out. All of these functionalities were covered under data ingestion block.

3.2 Predictive Analytics

The first contribution of this thesis addresses the problem of predicting complex events (this challenge is described earlier in sections 1.3 and 2.4.1). In unified system framework, this block will be responsible for predicting time series data in real-time. This thesis implements this functionality as part of a stand-alone system as shown in the Figure 3.2.

It includes data ingestion, real-time prediction, correlation using CEP and finally error estimation for dynamic IoT data streams. This work demonstrates the functionalities of first contribution using real-time traffic data which includes traffic speed and traffic intensity as input to predict traffic states.

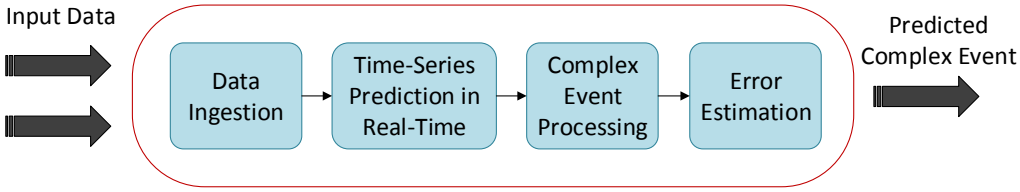


Figure 3.2: Contribution 1 Functionalities

3.3 Context-Aware Rule Generation

The second contribution of this thesis addresses the challenge of manual setting of rules and updating it according to the current context (this challenge is described in sections 1.3 and 2.4.2). This functionality can be deployed in a unified framework as shown in Figure 3.1 or deployed independently as shown in Figure 3.3. Data is ingested from IoT data streams and then optimized threshold values is found by analysing historical data using methods from machine learning domain. Models are updated according to current context in order to provide optimised threshold values. Finally, complex event is detected using CEP.

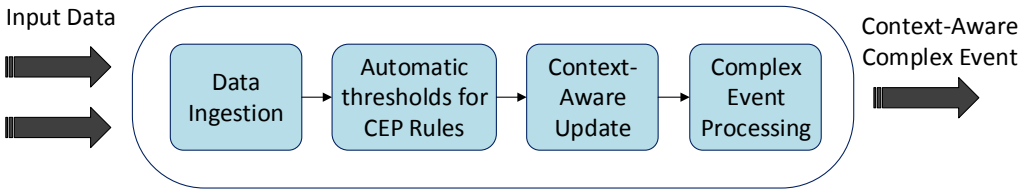


Figure 3.3: Contribution 2 Functionalities

3.4 Probabilistic Event Processing

The third contribution of this thesis addresses the challenge of handling uncertainty when combining heterogeneous events using CEP (this challenge is described in sections 1.3 and 2.4.3). This functionality is build on the top of contribution 2 by extending

it using Bayesian networks (BNs). Figure 3.4 shows different functionalities of contribution 3 where data ingestion and derived events block were part of contribution 2.

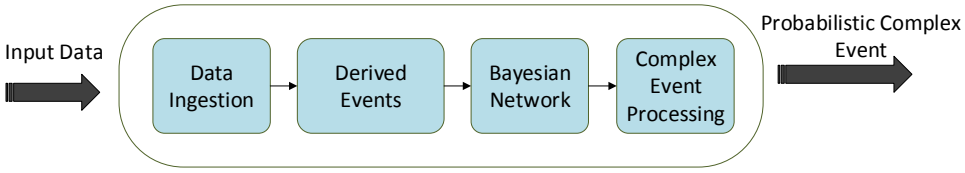


Figure 3.4: Contribution 3 Functionalities

3.5 Complex Event Processing

Complex event processing (CEP) represents a general component used in all three contributions. CEP can be further divided into three functional components; First event collector, which is responsible for collecting data streams from different IoT sources. After collecting the data, the event collector converts it to the specific format as required by CEP engine. Second, the CEP engine is the core of CEP component which detects high-level events by matching patterns using event processing language (EPL) statements. Third, event producer component detected complex events are forwarded to the relevant applications in a data format required by the applications using adapters in event producer component.

There are several CEP platforms available in the market such as WSO2[1], Tibco[2], Esper[3] and Apache Storm[4]. This work is developed using Esper CEP due to its enriched Java embedded architecture which supports strong CEP features set with high throughput. An open source status of Esper is another major factor which makes it suitable candidate for this research.

[1]https://wso2.com/products/complex-event-processor/
[2]https://www.tibco.com/streaming-analytics
[3]http://www.espertech.com/
[4]http://storm.apache.org/

3.6 Use-case of ITS

This work demonstrates the functionalities of proposed framework with the help of Intelligent transportation system (ITS) use-case. The use-case of ITS is chosen because of its massive data size and real-time processing requirements. In addition, ITS has an immense impact on the social and economical development of smart cities. The advent of IoT has made many data sources available but currently, most of them are deployed as a stand-alone systems. Such individual systems are limiting the true potential of IoT.

If we take the example of ITS, the use of traffic sensors is the conventional method for traffic administrators to observe and manage traffic. Although, many modern day cities have weather sensors installed and different research efforts [1][58] indicate the correlation of weather on traffic but it is not being exploited at city level applications. Similarly, in the current era of digital technology social media is one of the quickest way to detect city events effecting traffic. For example, a concert or a football match happening in nearby region can lead to bad traffic and analytics on social media data streams can help to detect it in timely manner. Although different research efforts demonstrate the potential of using social media data like twitter to detect city events in real-time [3] [101], they are seldom used in favour of more conventional sensor based methods for managing traffic.

In this work, data is ingested from traffic sensors, social media and weather data streams in real-time using the proposed framework. The proposed framework enables to analyse this data to extract high-level knowledge where high-level knowledge is in the form of current or predicted traffic state which can help the city administrators to manage traffic more efficiently.

Chapter 4

Predicting Complex Events

This chapter presents the approach towards predictive analytics as highlighted in chapter 3 as a first contribution of the thesis[1]. After a brief introduction about the problem in section 4.1 (more details can be found in section 2.4.1), proposed architecture is presented in section 4.2 along with the description of different components involved in the implementation. In this regard, a novel prediction algorithm called Adaptive Moving Window Regression (AMWR) is proposed for dynamic IoT data streams which is explained in the same section. In section 4.3, the feasibility of the proposed solution is demonstrated by implementing a prototype and evaluating the results on a real-world use-case. Moreover, a qualitative comparison is provided with other state-of-the-art regression algorithms where AMWR provides higher accuracy for different scenarios. Furthermore, error modelling for the prediction algorithm is provided using a parametric distribution along with the overall error of the system derivation. Finally, the chapter is concluded by highlighting how the proposed approach helps in solving the problem and closing the gap with respect to other state-of-the-art approaches (section 4.4).

[1]This approach has been presented in the following two publications:

1) A. Akbar, A. Khan, F. Carrez and K. Moessner, "Predictive Analytics for Complex IoT Data Streams," *IEEE IoT Journal,* PP(99):1-1, 2017.

2) A. Akbar, F. Carrez, K. Moessner and A. Zoha "Predicting complex events for pro-active iot applications," in *2015 IEEE World Forum on Internet of Things (WF-IoT),* pages 327-332, Dec 2015.

4.1 Introduction

CEP enables to correlate data coming from heterogeneous sources and extract high-level knowledge from it. Most CEP systems have SQL-like query language which enables to perform tasks like filtering, aggregation, joint and sliding window operations on different data streams and combine it with the help of simple rules. In contrast to batch processing techniques which store the data and later run queries on it, CEP instead stores queries and runs data through these queries. The inbuilt capability of CEP to handle multiple seemingly unrelated events and correlate them to infer complex events with minimum time latency provides CEP an edge on batch processing methods for many IoT applications.

Although CEP provides solutions to deal with data streams in real-time, it lacks the predictive power provided by batch processing methods including Machine Learning (ML) and statistical data analysis methods. Most of the CEP applications are intended to provide reactive solutions by correlating data streams using predefined rules as the events happen and does not exploit historical data at all. But in many applications like traffic management or health monitoring, early prediction of an event is more useful than detecting it after it has already occurred. The nature of IoT applications has recently changed from reactive to proactive leading to numerous research efforts towards hybrid solutions based on both ML and CEP as mentioned earlier in 2.4.1.

As mentioned above, there is growing interest in the research community to combine CEP and ML for predicting complex events; however, most of the research conducted in this direction is mainly theoretical and lacks implementation details or real-world use-case examples. To the best of author's knowledge, no work in literature has addressed the different challenges and open research issues for the efficient implementation of ML-based prediction with CEP. The complexity of underlying ML algorithms plays an important role as more complex algorithms are not suitable for real-time prediction and are unable to cope with the fast and dynamic IoT data streams. Although, streaming regression algorithms (e.g. Spark Streaming [97]) based on micro batch analysis [6] can provide faster solution but these algorithms do it at the expense of less accuracy [99].

In contrast to existing solutions based on CEP (mentioned in section 2.4.1, this work exploits both approaches (CEP and ML) and propose an hybrid solution addressing these drawbacks. The intuition behind the proposed work is that if the input to the CEP is predicted data, then the complex event detected by CEP using causal and temporal pattern recognition techniques will be a predicted complex event. In contrast to the current prediction methods which are based on static model parameters, an adaptive prediction algorithm called Adaptive Moving Window Regression (AMWR) is proposed for dynamic IoT environments, which utilizes moving window for training the model and updates the model as new data arrives. It tracks the error and prevent it from propagation. The size of the training window is found automatically by exploiting spectral components of time series data. The size of prediction window is also adaptive in nature in order to provide sufficient accuracy in the prediction. The proposed solution was implemented using open source components which are optimized for large-scale IoT applications.

In short, following contributions are made in this chapter:

- Propose and implement a generic framework based on open source components for combining ML with CEP in order to predict complex events for proactive IoT applications;
- Propose an adaptive prediction algorithm called adaptive moving window regression (AMWR) for dynamic IoT data streams and implemented it on a real-world use-case of ITS achieving accuracy ranging from 87 -96 % . AMWR is based on a novel method for finding optimum size for training window by exploiting spectral components of time series data;
- Model the error introduced by the prediction algorithm using a parametric distribution and derived expressions for the overall error of the system, as the error propagates through the CEP.

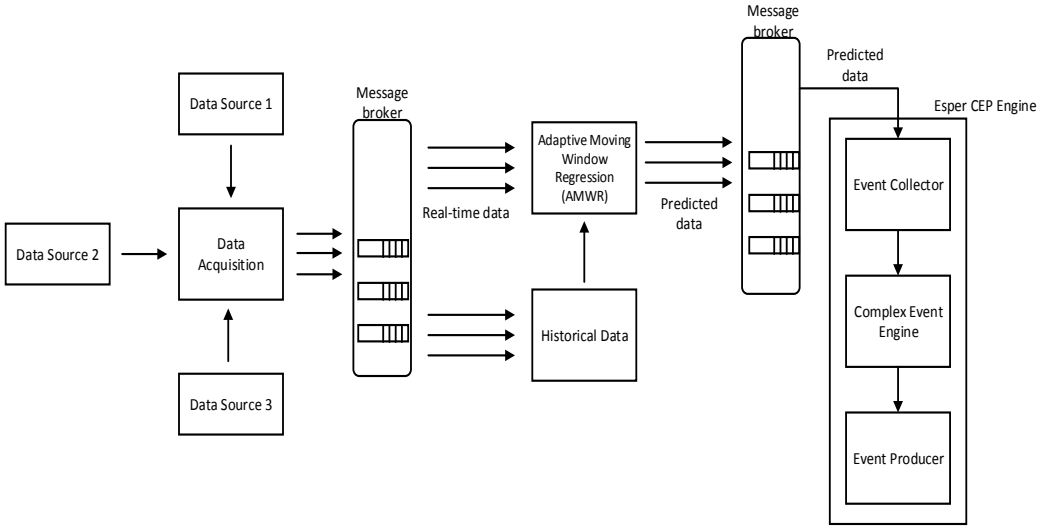


Figure 4.1: Proposed solution and block diagram

4.2 Proposed Solution

The proposed solution illustrating this contribution is shown in the Figure 4.1. One of the priorities of this research work is to propose a practical implementable solution for real-world applications and therefore, choosing the right components which are easily available, open source and scalable for the given application was an important factor. There were several issues which needed to be considered when choosing different components for the proposed architecture such as scalability and reliability of the system, integration of ML component with CEP, exchange of data across different components in near real-time, a common data format across all components in the architecture, are few of them.

In the proposed architecture, data acquisition block provides the front-end interface where data from different sources such as MQTT data feeds or a RESTful api are accessed; and after performing pre-processing tasks such as filtering redundant data and converting to required data format; it is published under a specific topic to the message broker. The AMWR block represents the ML component, it accesses the real-time data from the message broker and publishes back the predicted data under different topic in the form of an event tuple. The event collector module of CEP is listening to events and as soon as new events are available, it collects them, performs

pattern matching using CEP engine and detects the complex events.

More details about the proposed prediction algorithm and different components involved in the architecture are described below.

4.2.1 Adaptive Moving Window Regression (AMWR)

An adaptive prediction algorithm called Adaptive Moving Window Regression (AMWR) for dynamic IoT data was proposed and developed during this work. In general, prediction models are trained using large historical data and once the model is trained it is not updated due to limitations posed by large training time. Such models are not optimized to perform under phenomenon such as concept drift [89]. The context of the application may change resulting in the degradation of performance for prediction model. For such scenarios, this work proposes a prediction model which utilizes moving window of data for training the model; and once new data arrives, it calculates an error and retrains the model accordingly. As the model is trained using most recent data, it performs accurately even in the presence of concept drift. This work proposed to find the optimum size for training window by exploiting spectral components of time series data using Lomb Scargle method [63]. The proposed approach is adaptive in nature as it tracks down errors and prevents them from propagating by retraining the model periodically. The size of the prediction window (or forecast horizon) is also adaptive and derived by the performance of the model in order to ensure sufficient reliability in the prediction. The flowchart of the overall approach is shown in Figure 4.2. There are three main steps involved in the implementation of AMWR as described below:

1) Selection of regression algorithm;

2) Finding optimum training window size;

3) Size of the prediction horizon.

4.2.1.1 Selection of Regression Algorithm There are several algorithms available for time series regression (prediction) ranging from statistical to pure ML domain.

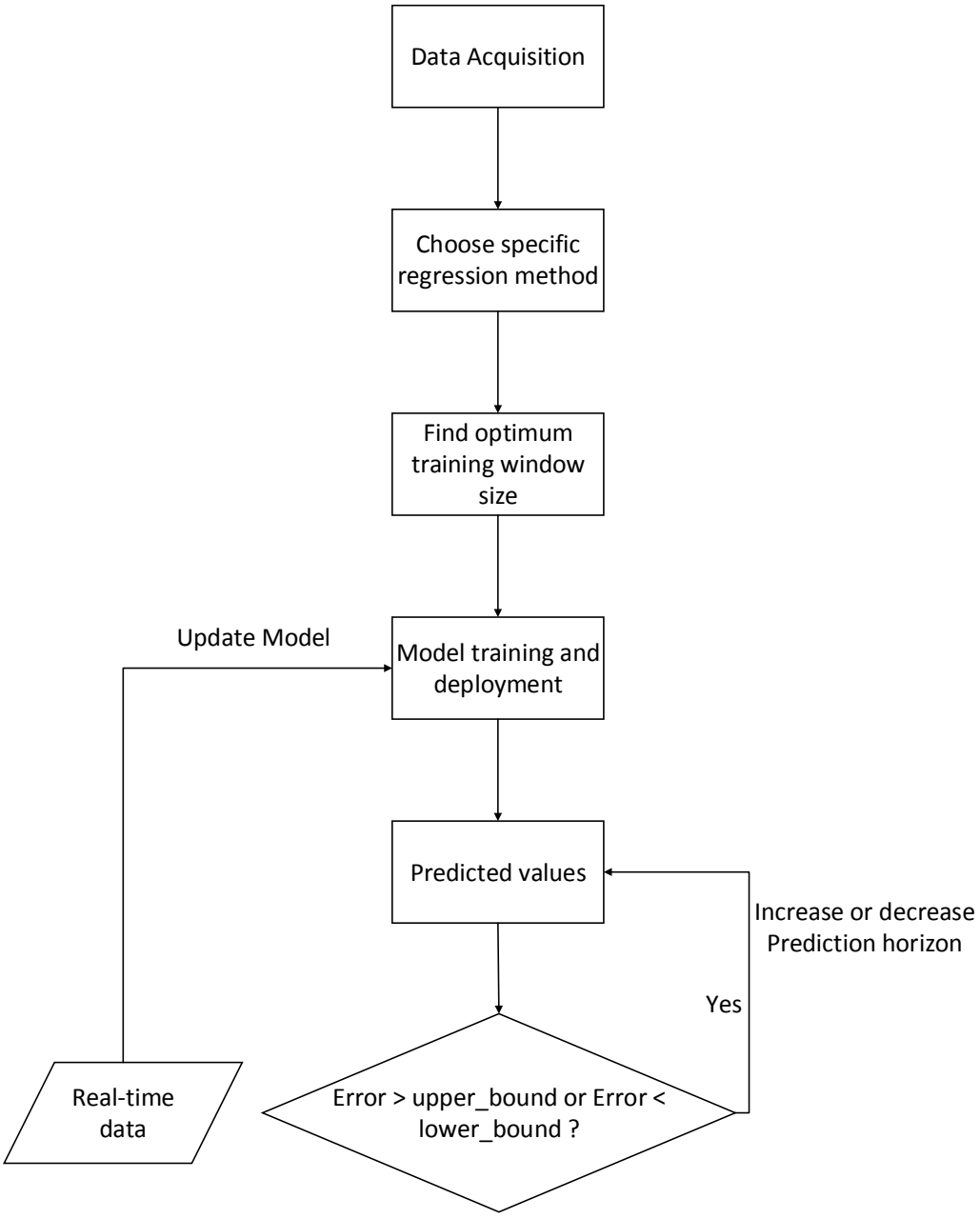


Figure 4.2: Flowchart for Adaptive Moving Window Regression

Traditionally, statistical methods like auto regressive moving average (ARMA) and auto regressive integrated moving average (ARIMA) [4] were used for time series regression. However, recently the trend has shifted towards more sophisticated ML models such as different variants of support vector regression (SVR) and artificial neural networks (ANN) because of their robustness and ability to provide more accurate solutions [69].

The proposed approach was implemented using SVR due to its ability to model non-linear data using kernel functions. SVR is an extension of SVM which is widely used for regression analysis [29]. The main idea is the same as in SVM, it maps the training data into higher feature space using kernel functions and find the optimum function which fits the training data using hyper-plane in higher dimension.

Methods based on SVR often provide more accurate models as their counterpart regression algorithms as demonstrated by various research efforts [91] [92]. It provides higher accuracy at the expense of additional complexity [7]. However, as in the proposed algorithm a small training window is used, the added complexity is almost negligible for such small datasets. The performance of several variants of SVR was compared and finally SVR with radial basis function (RBF) kernel was chosen as underlying regression algorithm.

4.2.1.2 Optimum Training Window Size The choice of the optimum training window size for ML models is an open research issue[69]. In general, the accuracy of prediction model increases as the size of training data increases which reflects to have large historical data for training prediction models so that it covers all possible patterns spanning time series. Although this approach generates generic and accurate model for prediction in most cases, there is one major drawback associated with it which was mentioned earlier as well: If the behavior or statistics of the underlying data changes, trained model is unable to track the changes and result into erroneous readings and the error will start accumulating in future predictions.

In contrast to this approach, researchers have proposed to use the moving window for training the ML model in which most recent data is fed to the models [40] [9]. The size of the optimum window is a challenging task with no generic solution. A large window size can have more accurate results but it increases the complexity of the model making it unsuitable for real-time applications whereas a small window size can result into an increased error and hence effecting the reliability of the overall system [74].

In order to overcome this issue, a novel and generic method based on time series analysis (Lomb Scargle method) to find the optimum window size was proposed in this

contribution and results were validated using real-world data. The proposed method exploits the inherent periodic nature of most of the real world time series data. Let's consider a simple example of a temperature data generated by a sensor deployed in Rio de Janeiro, Brazil [71] as shown in the Figure 4.3; although the pattern formed by the temperature data is very irregular, a repeating pattern in the temperature readings can be observed after every twenty four hours. This work exploits the fact that if the used training window is equal to the inherent periodicity of the data, it will learn all the local patterns and would be able to predict more accurately. It should be noted that the proposed approach does not assume the underlying data as periodic but instead looks for the highest periodic component.

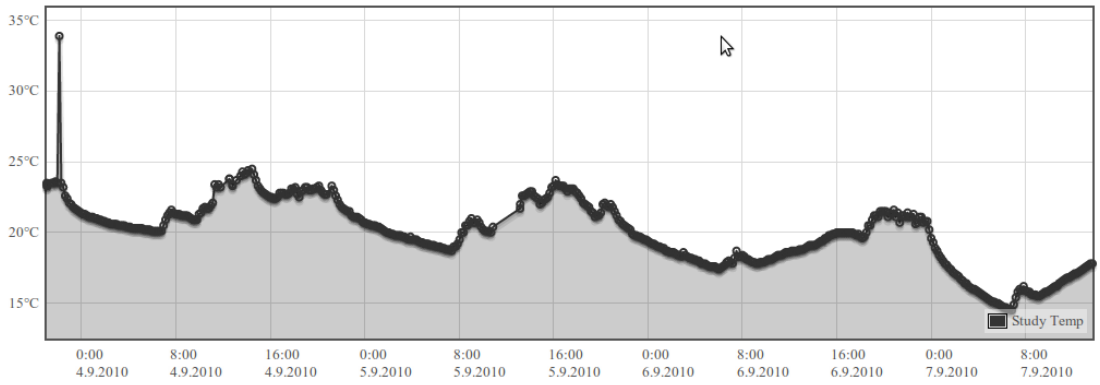


Figure 4.3: Temperature data [71]

A fast Fourier transform (FFT) algorithm is the most commonly used method for finding periodicity by searching for the sharp peaks in the periodogram calculated by Fourier transform of the time series. FFT requires the time series to be evenly spaced which is not always possible for most of the IoT data. Missing values is a common phenomenon in IoT and the inability of FFT to deal with it makes it unsuitable for our system. For such systems, another method called least-squares spectral analysis (LSSA) or more commonly known as Lomb Scargle can be used to find the highest periodic component in a time series data. Lomb first proposed the method while studying variable stars in astronomy [63]. Instead of taking simple dot products of the data with sine and cosine waveforms like in Fourier transform, Lomb Scargle method modified it to find a time delay τ which makes the pair of sinusoids mutually orthogonal at sample time t_n and also adjusted for the potentially unequal powers of these two basis

functions, to obtain a better estimate of the power at given frequency. Mathematically, it can be represented by following equations:

$$P_X(f) = \frac{1}{2\sigma^2}\Bigg\{\frac{\Big[\sum_{n=1}^{N}(x(t_n)-\overline{x})cos(2\pi f(t_n-\tau))\Big]^2}{\sum_{n=1}^{N}cos^2(2\pi f(t_n-\tau))} + \frac{\Big[\sum_{n=1}^{N}(x(t_n)-\overline{x})sin(2\pi f(t_n-\tau))\Big]^2}{\sum_{n=1}^{N}sin^2(2\pi f(t_n-\tau))}\Bigg\} \tag{4.1}$$

where $\overline{x}$ is the mean of the data, σ^2 represents variance of the data, t represents the time and the value of τ is defined as

$$tan(4\pi f\tau) = \frac{\Big(\sum_{n=1}^{N}sin(4\pi f t_n)\Big)}{\Big(\sum_{n=1}^{N}cos(4\pi f t_n)\Big)} \tag{4.2}$$

4.2.1.3 Adaptive Prediction Window This work also proposes to have an adaptive size for prediction window or more commonly known as prediction horizon in order to ensure a certain level of accuracy. The intuition behind it is to increase the size of prediction window if the accuracy of model is high and decrease it if the performance of the prediction model decreases. The performance of the model is evaluated by comparing the predicted data with actual data when it arrives. Algorithm 1 shows the approach for adaptive prediction window.

4.2.2 Error Propagation in CEP

The output of the prediction block is forwarded to CEP in the form of an event tuple through Kafka where CEP engine applies pre-defined rules in order to detect the complex event. According to [70], an abstract event tuple can be defined as $e = \langle s, t\rangle$ where e represents an event, s refers to a list of content attributes and t is a time stamp attached to an event. In the proposed architecture, every predicted event is accompanied

Algorithm 1 Adaptive Prediction Window Size

```
1: function PredictionWindow(y_act, y_pred)
2:     MAPE = mean(abs((y_act − y_pred)/y_act) ∗ 100)
3:     if MAPE > upperbound then
4:         PredictionWindow = PredictionWindow − 1
5:     else if MAPE < lowerbound then
6:         PredictionWindow = PredictionWindow + 1
7:     else
8:         PredictionWindow = PredictionWindow
9:     end if
10:    return PredictionWindow
11: end function
```

by prediction error, and CEP correlates these events using different rules. Prediction error introduces uncertainty in the complex events effecting the reliability of the system. In order to take prediction error into account, a probabilistic event processing approach was adopted for defining event tuples as mentioned in [18] encapsulating prediction error with attribute's value in event tuple as $e = \langle s = \{attr = val, pdf(\mu_1, \sigma_1)\}, t\rangle$ where *pdf* shows the probability density function of prediction error.

This section demonstrates how this error is propagated through the system while applying different CEP pattern matching rules.

4.2.2.1 Filtering Event filtering is the most basic functionality which supports other more complex patterns. Not every event is of interest for the consumers and a user might be interested in only specific events. Let's consider a simple example of a temperature sensor which generates a reading every second; If a user is only interested in temperatures higher than a specified threshold, it is defined in the filter. Then only if the conditions become true, the event would be published to the observer. Typical conditions include *equals* or *greater/less than* etc.

In such cases, when the rule is applied to a single event then the total error associated

with the complex event will be the probability of prediction error for that particular data source. For example, for a rule which generates an event if the data stream A_1 is less then threshold, the resulting error will be;

$$pdf_E(total) = pdf_E(A_1) \tag{4.3}$$

where *pdf* represents probability density function.

4.2.2.2 Joins The functionality of Joins is to correlate events from different data streams using simple logical operations such as conjunction *(and)* and disjunction *(or)*.

In case of conjunction, the probability of error associated with individual data stream will be multiplied with each other. For example if there are two events A_1 and A_2, and a complex event is defined as if A_1 is greater than threshold and A_2 is less then threshold; overall error is given by:

$$pdf_E(total) = pdf_E(A_1) * pdf_E(A_2) \tag{4.4}$$

Disjunction operator (or) is used to trigger a complex event when either of multiple conditions become true. For example, if a rule is defined as if A_1 is greater then threshold or A_2 is greater the threshold, generate a complex event. In such scenario, total error will be equal to the highest of the individual data stream prediction error as shown below;

$$pdf_E(total) = max(pdf_E(A_1), pdf_E(A_2)) \tag{4.5}$$

4.2.2.3 Windows Windows provide a tool to extract temporal patterns from the incoming events to infer a complex event. The two most basic type of windows are:
a) Time Window: It enables to define a time window to extract events lying only in that window. The temporal relation between different events plays an important role in evaluating complex event. For example five degree centigrade temperature change in a room in one hour will have different meaning as compared to the same temperature

change in one minute. The former observation can be resulting from the heater being switched on and later might be caused by a fire. The time window can be a fixed time window or a sliding time window. Simple arithmetic tasks like finding maximum, minimum or aggregated value also require the definition of time window.

b) Tuple Window: In contrast to time window, tuple window acts on the number of events defined. Aggregation of every five samples is a typical example of tuple window operation.

For both cases, total error will be the product of prediction error for the events falling in the window. For example; if a rule is defined as to generate a complex event if event A_1 is greater than threshold and A_2 is greater than threshold for n consecutive readings. In such case, the probability of total Error will be given by:

$$pdf_E(total) = n * (pdf_E(A_1) * pdf_E(A_2)) \tag{4.6}$$

Equations 4.3-4.9 show relative simple examples of applying different rules. In practice, CEP rules can be more complex by combining all these functionalities but overall error can be calculated by simply replacing the functionality with the above equations. Now, if the expression for probability of prediction error for individual data stream is available, it can be replaced in the equations 4.3-4.5 in order to calculate the expression for overall error.

4.3 Experimental Evaluation

This section presents the evaluation of the proposed method with the help of ITS use-case scenario.

4.3.1 Dataset and Experimental Setup

In order to evaluate the proposed method, traffic data provided by Madrid city is used. The Madrid city has deployed hundreds of traffic sensors at fixed locations around the city for measuring several traffic features including average traffic intensity (number of

vehicles per hour) and average traffic speed. Madrid city council publishes this data as a RESTful service[1]in xml format. This data needs then to be analysed automatically in near real-time in order to detect traffic patterns and to generate complex events such as bad traffic or a congestion. Esper CEP[2] provides the optimum solution for this scenario as it provides the capability to analyse this streaming data on the fly. Esper rules can be configured using EPL for pattern recognition in order to generate complex events.

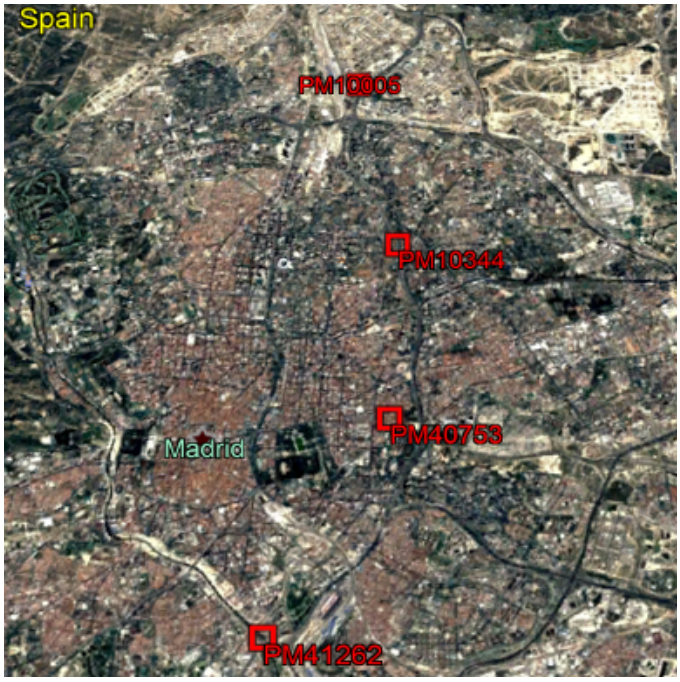


Figure 4.4: Selected locations for analytics

Four different locations were chosen from city of Madrid to get fair analysis of results as shown in the Figure 4.4. Table 4.1 shows the location IDs and coordinates for the selected locations. Data is collected from the RESTful service for over three months. It consist of different traffic features including traffic speed, traffic intensity, occupancy and type of vehicles. Madrid city council updates the aggregated data every five minutes.

Pseudo code of one simple rule for inferring complex event (bad traffic or congestion)

[1]http://informo.munimadrid.es/informo/tmadrid/pm.xml
[2]http://www.espertech.com/

Table 4.1: Selected data locations

Location No.	Location ID	Coordinate
1	PM10005	(40.29, -3.40)
2	PM10344	(40.27, -3.39)
3	PM40753	(40.25, -3.39)
4	PM41262	(40.22, 3.41)

is described in the algorithm 2. In this example, traffic speed and traffic intensity data streams are taken as inputs. A more complex rule may involve other data sources like weather forecast or social media data. CEP generates a complex event when the average traffic speed and average traffic flow is less than the threshold values for 3 consecutive readings. Now if the input is predicted data as in the proposed approach, the complex event detected will also be in the future and traffic administrators can take precautionary measures in order to avoid congestion. This is just one example that demonstrates how CEP rules can be exploited to find more complex events.

Algorithm 2 Example Rule for CEP

1: **for** $(speed, intensity) \in TupleWindow(3)$ **do**
2: **if** $(speed(t) < speed_{thr}$ and $intensity(t) < intensity_{thr}$ AND
3: $speed(t+1) < speed(t)$ and $intensity(t+1) < intensity(t)$ AND
4: $speed(t+2) < speed(t+1)$ and $intensity(t+2) < intensity(t+1))$ **then**
5: Generate complex event Congestion
6: **end if**
7: **end for**

4.3.2 Evaluation

As described in section 4.2, the proposed Lomb Scargle method was applied for finding optimum window size for training ML model. The resulting periodogram from one location for traffic speed is shown in the Figure 4.5. X-axis shows the frequency of data

on per sample basis and y-axis represents the power spectral density. Data periodogram has a peak at 0.066 which corresponds to a window size of 15 samples.

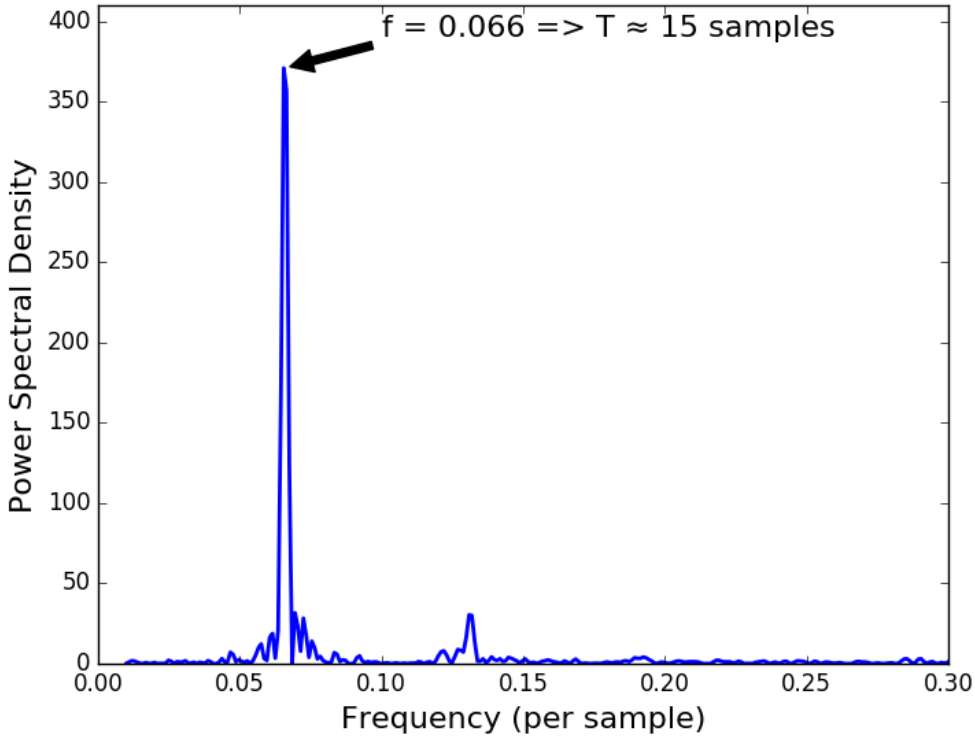
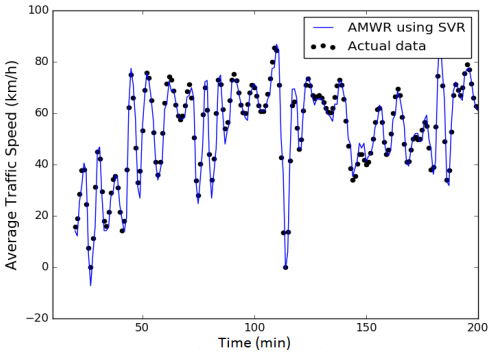


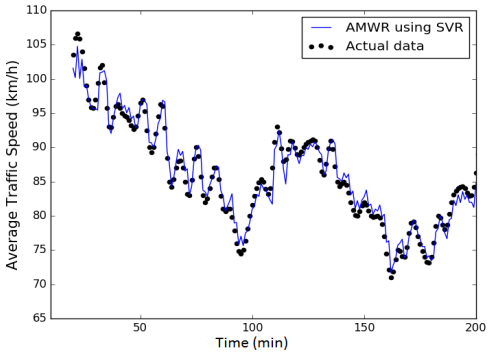
Figure 4.5: Periodogram for traffic speed data for location 1

Table 4.2: Error accumulated over 1 month period with different training window sizes for locations 1

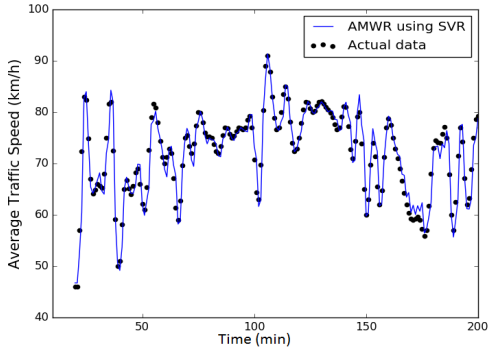
No.	Training window (No. of samples)	MAPE(%)
1	5	17.67
2	10	15.48
3	**15**	**14.36**
4	20	14.63
5	25	15.01
6	30	14.38
7	35	14.86
8	40	15.06
9	45	14.79
10	50	14.72



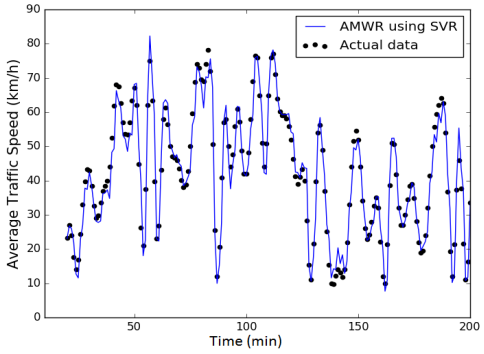
(a) Location 1 (MAPE=9.59 %)



(b) Location 2 (MAPE=3.69 %)

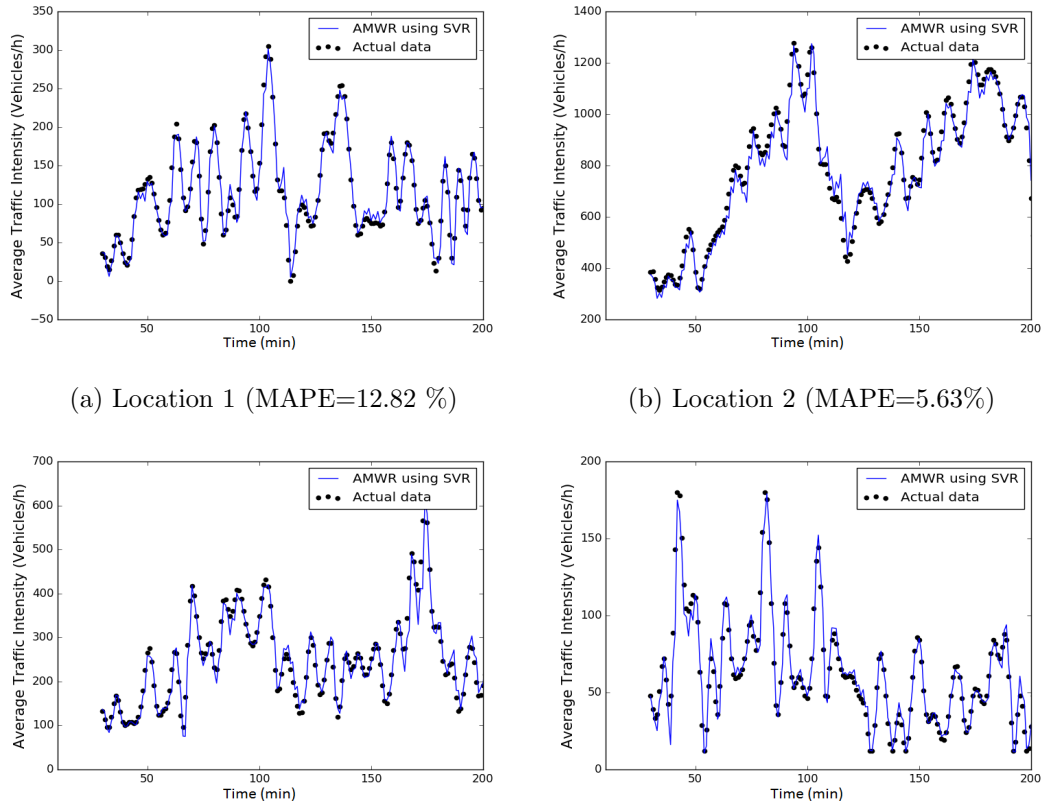


(c) Location 3 (MAPE=3.24 %)



(d) Location 4 (MAPE=11.29 %)

Figure 4.6: Prediction results on average traffic speed data from four different locations



(a) Location 1 (MAPE=12.82 %)

(b) Location 2 (MAPE=5.63%)

(c) Location 3 (MAPE=8.19%)

(d) Location 4 (MAPE=11.66%)

Figure 4.7: Prediction results on average traffic intensity data from four different locations

This work validated the proposed method by using different window sizes for prediction on one month of traffic data and noting corresponding mean absolute percentage error (MAPE) . MAPE is an accurate metric for evaluating the performance of prediction models and can be calculated as:

$$MAPE(\%) = 1/n \sum_{t=1}^{n} |(\frac{Y_t - Y_t^{'}}{Y_t})| \times 100 \tag{4.7}$$

where Y_t represents actual data, $Y_t^{'}$ represents predicted data and n represents the total number of predicted values. Table 4.2 shows the MAPE(%) for different window sizes and it can be seen that error is minimum when the window size of 15 samples or multiple of 15 is used which validates our approach.

Prediction results for traffic speed and traffic intensity data for all four locations is shown in Figure 4.6 and Figure 4.7. As it can be seen, that predicted values are tracking the actual data quite accurately. The reason behind it is that if there is an error in the predictions, it is incorporated and the model is updated accordingly and hence it prevents the error from propagating.

4.3.3 Comparison with Benchmark Solutions

In order to compare the performance of AMWR with other regression models, several state-of-the-art regression algorithms were implemented using python machine learning library scikit-learn [60]. Figure 4.8 and 4.9 shows the MAPE (%) plot of different regression algorithms implemented for all four locations , and Table 4.3 and 4.4 shows the results in numerical form. As it can be seen, AMWR outperforms other regression algorithms as it tracks the incoming data stream accurately. There are two main reasons for high accuracy provided by AMWR. 1) As new data arrive; AMWR takes the prediction error into account and retrain the model using more recent data. As the size of training window is very small, it is able to retrain and predict in near real-time. 2) It tracks the error and if prediction error starts to increase, it decreases the size of prediction window in order to maintain the level of accuracy.

MAPE accumulated over a period of time is dependent on the data characteristics of underlying sensing location. If the data has more variations with a high value of standard deviation, the resulting accuracy on prediction readings will be lower. Table 4.5 shows the input standard deviation for traffic speed and traffic intensity for all locations. Location 1 and location 4 have high standard deviation and hence high variance which resulted into more error as the spread of data points is effecting the predictions.

To further validate the performance of our prediction algorithm, a congestion point was captured in the input data for location 1. Figure 4.10 shows the comparison of AMWR based SVR with conventional SVR for predicting incoming data. As it can be seen in the figure that there is a congestion point after 170^{th} minute when traffic speed suddenly drops to 0. A conventional SVR regression algorithm is unable to track

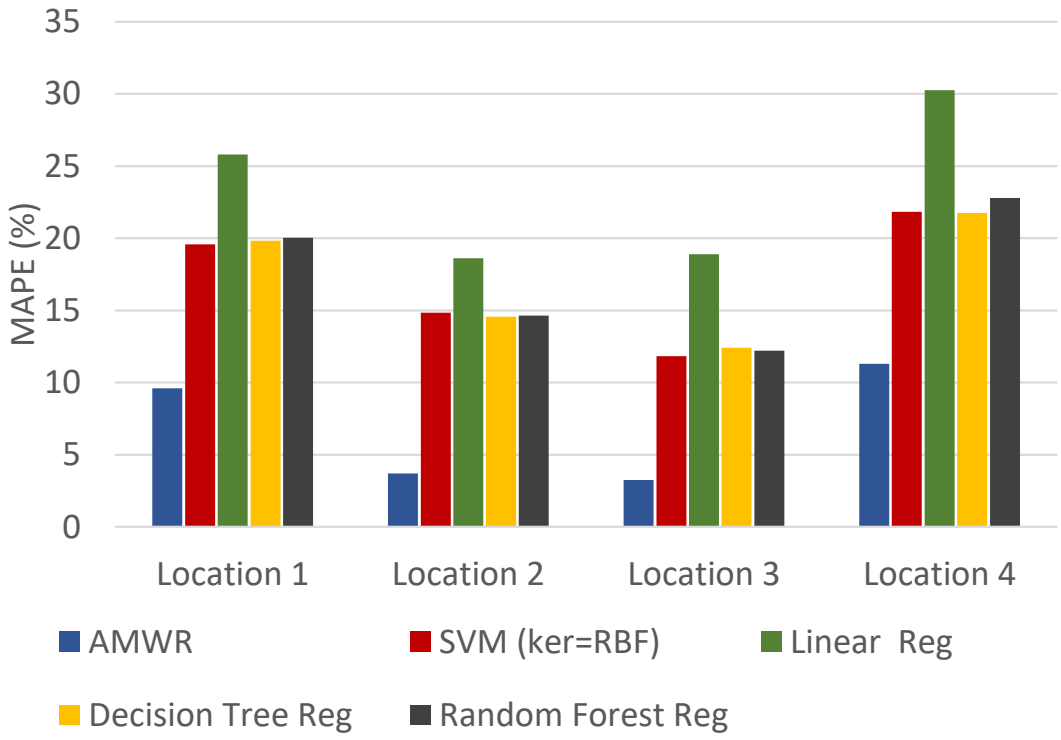


Figure 4.8: AMWR comparison with different models for traffic speed data

the data whereas AMWR based SVR tracks the actual data accurately and hence captures the congestion point. Finally, once the predicted values for traffic speed and traffic intensity are calculated, CEP can be used to infer traffic state using the rules mentioned in algorithm 2. Event detected by CEP will be in future providing enough time for traffic administrators to manage traffic pro-actively and avoiding congestion.

Table 4.3: MAPE(%) for traffic speed data

Method	location 1	location 2	location 3	location 4
AMWR	9.59	3.69	3.24	11.29
SVM-RBF	19.57	14.83	11.84	21.84
Linear Reg	25.80	18.61	18.90	30.25
Decision Tree Reg	19.82	14.57	12.40	21.76
Random For Reg	20.04	14.63	12.22	22.80

Overall error propagated through the system depends on the underlying CEP rule. As

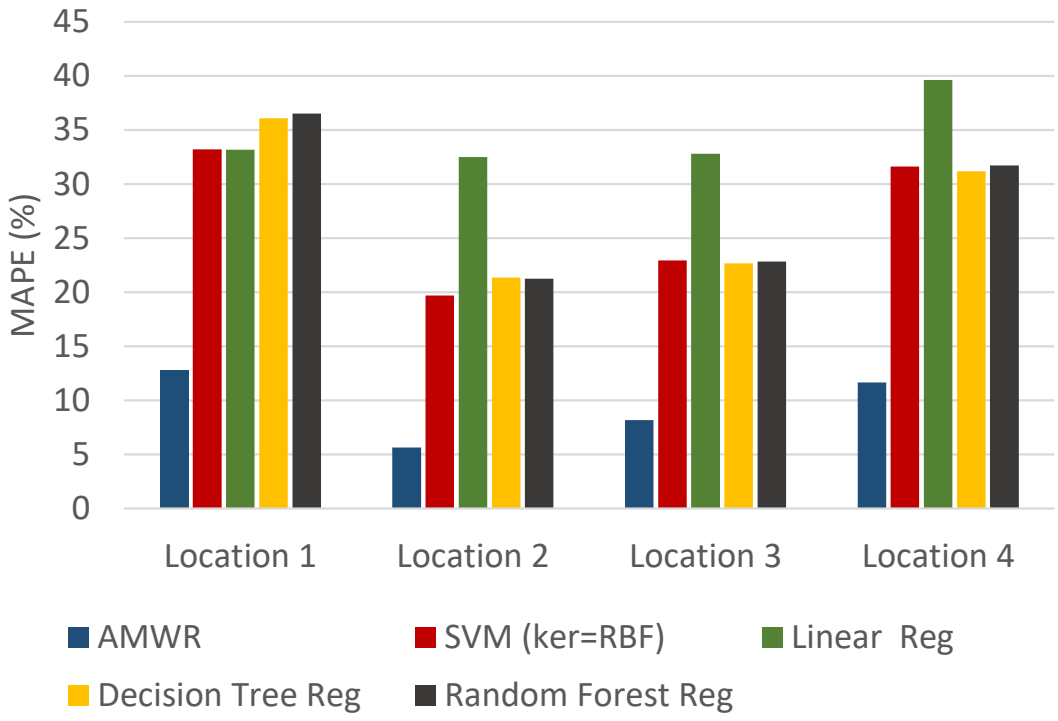


Figure 4.9: AMWR comparison with different models for traffic intensity data

Table 4.4: MAPE(%) for traffic intensity data

Method	location 1	location 2	location 3	location 4
AMWR	12.82	5.63	8.19	11.66
SVM-RBF	33.20	19.70	22.94	31.61
Linear Reg	33.16	32.50	32.81	39.63
Decision Tree Reg	36.06	21.35	22.68	31.19
Random For Reg	36.53	21.25	22.85	31.73

an example, consider the rule mentioned in algorithm 2. Equations 4.3-4.5 can be used to calculate the expression for overall error as:

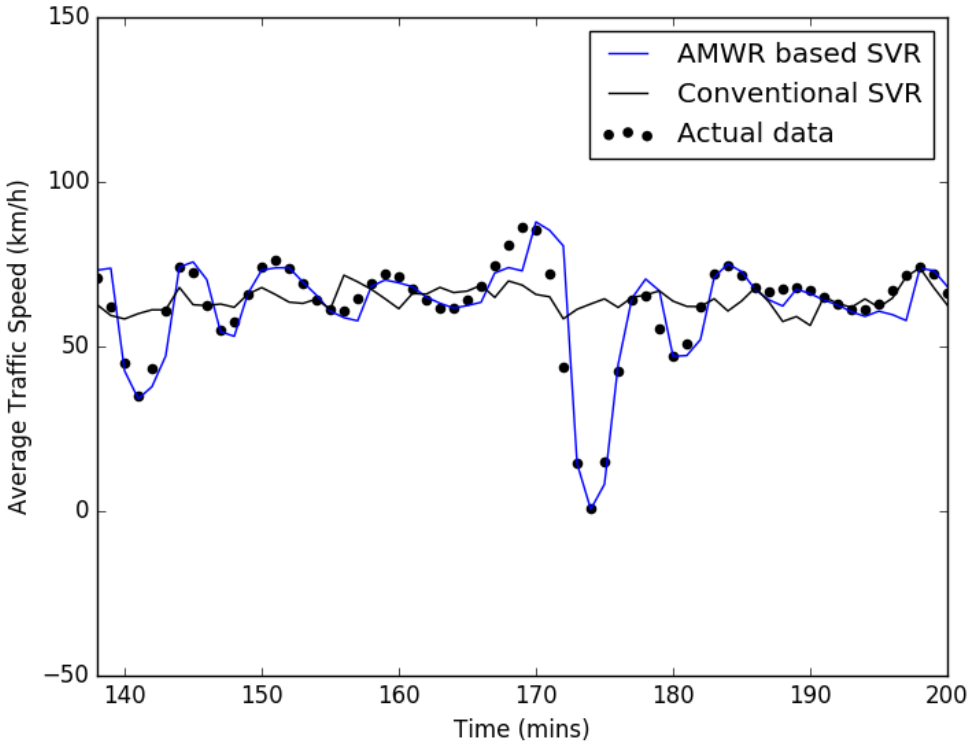


Figure 4.10: Capturing a congestion event

Table 4.5: Input standard deviation (σ)

Locations	traffic speed	traffic intensity
location 1	0.22	0.31
location 2	0.15	0.15
location 3	0.17	0.17
location 4	0.20	0.29

4.3.4 Error Modelling

In order to get the probability of density functions for predicted traffic speed and traffic intensity, prediction error was calculated for one month data and plotted in the form of an histogram. An histogram is a graphical representation of the data which provides an estimation of the probability distribution of the data. Error distribution for traffic

speed and traffic intensity for location 1 is shown in the Figure 4.11. Firstly, the Gaussian distribution was chosen to model the error propagation curve because of the bell shaped nature of the histogram as shown in Figure 4.11. The probability density function for the Gaussian distribution is as follows:

$$pdf_E(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{4.8}$$

where μ represents the mean and σ represents the standard deviation of the distribution. The Gaussian distribution was applied by using the curve fitting techniques and the parametric distribution for the error was found. The Gaussian distribution fitted well to model most of the data in error propagation histogram but it was ill-fitted for the outliers on the error histogram. Therefore a better fitted curve for error modelling was still needed.

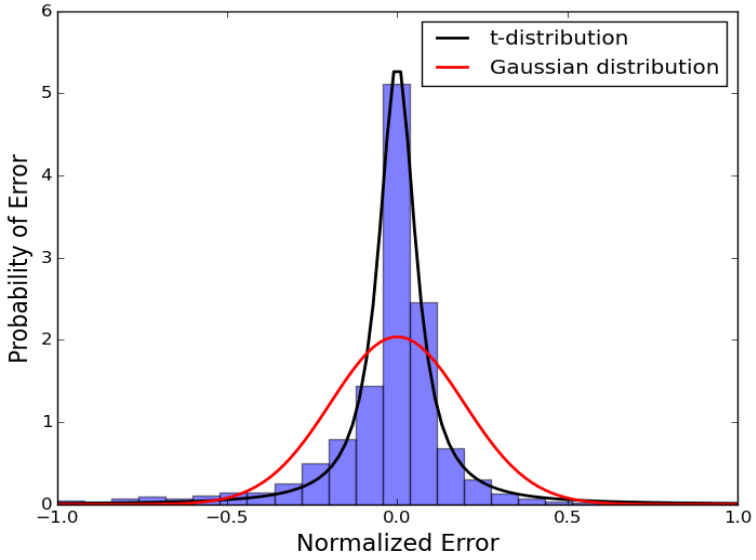
$$pdf_E(total) = n * (pdf_E(speed) * pdf_E(intensity)) \tag{4.9}$$

where n is the window length, $pdf_E(speed)$ is the probability of error for speed and $pdf_E(intensity)$ is probability of error for traffic intensity. If we have the generalized expressions for $pdf_E(speed)$ and $pdf_E(intensity)$, they can be substituted in equation 4.9 to get an expression for the overall error of the system.

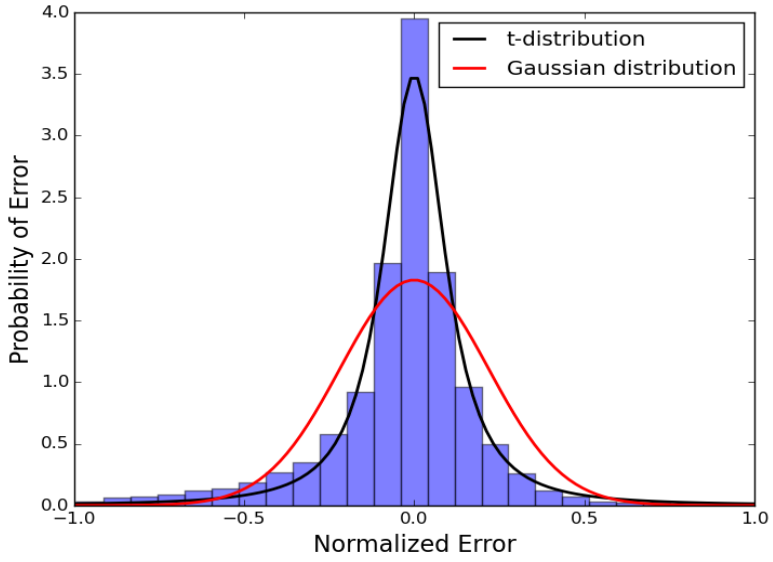
Table 4.6: Error distribution parameters for traffic intensity

No.	Gaussian distribution	t-distribution
location 1	$\mu = 0, \sigma = 0.27$	$\nu = 1.79, \mu = 0, \sigma = 0.13$
location 2	$\mu = 0, \sigma = 0.21$	$\nu = 1.90, \mu = 0, \sigma = 0.10$
location 3	$\mu = 0, \sigma = 0.22$	$\nu = 1.73, \mu = 0, \sigma = 0.10$
location 4	$\mu = 0, \sigma = 0.26$	$\nu = 1.67, \mu = 0, \sigma = 0.12$

The Student's t distribution [26] due to its heavy tail nature can better model the outliers. Also, the Student's t distribution has a degree of freedom parameter which can be changed to tune the shape of the distribution according to the nature of error



(a) Error distribution for traffic speed (location 1)



(b) Error distribution for traffic intensity (location 1)

Figure 4.11: Error modeling for location 1

Table 4.7: Error distribution parameters for traffic speed

No.	Gaussian distribution	t-distribution
location 1	$\mu = 0, \sigma = 0.24$	$\nu = 1.16, \mu = 0, \sigma = 0.08$
location 2	$\mu = 0, \sigma = 0.19$	$\nu = 0.07, \mu = 0, \sigma = 0.03$
location 3	$\mu = 0, \sigma = 0.20$	$\nu = 0.03, \mu = 0, \sigma = 0.07$
location 4	$\mu = 0, \sigma = 0.24$	$\nu = 0.45, \mu = 0, \sigma = 0.10$

propagation data. Hence the Student's t distribution can be used to further improve the modelling of error propagation data. The probability density function for Student's t-distribution is given by:

$$pdf_E(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}\Big(1 + \frac{x^2}{\nu}\Big)^{-\frac{\nu+1}{2}} \tag{4.10}$$

where ν is the degrees of freedom parameter and Γ is the gamma function. The degrees of freedom parameter ν can tune the variance and leptokurtic nature of the distribution. With decreasing ν, the tails of the distribution becomes heavier. More details about the t-distribution can be found in [38][65].

Table 4.6 and 4.7 shows the parameters for error distribution for both the Gaussian and Student's t distribution at all four locations.

4.4 Discussion

In this chapter, a method for predicting complex events for proactive IoT applications was proposed and implemented in order to address the first research challenge as highlighted in section 1.3. The proposed solution combines the power of real-time and historical data processing using CEP and ML respectively. This work highlights the common challenges for combining both technologies and implemented the proposed system by addressing those challenges. In this regard, a prediction algorithm called AMWR was proposed for near real-time data. In contrast to conventional methods, it utilizes a moving window for training ML model where the optimum size of training window is find automatically using the inherent periodicity of IoT data streams.

The proposed solution provides a generic predictive solution for different IoT applications. Different components involved can easily be configured according to requirements of a specific application. However, the optimum size for training window will be different for every data stream. And even though, many IoT data streams have some element of periodicity; nevertheless it is not always true. In such cases, the automatic method for finding optimised window size might not work efficiently and default size for training window will have to be defined. The default size will be dependent on the requirements of the scenario as large training window will take more time in training. The definition of real-time is scenario dependent, for instance in health monitoring scenario real-time is under 10 seconds whereas for ITS a delay of 1-2 minutes will still be counted as real-time.

This work demonstrates the proposed solution using data provided by Madrid city council which combines the data being generated from different sensors internally and provides a unified data stream as a web service. But for other applications, data might be generated by different systems and in such cases these data streams need to be combined and synchronized as part of data acquisition functionality.

Chapter 5

Context-Aware Event Processing

This chapter presents the approach towards context-aware rule generation as mentioned in chapter 3 as a second contribution of the thesis[1]. A brief introduction is given in section 5.1 followed by the problem definition in section 5.2. Section 5.3 describes the proposed novel method for context-aware event processing based on adaptive clustering and CEP. The performance of the proposed method is elaborated using traffic data provided by city of Madrid in section 5.4. Moreover, the proposed method is extended further to implement it for large-scale applications in section 5.5. Finally, the chapter is concluded by highlighting how the proposed approach solves the problem of context-aware event processing and provides an edge on existing technologies in section 5.7.

5.1 Introduction

In order to detect complex events, systems based on CEP require rules which have to be given manually by the system administrators, who are then expected to be skilled

[1]The results prsented in this chapter have been published in following two papers:

1) A. Akbar, F. Carrez, K. Moessner, J. Sancho and J. Rico "Context-aware stream processing for distributed iot applications," in *2015 IEEE World Forum on Internet of Things (WF-IoT),* pages 663-668, Dec 2015.

2) P. Ta-Shma, A. Akbar, G. Gerson-Golan, G. Hadash, F. Carrez and K. Moessner "An Ingestion and Analytics Architecture for IoT applied to Smart City Use Cases," *IEEE IoT Journal,* PP(99):1-1, 2017.

with required background knowledge; unfortunately sometimes it is neither available nor so precise.

In order to detect complex events, systems based on CEP deploy static rules and there is no means to update the rules automatically. These rules are defined manually by the system administrators, who are then expected to be skilled with required background knowledge; unfortunately sometimes it is neither available nor good enough. In real-time dynamic IoT applications, the context of the application is always changing and the performance of CEP will deteriorate in such scenarios. For example, a high temperature reading in summers will have different meaning as compared to the same reading in winters. Similarly, a high traffic density during the day is a normal event as compared to the same event happening late at night. Hence, a system based on CEP for extracting high-level knowledge should incorporate the current context into account and adapt its rules accordingly.

In contrast to existing approaches, a novel approach is proposed for finding optimized parameters for CEP rules automatically according to current context using machine learning methods. The proposed architecture is able to infer complex events from raw data streams in a scalable manner and provide adaptive solutions with respect to context at the same time. A real-world example of intelligent transportation system (ITS) is used to elaborate the approach.

In short, following contributions are made in this chapter:

- Propose and implement a novel method based on machine learning for calculating optimized parameters for CEP rules and update it according to the current context;
- Demonstration and validation of proposed method using real-world use-case of ITS;
- Propose an architecture for carrying analytics for large-scale IoT applications based on novel method and implemented it using open-source components optimized for big data applications.

5.2 Problem Definition

In this section, the problem of manual CEP rules and context-aware event processing is formalized with the help of a use-case of ITS. In the city of Madrid, thousands of heterogeneous traffic sensors have been deployed on different locations across the city. These sensors provide real-time information about the traffic flow in the city such as average traffic speed, average traffic intensity, type of traffic or type of road etc. CEP has the potential to provide a real-time solution for analysing, correlating and inferring high-level knowledge from these data streams. The core of the CEP is a rule-based engine which requires rules for extracting complex patterns. These rules are based on different threshold values. For example, traffic speed can be analysed using a simple rule as "if current speed is less than threshold speed then generate a *slow speed* event". These rules will be defined by traffic experts based on their prior knowledge about the traffic patterns which makes the system dependent on them. Secondly, every road segment has a different response. There might be a road segment with speed restrictions due to road works as compared to a free flow road segment so the definition of bad traffic will be different for both road segments. In this way, at city level there will be hundreds of road segments and it is almost impossible for the administrators to understand the behaviour of individual road segment.

In short, following drawbacks have been identified in current technologies based on CEP (more details can be found in section 2.4.2:

- Threshold values have to be set manually and there is no automatic way to find the optimal threshold values.
- Threshold values set are static and once set, CEP system is unable to update it during run-time.
- Current solutions are not context-aware and adaptive by nature.

5.3 Proposed Framework

This work proposes to exploit historical data in order to find optimized threshold values for CEP rules automatically. The proposed approach is based on clustering analysis which is an unsupervised machine learning method to group the underlying data into different events and the boundaries separating the events serve as the threshold values for CEP rules. A block diagram of the proposed solution is shown in the Figure 5.1.

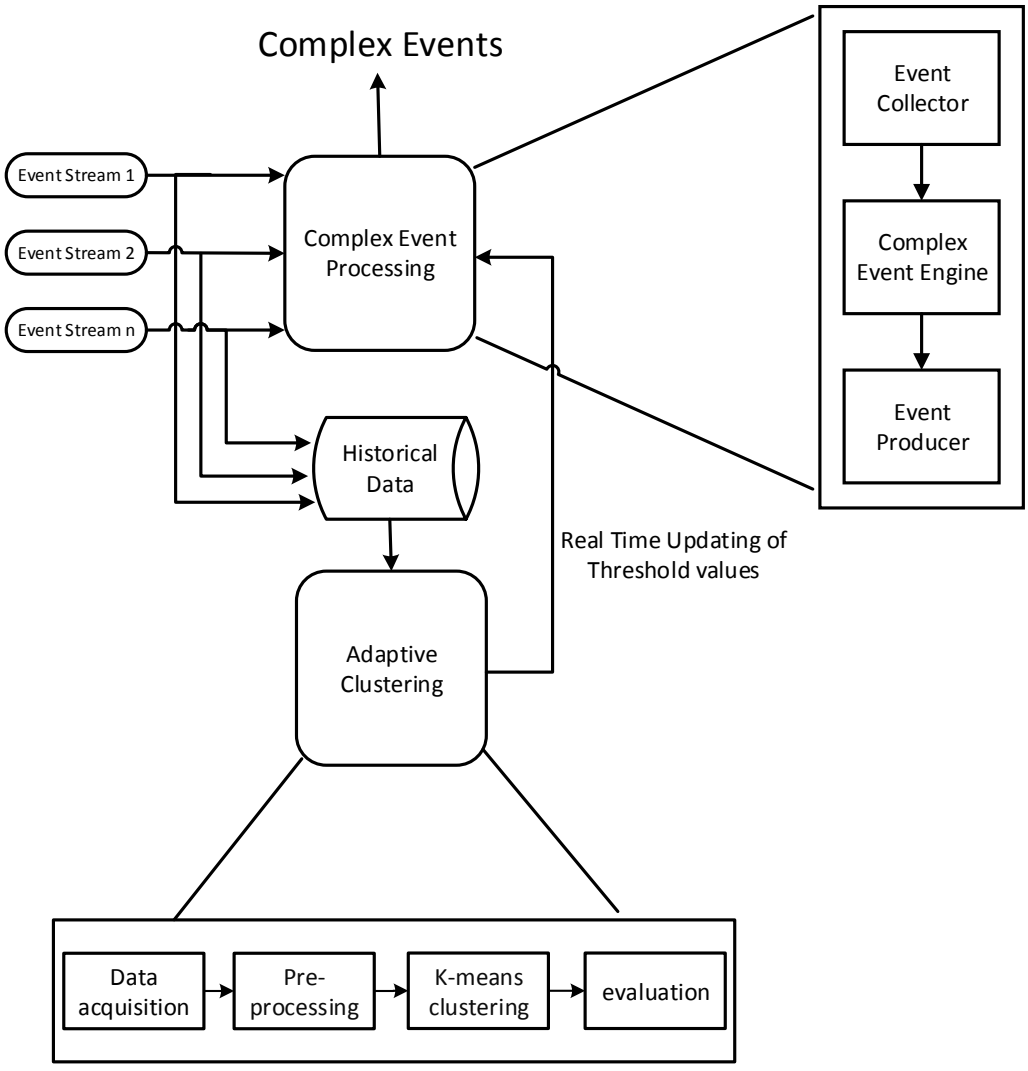


Figure 5.1: Proposed framework and block diagram

5.3.1 Adaptive Clustering

Clustering refers to the unsupervised machine learning method which is used for grouping similar objects on the basis of predefined metric such as distance or density. Clustering is a general technique used widely for knowledge discovery in big data sets and several variants of algorithms are found in the literature. Clustering separates the data into different possible events and group the data points from the same event together. The application of clustering is diverse and can be found in vast range of fields such as detecting credit card fraud from online transaction data [8] or detecting abnormalities from brain images data for medical research purposes [10].

In this work, the application of clustering is explored to find the boundaries between different events which can serve as threshold values for CEP rules. Figure 5.2 shows the flowchart of the proposed solution for adaptive clustering. In real-time scenarios, the definition of context changes with time and location. In our problem of ITS, traffic intensity is usually more in the morning hours with low average speed and hence the definition of bad traffic is different from the night hours when traffic is relatively smoother. The response of traffic is different during different time periods at every location and hence it is proposed to have different threshold values. This work time sliced the historical data in terms of morning, afternoon, evening and night traffic hours and extract the data only for specific time period. More details are described in the next section.

After extracting the data, feature scaling [52] is applied in order to optimize the clustering algorithm. Feature scaling is a method to bring all the features on the same scale so they contribute equally to the clustering algorithm. The range of values of different features of traffic data are on different scale. If one of the feature has considerably wider range as compared to other features, the optimization function for clustering will be governed by that particular feature and will impact the boundaries. The general expression for feature scaling using standardization is,

$$X_1' = \frac{X_1 - \mu_1}{\sigma_1} \tag{5.1}$$

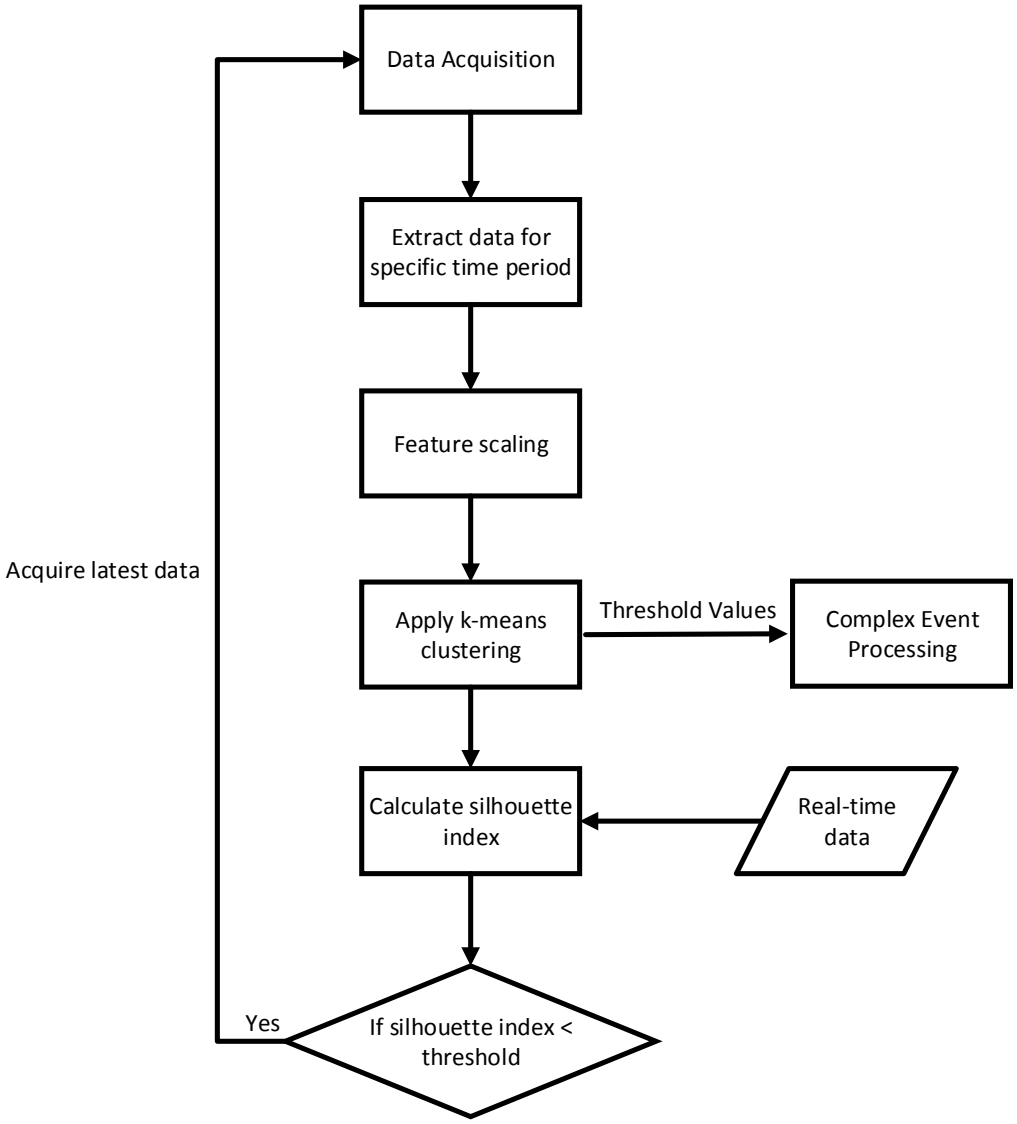


Figure 5.2: Flowchart for adaptive clustering

where X_1' is the new feature vector after scaling, X_1 is the initial feature vector, μ_1 is the mean value of feature vector and σ_1 is the variance of feature vector.

K-means clustering algorithm was implemented which is an iterative algorithm which forms the clusters by finding centroids such that the sum of the squares of distance from centroids to data is minimized [34]. For a data set X with n number of samples, it divides them into k number of disjoint clusters, each described by the centroid value.

In the first initialization step, it assigns initial centroids, most commonly by selecting k number of samples randomly from the data set X. In a second step it assigns each sample to its nearest centroids and forms k clusters. In the third and final step, it creates new centroids by calculating the mean value of the samples assigned to each previous centroid and calculates the difference between the old and new centroids. It keeps on iterating the process until the algorithm converges. K-means clustering was applied with $k = 2$, hence it resulted into two clusters. The reason for choosing $k = 2$ is that we are interested in two types of traffic events, i.e. good and bad traffic and the CEP rules were written accordingly. The third traffic can be introduced as normal traffic and in that case the value of k will be 3. The midpoint between the centroids divides the data into different events. These midpoints are proposed to use as threshold values for CEP rules as it defines the boundary between different events.

In general, threshold values for CEP rules are static and is a major drawback when deployed in real-world dynamic environments. Statistical properties of the underlying data may change over time resulting in inaccurate threshold values. Therefore, it is proposed to keep track of the changes in data distribution by assessing the quality of cluster as new data arrives. And as the quality of clusters deteriorate, k-means model is retrained and new threshold values are found using latest data.

Silhouette index $s(i)$ [61] is used to access the quality of clusters which is defined as

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{5.2}$$

where $a(i)$ is the mean intra cluster distance, and $b(i)$ is the mean nearest-cluster distance i.e. distance with the nearest cluster center which the data is not part of. $s(i)$ ranges from -1 to 1 where 1 is the highest score indicating perfect cluster quality and -1 as the lowest for cluster quality. As new data arrive, silhouette index is calculated using the cluster centroids and if $s(i) < threshold$, it acquires the latest data and repeat all the steps. In this way, the algorithm can track the changes in the underlying data. As an example, if the rain effects the traffic system, new data values will fit poorly to the old clusters and the quality of clusters will decrease. As the algorithm detects it, it will use the latest data to find new threshold values. In this research, the threshold

Table 5.1: Threshold values update for location 1 (weekdays)

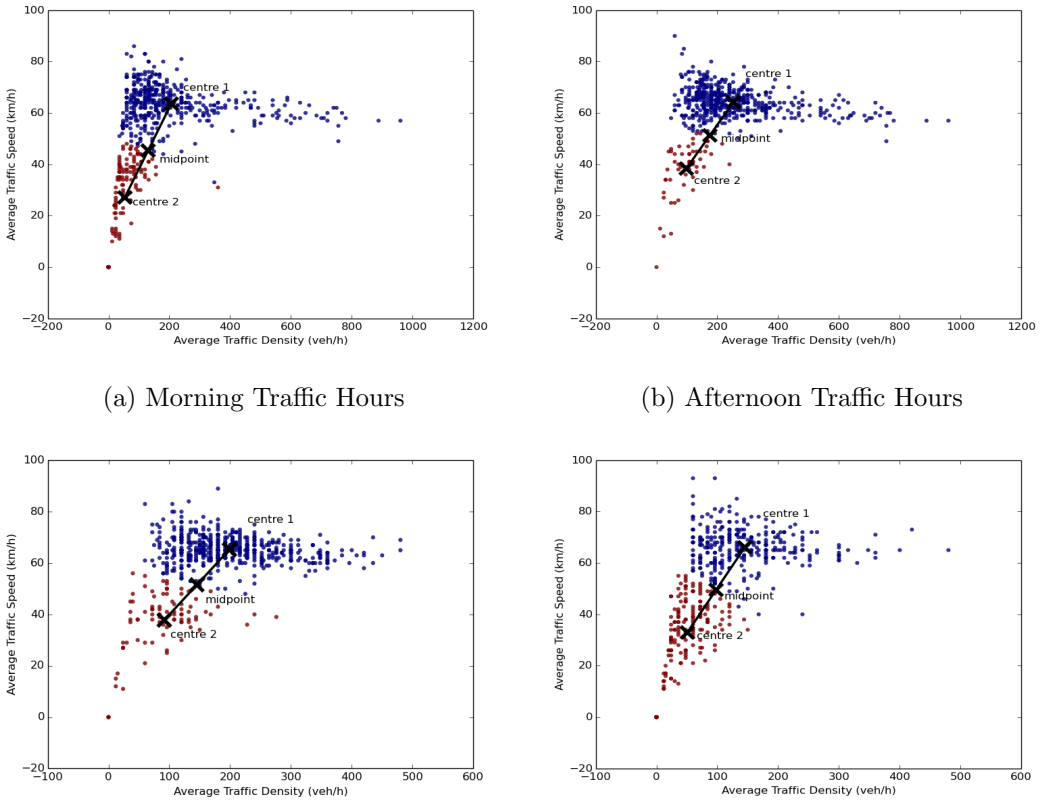
Traffic Period	Time Range	Threshold Values	Silhouette index
Morning	8 am to 12 pm	130 veh/h, 43 km/h	0.51
Afternoon	12 pm to 4 pm	175 veh/h, 51km/h	0.57
Evening	4 pm to 8 pm	145 veh/h, 49km/h	0.55
Night	8pm to 12 am	96veh/h, 48 km/h	0.50

value of 0.5 was used for silhouette index. The underlying data might not be perfect fit and a high value of silhouette index can result into running the algorithm infinitely (as it can be seen from flowchart that if silhouette index is less than threshold, it will repeat all the steps. The resulting clusters might not fit the criteria of high silhouette index and will keep on repeating). The value of 0.5 was selected after running several experiments on historical data and averaging the resulting silhouette indexes.

5.4 Experimental Evaluation

This section presents the results and evaluation of the proposed method. Dataset used for the evaluation of this contribution is the same as used in the first contribution and described earlier in section 4.3.1. Historical data is divided into four traffic periods depending on the time context as shown in the Table 5.1 and 5.2 and k-means clustering was applied with k=2. Figure 5.3 and 5.4 show the clustering results for individual traffic period. It results into two clusters as can be seen in the figures below. Blue cluster represents high average traffic speed and high traffic intensity indicating smooth traffic flow or good traffic state. Whereas red cluster represents traffic points with low average traffic speed and low traffic intensity which represents bad traffic state. Midpoint between both cluster centers represents the boundary separating both states and we used this boundary to define threshold values for detecting complex Events. Once new threshold values are found, they are passed to the CEP engine using message broker.

Once threshold values are calculated, CEP rules can be used to detect a complex Event.



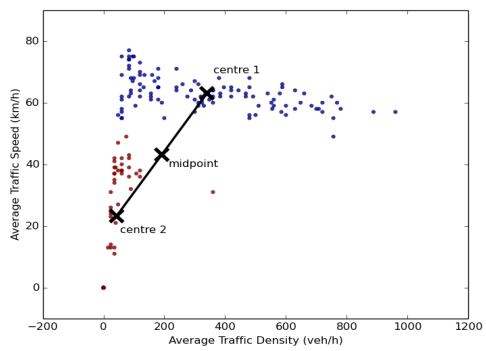
(a) Morning Traffic Hours

(b) Afternoon Traffic Hours

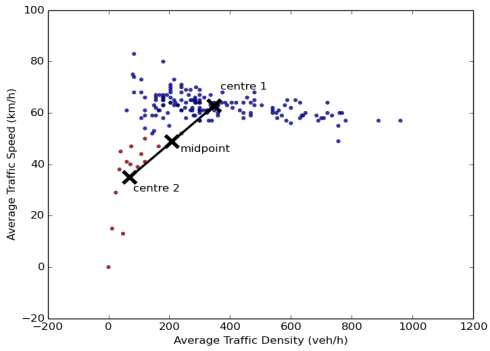
(c) Evening Traffic Hours

(d) Night Traffic Hours

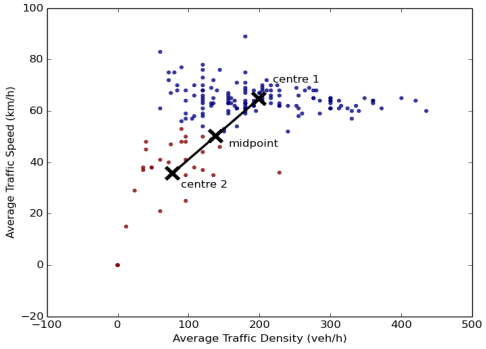
Figure 5.3: Threshold values for weekdays



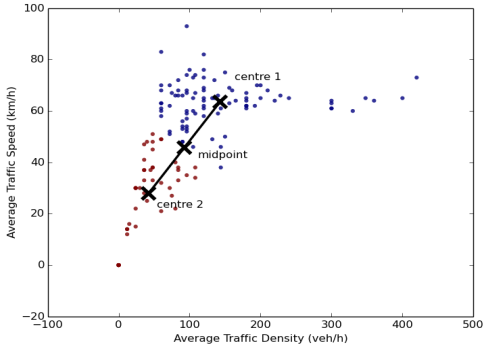
(a) Morning Traffic Hours



(b) Afternoon Traffic Hours



(c) Evening Traffic Hours



(d) Night Traffic Hours

Figure 5.4: Threshold values for weekends

Table 5.2: Threshold values update for location 1(weekends)

Traffic Period	Time Range	Threshold Values	Silhouette index
Morning	8 am to 12 pm	206 veh/h, 43 km/h	0.62
Afternoon	12 pm to 4 pm	235 veh/h, 45 km/h	0.54
Evening	4 pm to 8 pm	149 veh/h, 48 km/h	0.50
Night	8pm to 12 am	102 veh/h, 44 km/h	0.53

Algorithm 3 shows a pseudo code for detecting bad traffic state where a rule is defined to check if three consecutive readings of traffic speed and traffic intensity are less then the threshold values.

Algorithm 3 Example Rule for CEP

1: **for** $(speed, intensity) \in TupleWindow(3)$ **do**
2: **if** ($speed(t) < speed_{thr}$ and $intensity(t) < intensity_{thr}$ AND
3: $speed(t+1) < speed_{thr}$ and $intensity(t+1) < intensity_{thr}$ AND
4: $speed(t+2) < speed_{thr}$ and $intensity(t+2) < intensity_{thr}$) **then**
5: Generate complex event Bad Traffic
6: **end if**
7: **end for**

5.4.1 Evaluation

In order to evaluate the proposed solution, this work follows the approach outlined in [45]. An ideal set of threshold values was defined for the rule mentioned in algorithm 3 for four different locations with the help of traffic administrators from Madrid city council, and refer to this as Rule R. Complex events detected using Rule R provides the ground truth data about the traffic state. Whereas the threshold values generated by the proposed solution for algorithm 3 constitutes the Rule R^*. Threshold values used in both rules for location 1 is shown in the table 5.3. The performance of the system was then measured quantitatively by generating an evaluation history of traffic

events for both R and R^* to detect good and bad traffic events.

Table 5.3: Threshold values for Morning time period

Locations	Rule R	Rule R^*
Location 1	150 veh/h, 40 km/h	130 veh/h, 43 km/h
Location 2	500 veh/h, 50 km/h	455 veh/h, 51 km/h
Location 3	150 veh/h, 40 km/h	135 veh/h, 44 km/h
Location 4	500 veh/h, 50 km/h	514 veh/h, 48 km/h

This enables to measure the precision of the algorithm which is the ratio of the number of correct events to the total number of events detected; and the recall, which is the ratio of the number of events detected by R^* to the total number of events that should have been detected based on ideal Rule R.

Mathematically, they are represented as:

$$Precision = \frac{TP}{TP+FP}, \quad Recall = \frac{TP}{TP+FN} \tag{5.3}$$

where TP is true positive, FP is false positive and FN is false negative. Results are shown below in table 5.4. In general, we got high values of recall for all four locations which indicates high rule sensitivity (detecting 90% of events from the traffic data stream). The average value of precision lies at around 80% indicating a small proportion of false alarms.

Table 5.4: CEP rules evaluation for Madrid scenario

Locations	TP	FP	FN	Precision	Recall
1	97	17	8	0.85	0.92
2	68	14	7	0.82	0.91
3	71	12	11	0.85	0.86
4	112	29	9	0.79	0.93

5.5 Large-Scale Implementation

The context-aware event processing method was extended to propose and implement the architecture for large-scale IoT applications[1]. The proposed architecture is called the *Hut* architecture because its flow diagram takes the form of a hut as shown in Figure 5.5. This work addresses the ingestion and analytics challenges, integrates different big data components and implement the complete end to end system for large-scale IoT applications.

5.5.1 The Hut Architecture

Figure 5.5 presents the data flow diagram of the proposed *Hut* architecture, which forms the shape of a hut. The purple arrows denote the batch data flows which form the base of the hut, while the green arrows denote the real-time flows and form the roof of the hut.

Data acquisition denotes the process of collecting data from IoT devices and publishing it to a message broker. An event processing framework consumes events and possibly takes some action (actuation) affecting the same or other IoT devices or other entities such as a software application. Real-time flows can be stand alone, in cases where real-time data can be acted upon without benefiting from historical data, although usually historical data can provide further insight in order to make intelligent decisions on real-time data. For example, in order to recognize anomalies, a system first needs to learn normal behaviour from historical data [11].

The batch flows fulfil this purpose. Data is ingested from the message broker into a data storage framework for persistent storage. Data can then be retrieved and analysed using long running batch computations, for example, by applying ML algorithms. The result of such analysis can influence the behaviour of the real-time event processing

[1]Part of this work was done in collaboration with IBM Research Haifa. Details related to the development/integration of specific components which were done by IBM Haifa are not included in this thesis and can be found in [76].

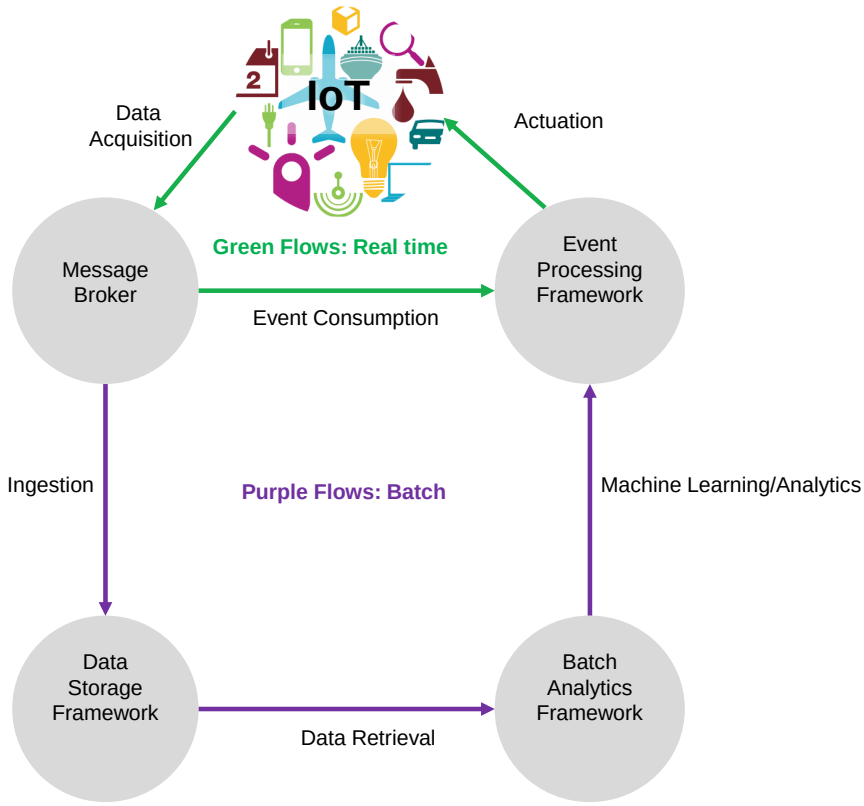


Figure 5.5: The Hut Architecture

framework. The batch flows can work independently of the real-time flows to provide long term insight or to train predictive models using historical datasets.

5.5.2 A Hut Architecture Instance

For each node in Figure 5.5, one can choose among various alternatives for its concrete implementation. Certain choice of such components are referred as a *Hut architecture instance*. To achieve high scalability and low deployment cost, a cloud based micro-services approach was adopted, where each capability (ingestion, storage, analytics etc.) is embodied in a separate scalable service. This approach is gaining widespread popularity for cloud platform-as-a-service (PaaS) [35], since each service specializes in what it does best, and can be managed and scaled independently of other services, avoiding monolithic software stacks. To achieve low development cost we adopt open source frameworks and extended them where needed. We choose "best of class" open

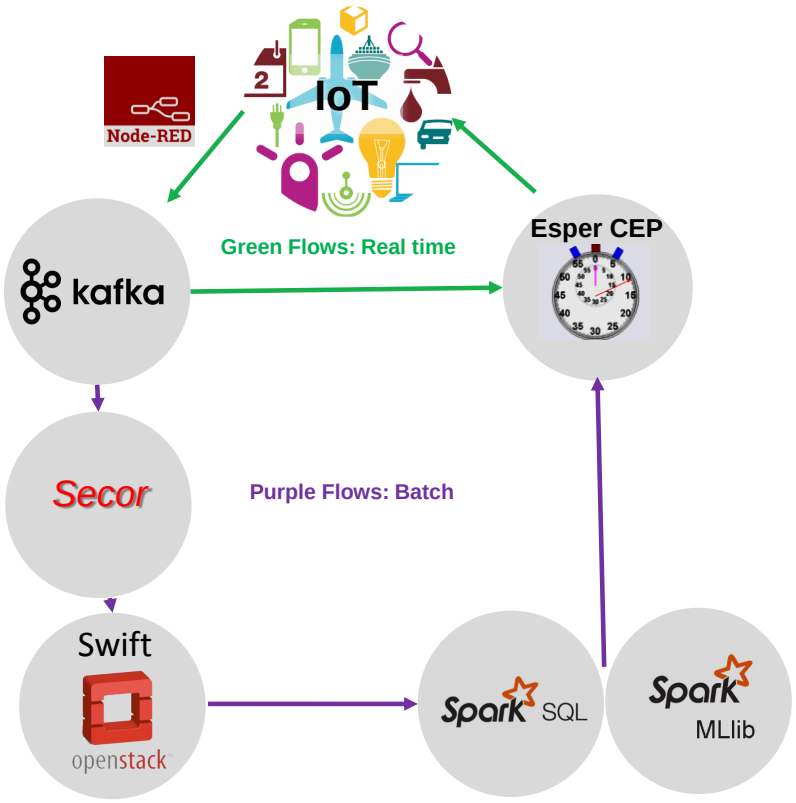


Figure 5.6: Instance of the *Hut* architecture

source frameworks for each capability, and show how they can be assembled to form an architecture for IoT applications. A diagram of this instance is shown in Figure 5.6.

The role of each component and how it fits into overall architecture is described below. Few of the components used in *the hut architecture* were already described in section 4.2 and hence detail of those components are skipped in this section.

5.5.2.1 Ingestion - Secor Secor is an open source tool[1] developed by Pinterest which allows uploading Apache Kafka messages to Amazon S3. Multiple messages are stored in a single object according to a time or size based policy. We enhanced Secor by enabling OpenStack Swift targets, so that data can be uploaded by Secor to Swift. In addition we enhanced Secor by enabling data to be stored in the Apache Parquet format, which is supported by Spark SQL, thereby preparing the data for analytics.

[1] https://github.com/pinterest/secor

Secor was selected because it is an open source connector between Kafka and object storage (OpenStack Swift).

5.5.2.2 Data Storage Framework - OpenStack Swift OpenStack[1] is an open source cloud computing software framework originally based on Rackspace Cloud Files. OpenStack is comprised of several components, and its object storage component is called Swift[2]. OpenStack Swift supports CReate, Update and Delete (CRUD) operations on objects using a REST API, and supports scalable and low cost deployment using clusters of commodity machines. For this reason Swift is suitable for long term storage of massive amounts of IoT data. We chose OpenStack Swift because it is an open source object storage framework.

5.5.2.3 Batch Analytics Framework - Spark Apache Spark is a general purpose analytics engine that can process large amounts of data from various data sources and has gained significant traction. It performs especially well for multi-pass applications which include many machine learning algorithms [94]. Spark maintains an abstraction called resilient distributed datasets (RDDs) which can be stored in memory without requiring replication and are still fault tolerant. Spark can analyse data from any storage system implementing the Hadoop FileSystem API, such as HDFS, Amazon S3 and OpenStack Swift, which, together with performance benefits and SQL support (see next section), is the reason for our choice.

5.5.2.4 Data Retrieval - Spark SQL RDDs which contain semi-structured data and have a schema are called DataFrames and can be queried according to an SQL interface. This applies to data in Hadoop compatible file systems as well as external data sources which implement a certain API, such as Cassandra and MongoDB. We implemented this API for OpenStack Swift with Parquet and Elastic Search, to allow taking advantage of metadata search for SQL queries.

[1] https://www.openstack.org/

[2] https://docs.openstack.org/swift/latest/

5.5.2.5 Machine Learning - Spark ML Spark MLlib [51] is Sparks library for machine learning. Its goal is to make practical ML scalable and easy to use. Spark ML is designed to address the limitations of Hadoop MapReduce and improve performance especially for iterative algorithms. Spark provides the ability to run computations in memory which enables much faster computation times for complex and iterative algorithms compared to traditional MapReduce which requires significant disk I/O. Spark MLlib consists of common machine learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as lower-level optimization primitives and higher-level pipeline APIs.

The proposed architecture is modular, so a particular component in this instance could be replaced by another. For example, Spark Streaming or Apache Storm could be used for the event processing framework instead of CEP software, and Hadoop map reduce could be used instead of Spark. Our focus here is on the architecture itself, and in order to demonstrate the architecture we made an intelligent choice of open source components as an architecture instance.

5.6 Demonstration of generality

In order to demonstrate the generality of the proposed architecture, second use-case scenario was chosen from smart energy management. Smart energy kits are gaining popularity for monitoring real-time energy usage to raise awareness about users energy consumption [27]. The Institute for Information Industry (III) Taiwan have deployed smart energy kits consisting of smart plugs and management gateways in over 200 residences. These smart plugs have built-in energy meters which keep track of real-time energy usage of connected appliances by logging electrical data measurements. They are connected to a management gateway via the ZigBee protocol, which is connected to the internet via WiFi.

The aim of the smart plugs is to monitor energy consumption data in real-time and automatically detect anomalies which are then communicated to the respective users. An anomaly can be defined as unusual or abnormal behaviour. For example, a malfunc-

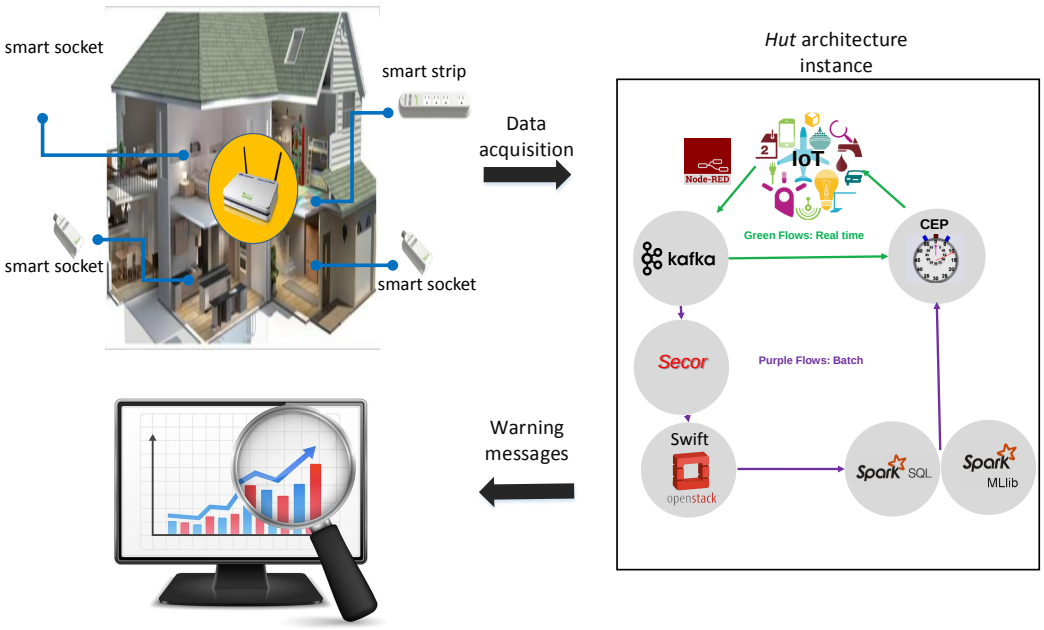


Figure 5.7: Smart energy use case

tioning electronic device or a fridge with its door left open can result in excessive power dissipation which should be detected and reported as soon as possible. Another type of anomaly is appliance usage at unusual times such as a radiator during the summer or an oven operated at 3am. Automatic monitoring of devices to detect anomalies can contribute to energy savings as well as enhanced safety.

III requests users to provide information on devices connected to smart plugs such as appliance type as well as expected behaviour such as expected wattage and current ranges. However expected behaviour is not usually known by users and is difficult for them to determine. The proposed approach of collecting historical appliance data for various time periods (summer versus winter, day versus night, weekday versus weekend) provides a way to automatically generate reliable information about expected behaviour. For each device and time context (such as weekday mornings during summer), the normal working range is calculated for current and power for an appliance using statistical methods. A CEP rule is defined based on this working range, and as soon as the readings are outside this range a CEP rule will be triggered generating a complex event representing an anomaly which can then be used to notify the user as

well.

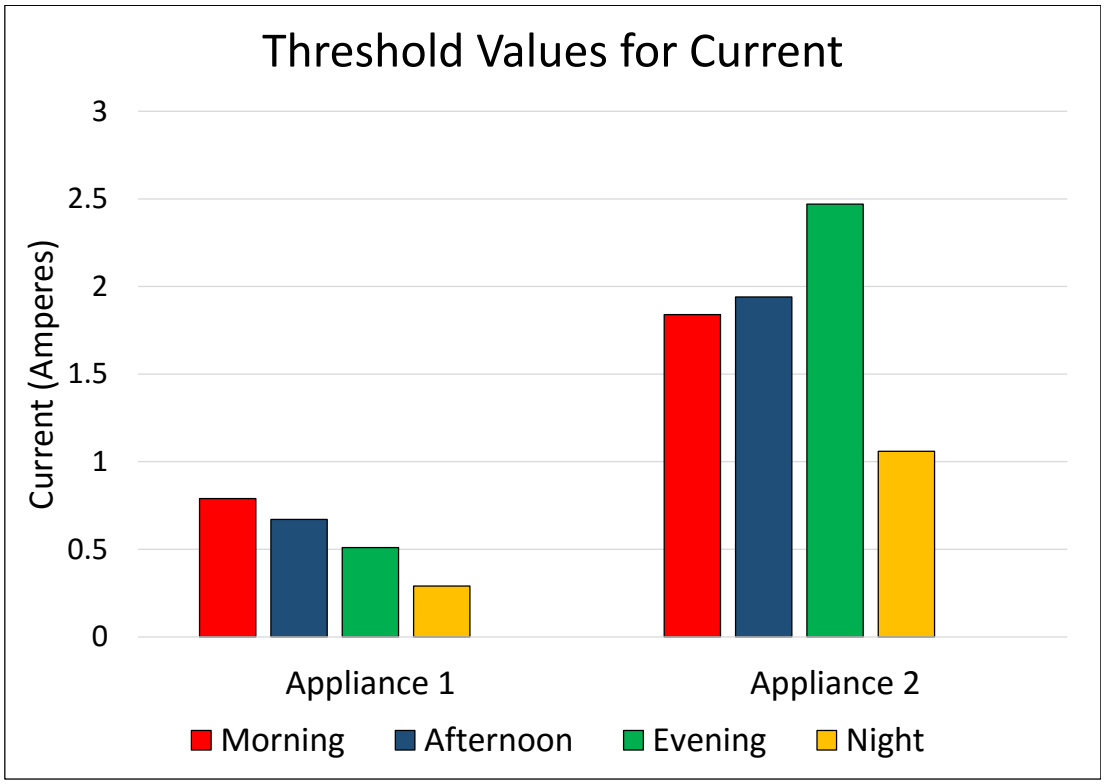


Figure 5.8: Appliances threshold values for monitoring current

An example of threshold values for two appliances during summer weekdays is shown in the Figure 5.8, calculated using the historical data of the specific device. As can be seen, both appliances have lower usage at night indicating smaller threshold values for current whereas appliance 1 has higher usage during mornings compared to appliance 2, which has a peak during evening time. A rule can be defined which compares the average current taken by an appliance over the specific time period to compare it with the expected readings for that time context.

In summary, the same data flow applies to this use case as for the Madrid Transportation use case described earlier. The main difference lies in how the historical data is analysed.

5.7 Discussion

In this chapter, a context-aware method was proposed to analyse and extract high-level knowledge from data streams in near real-time. The proposed method is automatic, adaptive and is able to cope with dynamic environments as opposed to current state of the art methods. The feasibility and usage of the proposed method was demonstrated with the help of a real-world use case scenarios of ITS.

The proposed method was extended to propose and implement the *Hut* architecture for carrying analytics for large-scale IoT applications. The proposed architecture supports both real-time and historical data analytics using its hybrid data processing model. The proposed architecture was implemented using open source components optimized for large-scale applications.

A major benefit of adopting such an architecture is the potential cost reduction at both development and deployment time by using a common framework for multiple IoT applications and plugging in various alternative components to generate variations as needed. Given the generality of the proposed architecture, it can also be applied to many other IoT scenarios such as monitoring goods in a supply chain or smart health care.

Chapter 6

Handling Uncertainty in Event Processing

This chapter extends the work from previous chapter by introducing the concept of uncertainty while detecting complex events from multiple low-level events and corresponds to the probabilistic event processing block of the overall system framework as highlighted in chapter 3[1]. In this regard, an hierarchical event processing framework is proposed which is based on two levels of analytics where first level provides a generic interface using a service oriented gateway to ingest data from multiple interfaces and IoT systems, store it in a scalable manner and analyse it in real-time to extract high-level events whereas second layer is responsible for probabilistic processing of these high-level events by exploiting Bayesian inference with CEP. The chapter starts with a brief introduction in section 6.1 followed by the description of illustrative scenario which is used to elaborate this contribution in section 6.2. Section 6.3 extends the *Hut* architecture to ingest and analyse heterogeneous data streams including twitter and weather data. The proposed approach towards probabilistic event processing based on Bayesian networks and CEP is described in section 6.4. Section 6.5 demonstrates the application of the proposed solution on different data streams with explanation of the

[1]This approach has been presented in the following publication:

1) A. Akbar, G. Kousiouris, J. Sancho, P. Ta-Shma, F. Carrez and K. Moessner "Real-time Probabilistic Data Fusion for Large-Scale IoT Applications," *IEEE Access, 2017* (under review).

results. Finally, the chapter is concluded by highlighting how the proposed approach solves the problem of probabilistic event processing and provides an edge on existing technologies in section 6.6.

6.1 Introduction

Forward-thinking cities and smart city solution providers recognize that the full potential of IoT cannot be reached by providing disparate smart city point solutions but rather the focus should be a scalable IoT infrastructure that integrates multiple systems or data streams efficiently. Currently, most of the IoT data is either not in use or have been under-used by limiting it for specific applications. For example, weather data sensors have been deployed for years but were seldom used for other applications. In the new world of connectivity, weather data has huge potential for a range of applications. It can be correlated with shopping centres sales to determine the effect of weather on shopping patterns or can be combined with traffic conditions to understand how weather can effect traffic. According to McKinsey, 40 percent of the total value that IoT can provide requires different IoT systems to work together [44].

There are many challenges in this regard from data ingestion and storage at one end to scalable and efficient data analytics at other end. Different IoT systems were built on different protocols and building a single and global ecosystem for IoT that can work together is certainly a non-trivial task. Moreover, different data streams have different level of uncertainty associated with them, and taking this uncertainty into account and extract high-level knowledge from these data streams requires new and novel solutions.

In contrast to existing approaches (as mentioned in section 2.4.3), a two layer analytics solution is proposed as shown in Figure 6.1. Level 1 analytics is responsible for ingesting and analysing individual IoT data streams coming from different IoT systems in the form of raw events to extract high level knowledge which we called derived events in our system. Layer 2 analytics is responsible for probabilistic processing of derived events by combining Bayesian Networks (BNs) with Complex Event Processing (CEP) as described in section 6.4. Different terms used in this contribution are defined below to have a clear understanding.

Raw Events: Raw data ingested from IoT data sources is referred as raw events in the proposed system. For example, data from traffic sensors constitutes raw events.

Derived Events: Derived events is any high-level events which is extracted after analysing raw events. It can be the combination of different type of raw events or extracted from single stream of raw events.

Probabilistic Complex Events: Probabilistic complex event in our system is referred as the final output which is extracted from the combination of different derived events using Bayesian networks and CEP.

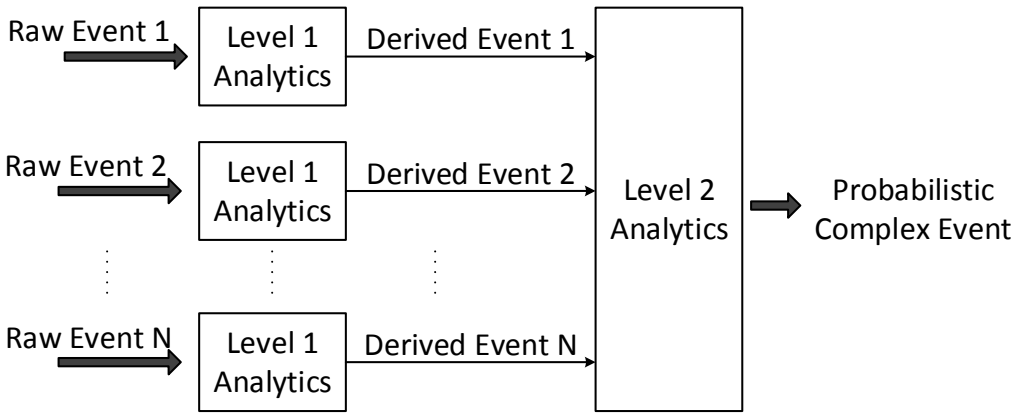


Figure 6.1: Proposed approach

A summary of contributions explained in this chapter is given below;

- Propose a solution based on probabilistic graphical models in order to take uncertainty into account while detecting complex events using CEP. Explore the use of Bayesian networks and extend it with CEP for large-scale systems.
- Validation of proposed solution on a real world scenario of intelligent transportation where multiple data streams including traffic, weather and social media data were used. Furthermore, development of a complete end-to-end solution with

demonstration on the application of proposed solution for real-time management of urban traffic control.

6.2 Illustrative Scenario

This work demonstrates the proposed solution using ITS use-case scenario as described in section 3.6. In our approach, we explore the use of social media data and weather data along with traffic sensors to predict traffic conditions more accurately. This work proposes an hierarchical architecture for analysing heterogeneous data streams in a scalable manner. For ITS use case, traffic data is captured by traffic sensors deployed by Madrid city council, social media data is ingested in the form of twitter streams and weather data is gathered from open source APIs. A summary of data streams and high-level knowledge derived from them is given in table 6.1. In order to get fair analysis of results, three different locations were selected from city of Madrid as shown in the Figure 6.2. Our proposed solution provides a novel framework to combine high-level knowledge derived from individual data streams in a probabilistic manner.

Figure 6.2: Monitoring locations for analytics with bounding boxes in Madrid city

Table 6.1: Input data streams and derived events after level 1 analytics

Input Data Stream	Input Features	Date Range	Derived Event	API
Traffic data	traffic speed, traffic intensity	01/06/16 - 30/08/16	Traffic Congestion (C)	http://informo.munimadrid.es/informo/tmadrid/pm.xml
Twitter data	Geo-tagged tweets	01/06/16 - 30/08/16	Large Crowd Concentration (LCC)	https://developer.twitter.com/
Weather data	temperature, humidity, pressure, visibility, textual description	01/06/16 - 30/08/16	Weather conditions (W)	http://api.wunderground.com/

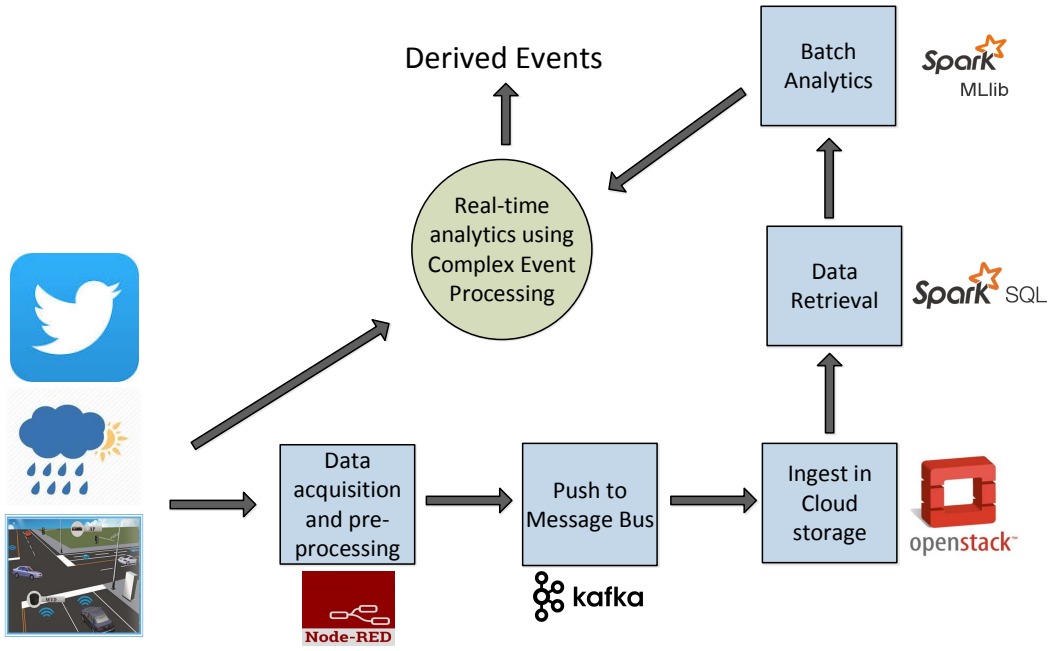


Figure 6.3: Proposed data flow for first level of analytics

6.3 Level 1 - Data Collection and Analytics

The framework and data flow for level 1 analytics is same as described earlier in section 5.5.2. In this work, the *Hut* architecture is modified to ingest social media data in the form of twitter and weather data through open source APIs. The requirements for every data stream (such as schema, api, data format) is different leading us to modify the components accordingly. Details about the methodology and individual data streams are explained below.

6.3.1 Traffic Data

The analytics on traffic data was explained earlier in section 5.4. Traffic data was ingested in the system using Node-Red flows, meta data such as time and location was added, pushed data to Kafka and stored in the form of objects in openstack swift cloud storage. The published data on Kafka has the following schema, where *intensidad*

denotes traffic intensity, *velocidad* denotes traffic speed, *ts* denotes the timestamp in epoch format and *tf* denotes the time of day.

```
{"namespace": "cosmos",
  "type": "record",
  "name": "TrafficFlowMadridPM",
  "fields": [
    {"name": "codigo", "type": "string"},
    {"name": "ocupacion", "type": "int"},
    {"name": "carga", "type": "int"},
    {"name": "nivelServicio", "type": "int"},
    {"name": "velocidad", "type": ["null","int"]},
    {"name": "intensidad", "type": ["null","int"]},
    {"name": "error", "type": "string"},
    {"name": "subarea",  "type": ["null","int"]},
    {"name": "ts", "type": "long"},
    {"name": "tf", "type": "string"}
  ]
}
```

Traffic data was gathered from the Madrid council API for three months. New values are provided every five minutes which result into approximately 25000 data points for each location. This data is analysed to generate optimised values of threshold for CEP rules using the approach described earlier in chapter 5. The implementation was done using SparkSQL and Spark MLlib. Complex events detected are stored in the cloud storage with current intensity and velocity readings as shown in the Figure 6.4 where '1' indicates a congestion event.

6.3.2 Twitter Data

Social networks have been identified as a rich source of information due to their extended uptake, through which significant inference may be achieved with relation to circumstances affecting the societal status. In this context, Twitter data is used as another source of information that can aid in the prediction of the Madrid Traffic state.

time_stamp	intensity	velocity	congestion
1.458234E9	4212	26	1
1.4576274E9	3784	31	1
1.4584788E9	3162	91	0
1.458315E9	6118	85	0
1.4581926E9	5952	89	0
1.457793E9	3880	91	0
1.4581206E9	4766	87	0
1.4579568E9	7100	81	0
1.4582844E9	4940	87	0
1.4584896E9	4210	89	0

only showing top 10 rows

Figure 6.4: Traffic events table layout in cloud storage

Exploring social media analytics is beyond the scope of current work, which is primarily the construction of an integrated system with heterogeneous data streams and probabilistic inference framework. In contrast to existing sophisticated methods for analysing social media data, this work followed a simple yet effective approach. The intuition behind the proposed approach is that number of tweets coming from a specific region is the indicative of number of people gathered in a region. For example, if tweets are extracted from a bounding region including a football stadium, the number of tweets will be far higher on a match day as compared to other days indicating a large crowd concentration. Similarly, there will be more tweets coming from a region including city centre or shopping malls during Christmas holidays as compared to a normal weekday indicating a crowd concentration. Several experiments were performed around Santiago Bernabeu stadium in Madrid in order to detect the large crowd concentration for football matches and is described in [37].

One reason for selecting twitter data is its easily available open API through which it can be ingested into the system and dynamically create alerts about surges in population concentration in a given area, when compared to the normal tweeting activity of the past. Increased Twitter activity in an area can be considered as an indirect indication about abnormal activity, especially when this relates to highways and not residential areas that could include an increased number of false positives (e.g. due to

popular Twitter trends in the specific timeslot).

In order to focus on a specific area (a bounding box around the location), filtering of the acquired messages from Twitter needs to be performed. This is done upon registration, in which we define the geographical bounding box from where we need the relevant tweets to be forwarded. This is taken under consideration by the respective Twitter API and only the tweets that have enabled geolocation and fall within this box are forwarded (in our case the bounding boxes for every location is highlighted in Figure 6.2). It is necessary to stress that not all of the tweets are forwarded, but only a percentage of them, according to the Twitter API documentation[1]. Following tweets acquisition, ingestion takes place as follows:

1) Initial filtering is performed in order to reduce each tweet size, by discarding fields of no interest and thus reducing needed storage space

2) Enrichment with sentiment analysis information based on the tweet content is performed. This information is not currently used but it was considered a promising feature for future work.

3) Definition of an AVRO schema necessary for describing how the data fields information will be stored in the Cloud storage and which fields will be maintained, which also affects the Node-RED manipulation

4) Adaptation of the incoming tweets to the AVRO format and JSON structure of the output to the Message Bus (Apache Kafka), from which it is collected by Openstack Swift

The published data on Kafka has the following schema;

```
{"namespace": "cosmos",
  "type": "record",
  "name": "TwitterData",
  "fields": [
    {"name": "ts", "type": "long"},
```

[1] https://developer.twitter.com/en/docs

```
    {"name": "text", "type": "string"},
    {"name": "lon", "type": "double"},
    {"name": "lat", "type": "double"},
    {"name": "twitter_id", "type": "string"},
    {"name": "sentiment", "type": "double"},
  ]
}
```

Once the ingestion flow has been established and sufficient data have been collected, they can be included in the runtime operation of the LCC event identification. Historical data is analysed using Spark SQL and Spark MLlib to learn about expected number of tweets for given location and time and CEP is used to calculate the running sum of number of tweets. Depending on the number of tweets, it generates different levels of Large Crowd Concentration (LCC) event.

Once a LCC event is detected, it is stored in the cloud storage along with twitter counts and time stamp. We have used three levels of LCC where '0' indicates no LCC or normal conditions, '1' indicates first level of LCC and '2' indicates second level of LCC event indicating a Large crowd in the region. More details can be found in our earlier work [37]. An instance of LCC data stored in cloud storage is shown in Figure 6.5.

```
+-----------+--------------+---------+
| time_stamp|twitter_counts|LCC_level|
+-----------+--------------+---------+
| 1.459134E9|             9|        0|
|1.4592978E9|            27|        0|
| 1.462329E9|            11|        0|
|1.4589702E9|            12|        0|
|1.4621652E9|            17|        0|
|1.4626566E9|            87|        0|
|1.4588064E9|            25|        0|
|1.4624928E9|            37|        0|
|1.4586426E9|            60|        0|
|1.4620014E9|            43|        0|
+-----------+--------------+---------+
only showing top 10 rows
```

Figure 6.5: LCC events table layout in object storage

6.3.3 Weather Data

There are number of open source weather services available for retrieving real-time weather data. For our work, we used open API provided by Weather Underground[1] to access data from the nearest weather station to our selected locations. They have several weather stations installed in the Madrid city as shown in the Figure 6.6. Location coordinates were passed in the API, and it returns the response conatining weather parameters in json format from the closest station automatically.

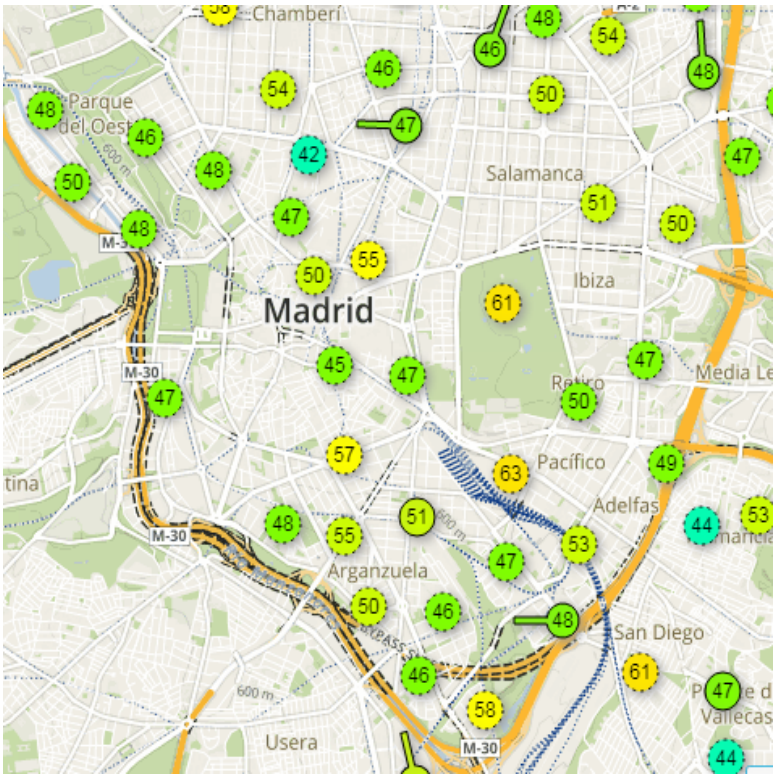


Figure 6.6: Weather data stations for Madrid

Combined with a Node-RED flow, we parsed the received data in a JSON file and filter out the needed parameters according to our defined schema. We assigned weather data different levels depending on the conditions column where '0' indicates a clear,sunny or cloudy weather, '1' indicates light rain or light shower and '2' represents heavy rain or stormy weather. An instance of weather table stored in cloud storage after processing raw data is shown in Figure 6.7.

[1]http://api.wunderground.com/

```
+-----------+------------+--------+-------------+
| time_stamp|TemperatureC|Humidity|weather_level|
+-----------+------------+--------+-------------+
|1.4575644E9|           7|      33|            0|
|1.4575644E9|         7.0|      46|            0|
|1.4575662E9|         6.0|      49|            0|
| 1.457568E9|           7|      34|            0|
| 1.457568E9|         6.0|      49|            0|
|1.4575698E9|         6.0|      49|            0|
|1.4575716E9|           6|      40|            0|
|1.4575716E9|         5.0|      57|            0|
|1.4575734E9|         5.0|      57|            0|
|1.4575752E9|           5|      47|            0|
+-----------+------------+--------+-------------+
only showing top 10 rows
```

Figure 6.7: Weather events table layout in object storage

```
{"namespace": "cosmos",
  "type": "record",
  "name": "WeatherData",
  "fields": [
    {"name": "ts", "type": "long"},
    {"name": "text", "type": "string"},
    {"name": "lon", "type": "double"},
    {"name": "lat", "type": "double"},
    {"name": "temperature", "type": "float"},
    {"name": "relative humidity", "type": "float"},
    {"name": "weather", "type": "text"},
    {"name": "visibility", "type": "float"},
    {"name": "pressure", "type": "double"},
  ]
}
```

6.4 Level 2: Probabilistic Event Processing

Once high-level events are extracted from individual data streams, they are correlated using CEP running at the second layer. In state-of-the-art CEP systems, events are

correlated using absolute rules where complex event detected is either true or false. In order to generate a complex event, incoming data streams should completely match the defined pattern and even if a single condition is violated, complex event will not be generated. It is a major drawback given the random and probabilistic nature of real-world events as explained in section 2.4.3. In contrast to conventional CEP systems, we propose a probabilistic CEP approach using BNs where complex events are detected in terms of probabilities. If the incoming data streams does not match completely to defined pattern, it still generates the complex event but with reduced probability. In our system, BNs were trained using large historical data and integrated with CEP rules.

6.4.1 What is a Bayesian Network

A Bayesian Network (BN) is a graphical structure that allows to represent and reason the inherent uncertainty in real world problems. The nodes in a BN represent a set of random variables, $X = X_1, ..X_i, ...X_n$, from a specific domain. A set of directed edges connect pairs of nodes, $Xi \rightarrow Xj$, representing the direct dependencies between variables. The strength of the relationship between different random variables is quantified by conditional probability distributions associated with each node. The only constraint on the edges allowed in a BN is that there must not be any directed cycles i.e. you cannot return to a node simply by following directed edges. Such networks are called directed acyclic graphs (DAG) . The relationship between different nodes is specified using a conditional probability distribution for every node. In case of discrete variables, it takes the form of a conditional probability table (CPT) . BNs model the quantitative strength of the connections between variables which allows it to update probabilistic beliefs about them automatically as new data arrive.

Markovian assumption is an important property when modelling with Bayesian networks. In order to understand Markovian assumption, consider the following notation;

- $Parents(X)$ are the parents of X in DAG G, that is, the set of variables Y with an edge from Y to X.
- $Descendants(X)$ are the descendants of X in DAG G, that is, the set of variables Y with a directed path from X to Y.

- $Non - Descendants(X)$ are all variables in DAG G other than X, Parents (X), and Descendants (X).

For the given notation, Markov property can be represented as:

$$I(X, Parents(X), non - descendants(X)) \tag{6.1}$$

for all variables X in DAG G. That is, every variable is conditionally independent of its non-descendants given its parents.

The probabilistic semantics of BNs can be interpreted using joint probability distribution. For a BN containing the n nodes, $X1 \rightarrow Xn$, taken in that order, a particular value in the joint distribution is represented by $P(X1 = x1, X2 = x2, ..., Xn = xn)$, or more compactly, $P(x1, x2, ..., xn)$. The chain rule of probability theory allows us to factorize joint probabilities so:

$$\begin{aligned} P(x_1, x_2, ..., x_n) &= P(x_1) \times P(x_2|x_1)..., \\ &\quad \times P(x_n|x_1, ..., x_{n-1}) \\ &= \prod_i P(xi|x1, ..., x_{i-1}) \end{aligned} \tag{6.2}$$

Applying Markovian assumption on BN structure implies that the value of a particular node is conditional only on the values of its parent nodes, this reduces to;

$$P(x1, x2, ..., xn) = \prod_i P(x_i|Parents(X_i)) \tag{6.3}$$

6.4.2 Why Bayesian Network

Graphical models like BNs have several advantages when they are used in conjunction with statistical methods for data analysis which are summarized below:

- Bayesian model encodes dependencies among all variables which enables it to handle situations where some data entries are missing. The dependencies between variables can be exploited to infer missing information.

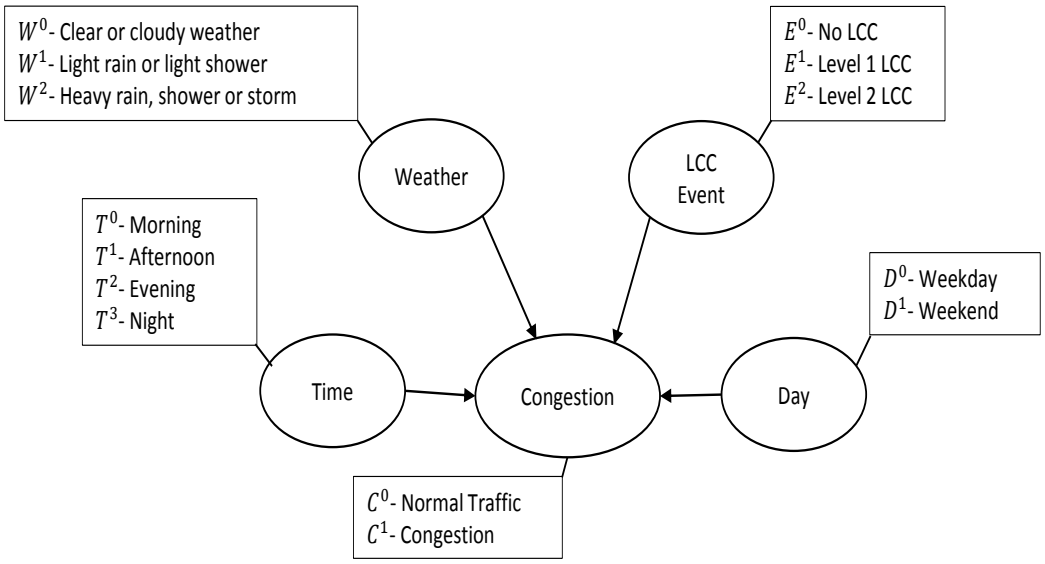


Figure 6.8: Proposed Bayesian network

- The model has both a causal and probabilistic semantics which makes it as an ideal representation for combining prior knowledge (which often comes in causal form) and data. Anyone having performed a real-world analysis knows the importance of prior or domain knowledge, especially when data is scarce or expensive.

- A BN can be used to learn causal relationships, and hence can be used to gain understanding about a problem domain and to predict the consequences of different combination of events.

- Finally, statistical methods in conjunction with BNs provides a rather generic and efficient solution avoiding the over-fitting of data. For BNs, there is no need to hold data for testing, instead all available data can be used for training.

6.4.3 Bayesian Network for Intelligent Transportation System

A major drawback with BNs is its high computational complexity for probabilistic inference which limits its use for large-scale applications [25]. Our *Hut* architecture provides a scalable approach to handle computational complexity and at the same time enables to handle uncertainty induced in CEP rules. In this section, we describe our approach for building BN and how it is explored to handle uncertainty in CEP rules.

To construct a Bayesian network for a given set of variables, we first identify cause and effect variables and then imply arcs from cause variables to their immediate effects. Traffic Congestion (C), Weather (W), Large Crowd Concentration Event (E), Time (T) and Day (D) are the set of variables in our system and we are interested to find the causal effect of these variables on a traffic Congestion (C). Assuming conditional independence between weather, LCC, time and and day, we come up with a simplified Bayesian structure as shown in the Figure 6.8.

6.4.3.1 Probabilistic Inference Once a Bayesian network is constructed, we need to estimate prior and conditional probabilities from the historical data. Our final goal is to get the probability of congestion given observations of the other variables. This probability is not stored directly in the model, and hence needs to be computed. In general, the computation of a probability of interest given a model is known as probabilistic inference. Because a Bayesian network for X variables determines a joint probability distribution for X variables, we can in principle use the Bayesian network to compute any probability of interest.

time_stamp	intensity	velocity	congestion	twitter_counts	LCC_level	weather_level	weekday	day
1.4586426E9	5080	86	0	60	0	0.0	2	0
1.4620014E9	4660	88	0	43	0	0.0	6	1
1.4621652E9	1418	91	0	17	0	0.0	1	0
1.4628204E9	2116	90	0	93	1	0.0	1	0
1.4620428E9	2138	90	0	88	0	0.0	6	1
1.4622066E9	3876	89	0	83	0	0.0	1	0
1.4623704E9	7076	65	0	74	0	0.0	3	0
1.4625342E9	6716	56	0	67	0	0.0	5	0
1.462698E9	4488	81	0	65	0	2.0	7	1
1.4599638E9	4132	88	0	73	0	0.0	3	0

only showing top 10 rows

Figure 6.9: An instant of all data combined in the form of table with unified time stamps

6.4.3.2 Data Fusion and Analytics Different data sources have different sampling time and every data reading has a different time stamp. Traffic data is updated every

five minutes, weather data is refreshed every thirty minutes and twitter data generates several tweets every minute. In order to combine all data sources, we need to bring all data on a common time scale. We performed linear interpolation to fill missing values and aggregated available data for every thirty minutes and combined data from all derived events history. The instance of resulting table with all data is shown in the Figure 6.9. The choice for aggregation of the data for every thirty minutes was done due to the availability of weather data for that interval. An alternative approach was to interpolate the weather data by assuming that weather remains the same over the thirty minutes and generate new readings every five minute. Once, all the data with individual derived events are combined in a single table, the calculation of conditional probabilities is a simple task using Spark SQL. As an example, probability of congestion when weather is good (represented by 0 in data), and no LCC event (represented by 0 in data) and afternoon time on a weekday can be calculated as below;

```
sqlContext.sql(" SELECT * FROM total_table
WHERE congestion = '0' AND LCC_level = '0'
AND day = '0' AND tf >= '08:00:00' AND
tf <= '12:00:00'  ").count()/ total_count
```

where total_table is the combined table of all data (an instance is shown in Figure 6.9)

Spark and Spark SQL provides an efficient and scalable approach for calculating conditional probabilities from this large amount of data from multiple tables in a timely manner. The number of possible combinations for conditional probability depends on the number of variables and their possible output states in the system. For our scenario, it equates to 4 x 3 x 3 x 2 x 2 = 144 possible combinations. An instance of conditional probability table for location 1 for morning time is shown in the Table 6.2.

6.5 Results and Discussion

Figure 6.11 shows the Bayesian network simulation for different combinations of input events at location 1. The simulations were done using SamIam (Sensitivity Analysis Modeling Inference And More), which is a comprehensive tool for modelling and

Table 6.2: Conditional probability table for location 1 for morning time

No.	E, W, D	C^0	C^1
1	E^0, W^0, D^0	0.82	0.18
2	E^0, W^0, D^1	0.98	0.02
3	E^0, W^1, D^0	0.88	0.12
4	E^0, W^1, D^1	0.96	0.04
5	E^0, W^2, D^0	0.78	0.22
6	E^0, W^2, D^1	0.97	0.03
7	E^1, W^0, D^0	0.81	0.19
8	E^1, W^0, D^1	0.86	0.14
9	E^1, W^1, D^0	0.79	0.21
10	E^1, W^1, D^1	0.94	0.06
11	E^1, W^2, D^0	0.74	0.26
12	E^1, W^2, D^1	0.96	0.04
13	E^2, W^0, D^0	0.80	0.20
14	E^2, W^0, D^1	0.93	0.07
15	E^2, W^1, D^0	0.77	0.23
16	E^2, W^1, D^1	0.92	0.08
17	E^2, W^2, D^0	0.73	0.27
18	E^2, W^2, D^1	0.93	0.07

reasoning with BNs developed at the University of California[1]. Conditional probabilities calculated in section 6.4.3.2 were used to implement Bayesian network. Figure 6.11a shows the probability of congestion when no prior information about any event is given. As soon as the current day and time events are given, probability of congestion is updated using probabilistic inference. Congestion probability increases from 11.64% to 24.60% as it is morning time on a weekday as shown in Figure 6.11b. Probability of congestion further increases when *LightRain* event is detected from weather data stream as shown in Figure 6.11c. Finally, as the system detects the final input event

[1]http://reasoning.cs.ucla.edu/samiam/

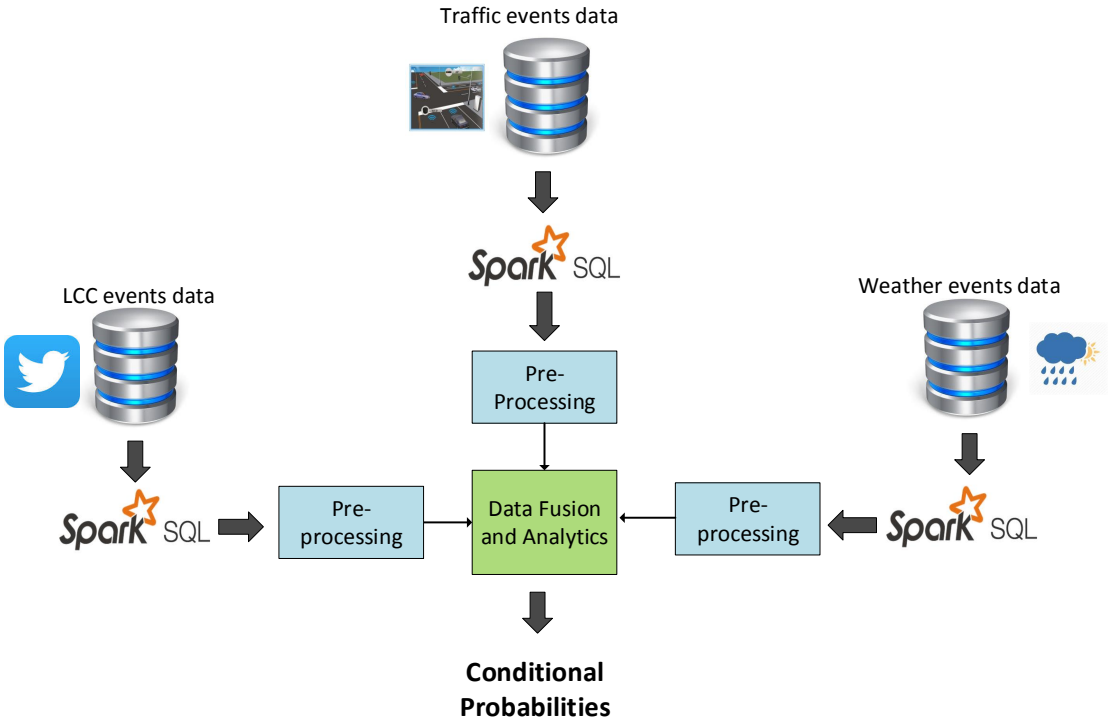
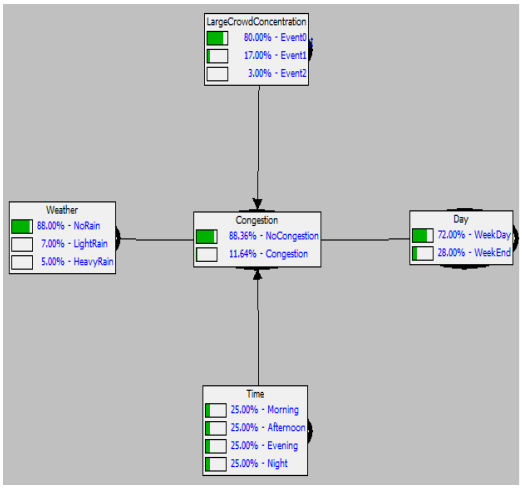


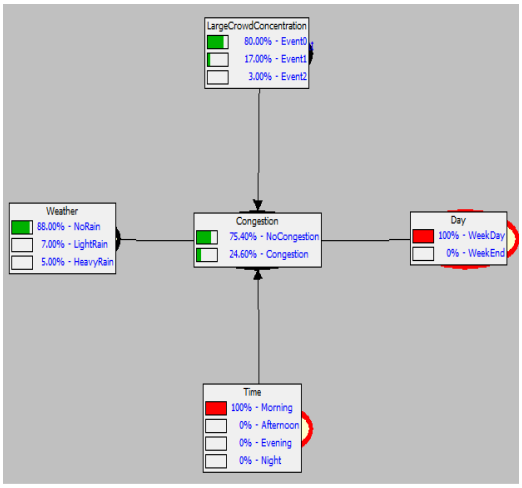
Figure 6.10: Calculation of conditional probabilities

LCC from twitter data, it updates the probability of congestion for the given context. Bayesian network implementation enables the CEP to take decisions and predict the output with incomplete data as evident from the different scenarios demonstrated in the Figure 6.11.

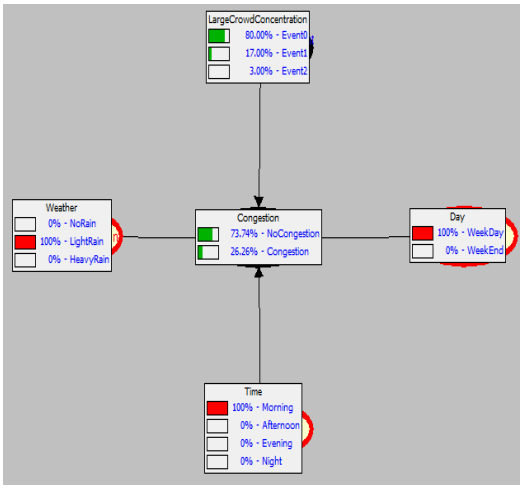
After simulating different scenarios, we embed the Bayesian probabilities in CEP rules and deployed it for our scenario. As the Bayesian implementation is done outside the CEP, our proposed solution is independent of the choice of particular CEP. In order to validate our solution, we worked with Madrid city council team responsible for managing traffic. We developed a web GUI using the worldmap node of Node-RED, in order to display different events and provide a visual output of our system as shown in the Figure 6.12. City of Madrid have cameras installed at selective locations for monitoring traffic and we have chosen the same locations in order to verify the prediction results. The output panel in Figure 6.12 shows the current state of traffic, LCC event and weather and the probability of congestion for one of the selected location.



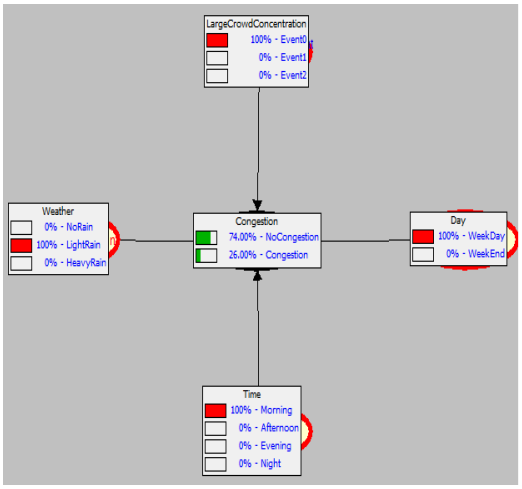
(a) scenario 1: when no event is given



(b) scenario 2: time and day given



(c) scenario 3: light rain event detected from weather data stream



(d) scenario 4: no LCC detected from twitter data stream

Figure 6.11: Different scenarios for location 1 simulated for Bayesian network

Traffic administrators can use these predictions and manage the traffic in pro-active manner. We have also developed a rating system where traffic administrators can rate the quality of prediction by giving feedback.

The output of the system is the probability of congestion at any time with respect to current context (location, time, day, weather and LCC events). In order to further evaluate our system, we set different threshold values on the probability of congestion and match if it correctly predicts the congestion in near future. The performance is evaluated in terms of precision and recall [50] which are defined as :

$$Precision = \frac{TP}{TP+FP}, \quad Recall = \frac{TP}{TP+FN} \tag{6.4}$$

where TP is true positive, FP is false positive and FN is false negative. Results are shown below in table 6.3. In general, setting low threshold values on probabilities to generate congestion events results in low precision whereas high threshold values results in low recall. Finding right balance between precision and recall is important, otherwise either it will generate many false alarms or will miss actual congestion events.

Table 6.3: Evaluation for congestion prediction

Threshold	TP	FP	FN	Precision	Recall
40%	84	27	4	0.75	0.95
50%	81	21	13	0.79	0.86
60%	72	14	29	0.84	0.71
70%	28	4	83	0.87	0.25

6.6 Discussion

This chapter presents an approach for handling uncertainty in real-world events when extracting high-level knowledge using two layer architecture. Layer 1 provides a generic interface for ingesting and analysing data from different IoT systems in a scalable manner. It was introduced in chapter 4 but extended in this chapter to apply on

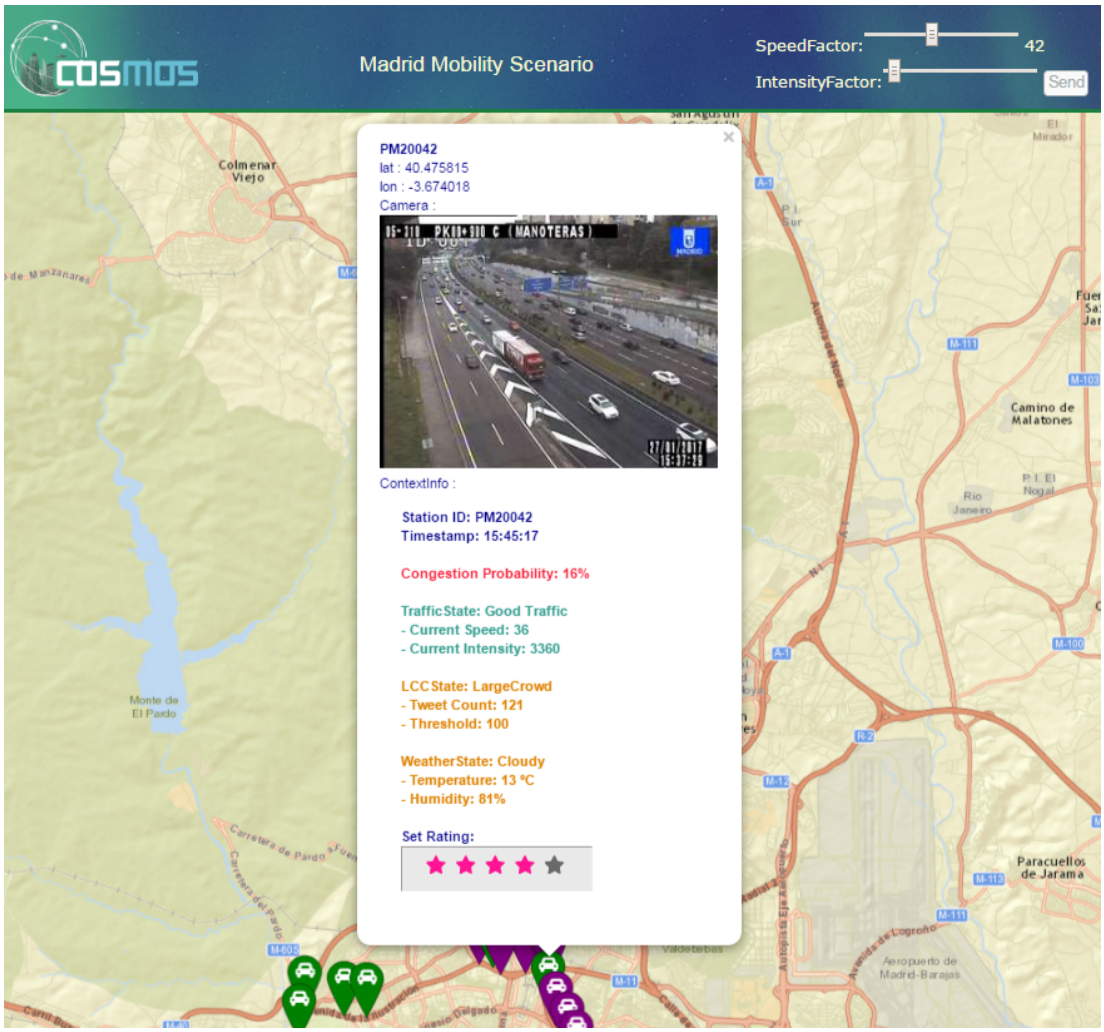


Figure 6.12: Display panel showing the output of the system

heterogeneous data streams. It provides an architecture based on the combination of batch and event processing methods for extracting high-level events from individual IoT data streams. Level 2 extends state-of-the-art CEP mechanisms to take inherent uncertainty of events into account and provide a probabilistic solution for correlating high-level events based on BN and CEP.

The feasibility of the proposed architecture was demonstrated with the help of a real-world use case from ITS where external data feeds of social media and weather were explored along with conventional traffic sensors in order to improve existing systems and generate early warnings of traffic congestion enabling system administrators to

manage traffic in a pro-active manner.

It should be noted that for this research all the locations were chosen from main highway of Madrid city, as traffic sensors were installed only at selective locations. Data was collected for over two months and congestion instants were limited during this time period. It introduces some bias as the effect of twitter events and weather data might be more prominent inside city and busy locations such as around city centre or around main tourist attractions.

Chapter 7

Conclusion

This chapter draws conclusions about this thesis and outlines future research directions on extracting knowledge from heterogeneous IoT data streams. It is this author's opinion that EDA architectures including CEP will play a vital role in future to analyse complex IoT data streams. However, the trend will be towards more hybrid approaches where CEP concepts will be combined with ML in order to get the best of both worlds.

7.1 Closing Remarks

This thesis addresses the problem of knowledge extraction from raw IoT data streams in near real-time. The process of knowledge extraction from these data streams is complex due to ever increasing amount of data produced by heterogeneous devices at very fast rate. This thesis proposed an event-driven approach based on CEP in order to tackle these challenges and extract high-level knowledge from IoT data streams.

The work presented in this thesis tackled four major challenges towards the successful implementation of CEP for IoT applications. CEP has been designed to provide reactive solutions by correlating data streams using predefined rules as the events happen, and do not exploit historical data. As the notion of many applications is changing from reactive to proactive, solutions based on CEP required an extension to address this issue. The first contribution of this thesis addressed this challenge by proposing an

hybrid architecture based on ML and CEP where historical data was exploited using ML part and combined with real-time flow of CEP to provide the basis for predictive event processing. The promise behind the proposed work is that if the input to the CEP is predicted data, then the complex event detected by CEP using causal and temporal pattern recognition techniques will be a predicted complex event. In contrast to the existing prediction methods which are based on static model parameters, this work proposed an adaptive prediction algorithm called adaptive moving window regression (AMWR) for dynamic IoT environments, which utilizes moving window for training the model and updates the model as new data arrives. The size of the training window is found automatically based on the assumption that the underlying data has an element of periodicity which might not be true for some IoT scenarios. In such cases, the performance of the system might degrade and require an improvement in the algorithm.

The second major challenge addressed by this work was towards providing automatic and context-aware solutions using CEP. In this regard, the second contribution proposed a novel method based on ML to find threshold values for CEP rules automatically and update it according to the current context. The application of proposed method for real-world problems was demonstrated with the help of traffic and energy data. This work also validates the accuracy of proposed method with the help of traffic administrators. The proposed method is unsupervised by nature however, it is dependent on the semantics of CEP rules. For example, k-means clustering was applied with $k = 2$ because the rules were defined to detect two traffic events as good and bad. In case, if the semantics of the CEP rules is changed, the underlying algorithm will need to updated accordingly.

The third challenge addressed by this thesis was incorporating uncertainty associated with real-world events into account while detecting complex events. This work proposed a probabilistic event processing approach based on two layers of analytics. The first layer is based on second contribution and responsible for analysing individual data streams to detect high-level knowledge whereas second layer extends state-of-the-art event processing using Bayesian Networks (BNs) in order to take uncertainty into account while detecting complex events. This work demonstrates the complete system from data ingestion to storage and finally analysing it using BNs and CEP. It high-

lights the practical problems towards calculation of conditional probabilities for BNs in large-scale systems and provide optimised solution. For evaluation, this work relied on the qualitative feedback provide by the traffic administrators from city of Madrid. In future, that can be improved further by utilising data provided from google maps[1] or Tomtom APIs[2].

The fourth challenge addressed by this research was to handle the largeness and complexity of IoT data when extracting knowledge. In order to address it, a novel *Hut* architecture was proposed by combining batch and event analysis frameworks. The proposed architecture was implemented using open-source components which are optimized for large-scale applications. It represents a common challenge which was addressed in all other three contributions using the proposed *Hut*architecture.

The work done in this thesis helps to bridge the gap between the academics and industry as all the proposed solutions were implemented for real-world problems. The author is of the opinion that there is a huge gap between the academic research and its successful implementation for real-world scenarios. Therefore, this research proposed solutions for extracting knowledge from raw IoT data streams which are practically implementable and adds value to real-world problems.

7.2 Future Work

The focus of this research was to develop novel methods to extract high-level knowledge from raw IoT data streams efficiently. Even though the solutions presented in this research contribute significantly to improve the state-of-the-art, still they can be improved further for the wide spread usage of these methods. In this section, we briefly discuss promising research directions for future work that might lead to better solutions to the problem at hand.

[1] https://developers.google.com/maps/

[2] https://developer.tomtom.com/online-traffic/

7.2.1 Algorithm Development

In the first contribution, this thesis proposed to adapt the prediction window based on the output accuracy of predictions using hard-coded upper and lower bounds. In future, the size of the prediction window can be updated using fuzzy logics. This thesis proposed a novel algorithm for context-aware event processing by finding threshold values for CEP rules automatically as second contribution. The proposed approach is based on k-means clustering which is an unsupervised ML method. It will be interesting to explore other clustering methods including Fuzzy clustering, density-based spatial clustering of applications with noise (DBSCAN) or hierarchical clustering for the same problem. Another option is to explore classification methods from supervised ML domain. However, gathering of ground truth data for supervised ML methods is a complex and challenging process. In this regard, an interesting option will be to explore proprietary data sources such as Google maps or Tomtom maps to gather the ground truth data. The same sources can also be used in future to further validate the proposed solutions which are presented in this thesis. This work used historical data to find patterns and extract threshold values. In future, methods based on micro-batch analysis can also be explored to train the models in real-time. The third contribution of this thesis extended BN with CEP to provide probabilistic event processing solution. In future, it can be extended with dynamic BN in order to exploit temporal patterns between different data sources.

7.2.2 Architecture Development

This thesis proposed an architecture based on ML and CEP for extracting knowledge and used different components for ingesting, analysing and correlating data streams. Although, the proposed solution is generic and modular where different components can easily be integrated or replaced but still it requires some expertise in order to operate it. In future, it can be made more generic with improved interfaces to encourage wide-spread usage of the solution.

In this thesis, ML part was implemented separately and was integrated with CEP using external components. The author is of the opinion that in future, CEP technology will

be extended and part of ML capability will be provided as built-in solution to assist the users. Industry is heading towards more hybrid solutions where such functionalities might be provided as a service. Furthermore, CEP used in this work was running on a central server but with the advent of ubiquitous computing, in future some of the CEP functionalities will be shifted towards edge as part of middleware solution and presents an interesting research direction. The processing capability of CEP will depend on the hardware processing power and handling scalability will be a major challenge.

7.2.3 Use-cases Development

Although the architecture and methods presented in this research are generic, it was mostly demonstrated with the help of ITS use-case. In future, same architecture can be extended to apply for other IoT use-cases. The generic nature of architecture enables it to configure easily for new use-cases.

Bibliography

[1] D. Akin, V. P. Sisiopiku, and A. Skabardonis. Impacts of weather on traffic flow characteristics of urban freeways in istanbul. *Procedia-Social and Behavioral Sciences*, 16:89–99, 2011.

[2] E. Alevizos, A. Skarlatidis, A. Artikis, and G. Paliouras. Probabilistic complex event recognition: A survey. *ACM Comput. Surv.*, 50(5):71:1–71:31, September 2017.

[3] P. Anantharam, P. Barnaghi, K. Thirunarayan, and A. Sheth. Extracting city traffic events from social streams. *ACM Trans. Intell. Syst. Technol.*, 6(4):43:1–43:27, July 2015.

[4] M. A. Benjamin, R. A. Rigby, and D. M. Stasinopoulos. Generalized autoregressive moving average models. *Journal of the American Statistical Association*, 98(461):214–223, 2003.

[5] T. Bierhoff, J. Fernando, J. Safont, A. Alonso, and W. Thronicke. Real time traffic forecast. *Atos scientific white paper*, 2013, 2013.

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[7] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, Jun 1998.

[8] G. Cabanes, Y. Bennani, and N. Grozavu. Unsupervised learning for analyzing the dynamic behavior of online banking fraud. In *Data Mining Workshops*

(ICDMW), 2013 IEEE 13th International Conference on, pages 513–520. IEEE, 2013.

[9] E. Ceperic, V. Ceperic, and A. Baric. A strategy for short-term load forecasting by support vector regression machines. *IEEE Transactions on Power Systems*, 28(4):4356–4364, Nov 2013.

[10] T. Chaira. A novel intuitionistic fuzzy c means clustering algorithm and its application to medical images. *Applied Soft Computing*, 11(2):1711–1717, 2011.

[11] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[12] C. Y. Chen, J. H. Fu, T. Sung, P. F. Wang, E. Jou, and M. W. Feng. Complex event processing for the internet of things and its applications. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1144–1149, Aug 2014.

[13] C. Y. Chen, J. H Fu, T. Sung, P. F. Wang, E. Jou, and M. W. Feng. Complex event processing for the internet of things and its applications. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 1144–1149. IEEE, 2014.

[14] G. Cugola and A. Margara. Tesla: A formally defined event specification language. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, DEBS '10, pages 50–61, New York, NY, USA, 2010. ACM.

[15] G. Cugola and A. Margara. Complex event processing with t-rex. *Journal of Systems and Software*, 85(8):1709–1728, 2012.

[16] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, June 2012.

[17] G. Cugola, A. Margara, M. Matteucci, and G. Tamburrelli. Introducing uncertainty in complex event processing: Model, implementation, and validation. *Computing*, 97(2):103–144, February 2015.

[18] G. Cugola, A. Margara, M. Matteucci, and G. Tamburrelli. Introducing uncertainty in complex event processing: model, implementation, and validation. *Computing*, 97(2):103–144, 2015.

[19] E. Curry. Message-oriented middleware. *Middleware for communications*, pages 1–28, 2004.

[20] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[21] J. Dunkel, A. Fernndez, R. Ortiz, and S. Ossowski. Event-driven architecture for decision support in traffic management systems. *Expert Systems with Applications*, 38(6):6530 – 6539, 2011.

[22] O. Etzion. *Towards an Event-Driven Architecture: An Infrastructure for Event Processing Position Paper*, pages 1–7. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[23] O. Etzion, P. Niblett, and D. C. Luckham. *Event processing in action*. Manning Greenwich, 2011.

[24] I. Flouris, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Kamp, and M. Mock. Issues in complex event processing: Status and prospects in the big data era. *Journal of Systems and Software*, 127(Supplement C):217 – 236, 2017.

[25] F. C. Gregory. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2):393 – 405, 1990.

[26] E. Grosswald. The student t-distribution of any degree of freedom is infinitely divisible. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 36(2):103–109, Jun 1976.

[27] T. Hargreaves, M. Nye, and J. Burgess. Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors. *Energy policy*, 38(10):6111–6119, 2010.

[28] J. D Ichbiah and S. P Morse. General concepts of the simula 67 programming language. *Annual Review in Automatic Programming*, 7:65 – 93, 1972.

[29] S. Alex J and S. Bernhard. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[30] F. Lajos Jenő, B. Árpád, T. Gabriella, D. Hunor, V. László, and F. Lóránt. Predictive complex event processing: a conceptual framework for combining complex event processing and predictive analytics. In *Proceedings of the Fifth Balkan Conference in Informatics*, pages 26–31. ACM, 2012.

[31] F. Lajos Jeno, T. Gabriella, R. Racz, J. Panczel, T. Gergely, A. Beszedes, and L. Farkas. Survey on complex event processing and predictive analytics. In *Proceedings of the Fifth Balkan Conference in Informatics*, pages 26–31. Citeseer, 2010.

[32] J. B. Jun, S. H. Jacobson, and J. R. Swisher. Application of discrete-event simulation in health care clinics: A survey. *The Journal of the Operational Research Society*, 50(2):109–123, 1999.

[33] M. Tood K. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.

[34] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

[35] E. Keller and J. Rexford. The "platform as a service" model for networking. In *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, pages 4–4, Berkeley, CA, USA, 2010. USENIX Association.

[36] N. Khoussainova, M. Balazinska, and D. Suciu. Peex: Extracting probabilistic events from rfid data. Technical report, 2007.

[37] G. Kousiouris, A. Akbar, J. Sancho, P. Ta-shma, A. Psychas, D. Kyriazis, and T. Varvarigou. An integrated information lifecycle management framework for

exploiting social network data to identify dynamic large crowd concentration events in smart cities applications. *Future Generation Computer Systems*, 2017.

[38] K. Lange, R. Little, and J. Taylor. Robust statistical modeling using the t distribution. *Journal of the American Statistical Association*, 84(408):881–896, 1989.

[39] O. J. Lee and J. E. Jung. Sequence clustering-based automated rule generation for adaptive complex event processing. *Future Generation Computer Systems*, 66:100 – 109, 2017.

[40] L. Li, F. Noorian, D. J. M. Moss, and P. H. W. Leong. Rolling window time series prediction using mapreduce. In *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*, pages 757–764, Aug 2014.

[41] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, 34(4):579–595, 2000.

[42] D. Luckham. *The power of events*, volume 204. Addison-Wesley Reading, 2002.

[43] D. C. Luckham, J. J. Kenney, L. M. Augustin, J. Vera, D. Bryan, and W. Mann. Specification and analysis of system architecture using rapide. *IEEE Transactions on Software Engineering*, 21(4):336–354, Apr 1995.

[44] James Manyika, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin, and Dan Aharon. Unlocking the potential of the internet of things. *McKinsey Global Institute*, 2015.

[45] A. Margara, G. Cugola, and G. Tamburrelli. Learning from the past: Automated rule generation for complex event processing. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, pages 47–58, New York, NY, USA, 2014. ACM.

[46] Alessandro Margara, Gianpaolo Cugola, and Giordano Tamburrelli. Learning from the past: Automated rule generation for complex event processing. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, pages 47–58, New York, NY, USA, 2014. ACM.

[47] N. Marz and J. Warren. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2015.

[48] R. Mayer, C. Mayer, M. A Tariq, and K. Rothermel. Graphcep: Real-time data analytics using parallel complex event and graph processing. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, DEBS '16, pages 309–316, New York, NY, USA, 2016. ACM.

[49] N. Mehdiyev, J. Krumeich, D. Enke, D. Werth, and P. Loos. Determination of rule patterns in complex event processing using machine learning techniques. *Procedia Computer Science*, 61(Supplement C):395 – 401, 2015. Complex Adaptive Systems San Jose, CA November 2-4, 2015.

[50] I. Dan Melamed, Ryan Green, and Joseph P. Turian. Precision and recall of machine translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*, NAACL-Short '03, pages 61–63, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[51] X. Meng, J. Bradley, B. Yuvaz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *JMLR*, 17(34):1–7, 2016.

[52] G. W. Milligan and M. C. Cooper. A study of standardization of variables in cluster analysis. *Journal of classification*, 5(2):181–204, 1988.

[53] M. Mongiello, L. Patrono, T. Di Noia, F. Nocera, A. ParchitelH, I. Sergi, and P. Rametta. A complex event processing based smart aid system for fire and danger management. In *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, pages 44–49, June 2017.

[54] R. Mousheimish, Y. Taher, and K. Zeitouni. *autoCEP: Automatic Learning of Predictive Rules for Complex Event Processing*, pages 586–593. Springer International Publishing, Cham, 2016.

[55] S. Nechifor, B. Trnauc, L. Sasu, D. Puiu, A. Petrescu, J. Teutsch, W. Waterfeld, and F. Moldoveanu. Autonomic monitoring approach based on cep and ml for logistic of sensitive goods. In *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014*, pages 67–72, July 2014.

[56] Bruce Jay Nelson. *Remote Procedure Call*. PhD thesis, Pittsburgh, PA, USA, 1981. AAI8204168.

[57] T. A Nguyen and M. Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and Buildings*, 56:244 – 257, 2013.

[58] J. S. Oh, Yong Un Shim, and Yoon Ho Cho. Effect of weather conditions to traffic flow on freeway. *KSCE Journal of Civil Engineering*, 6(4):413–420, 2002.

[59] E. Opher and N. Peter. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[61] S. Petrovic. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic Workshop of Secure IT Systems*, pages 53–64, 2006.

[62] A. L Pope. *The CORBA Reference Guide: Understanding the Common Object Request Broker Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[63] W. H Press and G. B Rybicki. Fast algorithm for spectral analysis of unevenly sampled data. *The Astrophysical Journal*, 338:277–280, 1989.

[64] Y. Qiao, K. Zhong, H. Wang, and X. Li. Developing event-condition-action rules in real-time active database. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 511–516, New York, NY, USA, 2007. ACM.

[65] W. Rafique, S. M. Naqvi, P. J. B. Jackson, and J. A. Chambers. Iva algorithms using a multivariate student's t source prior for speech source separation in real room environments. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 474–478, April 2015.

[66] V. Ranadivé and K. Maney. *The two-second advantage: How we succeed by anticipating the future–just enough.* Crown Business, 2011.

[67] L. Richardson and S. Ruby. *RESTful web services.* " O'Reilly Media, Inc.", 2008.

[68] H. Roitman, J. Mamou, S. Mehta, A. Satt, and L. Subramaniam. Harnessing the crowds for smart city sensing. In *Proceedings of the 1st international workshop on Multimodal crowd sensing*, pages 17–18. ACM, 2012.

[69] N. I. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, 4(2):24–38, May 2009.

[70] N. P. Schultz-Møller, M. Migliavacca, and P. Pietzuch. Distributed complex event processing with query rewriting. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, pages 4:1–4:12, New York, NY, USA, 2009. ACM.

[71] Scouris. Scouris's Arduino Adventures . `https://scouris.wordpress.com/2010/09/10/temperature-sensing-with-arduino-php-and-jquery/`, 2010. [Online; accessed 5-May-2016].

[72] N. Septimiu, T. Bogdan, S. Lucian, P. Dan, P. Anca, T. Joachim, W. Walter, and M. Florin. Autonomic monitoring approach based on cep and ml for logistic of sensitive goods. In *Intelligent Engineering Systems (INES), 2014 18th International Conference on*, pages 67–72. IEEE, 2014.

[73] S. Shi, D. Jin, and G. Tiong-Thye. Real-time public mood tracking of chinese microblog streams with complex event processing. *IEEE Access*, 5:421–431, 2017.

[74] S. Shwartz, S. Srebro, and A. Nathan. Svm optimization: Inverse dependence on

training set size. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 928–935, New York, NY, USA, 2008. ACM.

[75] U. Sivarajah, M. Mustafa Kamal, Z. Irani, and V. Weerakkody. Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, 70(Supplement C):263 – 286, 2017.

[76] P. Ta-Shma, A. Akbar, G. Gerson-Golan, G. Hadash, F. Carrez, and K. Moessner. An ingestion and analytics architecture for iot applied to smart city use cases. *IEEE Internet of Things Journal*, PP(99):1–1, 2017.

[77] B. Tarnauca, D. Puiu, D. Damian, and V. Comnac. Traffic condition monitoring using complex event processing. In *System Science and Engineering (ICSSE), 2013 International Conference on*, pages 123–128. IEEE, 2013.

[78] F. Terroso-Senz, M. Valds-Vela, F. Campuzano, J. A. Botia, and A. F. Skarmeta-Gmez. A complex event processing approach to perceive the vehicular context. *Information Fusion*, 21:187 – 209, 2015.

[79] C. Tsai, C. Lai, M. Chiang, and L. Yang. Data mining for internet of things: a survey. *Communications Surveys & Tutorials, IEEE*, 16(1):77–97, 2014.

[80] Y. Turchin, A. Gal, and S. Wasserkrug. Tuning complex event processing rules using the prediction-correction paradigm. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, pages 10:1–10:12, New York, NY, USA, 2009. ACM.

[81] B. Trnauc, D. Puiu, S. Nechifor, and V. Comnac. Using complex event processing for implementing a geofencing service. In *2013 IEEE 11th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 391–396, Sept 2013.

[82] K. Vikhorev, R. Greenough, and N. Brown. An advanced energy management framework to promote energy awareness. *Journal of Cleaner Production*, 43:103 – 112, 2013.

[83] A. Wagner, D. Anicic, R. Stühmer, N. Stojanovic, A. Harth, and R. Studer. Linked data and complex event processing for the smart energy grid. *Proc. of Linked Data in the Future Internet at the Future Internet Assembly*, 2010.

[84] Qiang Wan, W. Zhang, Y. Xu, and I. Khan. Distributed control for energy management in a microgrid. In *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T D)*, pages 1–5, May 2016.

[85] D. Wang, E. A. Rundensteiner, H. Wang, and R. Ellison. Active complex event processing: Applications in real-time health care. *Proc. VLDB Endow.*, 3(1-2):1545–1548, September 2010.

[86] Y. H. Wang, K. Cao, and X. M. Zhang. Complex event processing over distributed probabilistic event streams. *Comput. Math. Appl.*, 66(10):1808–1821, December 2013.

[87] S. Wasserkrug, A. Gal, and O. Etzion. A model for reasoning with uncertain rules in event composition systems. *arXiv preprint arXiv:1207.1427*, 2012.

[88] S. Wasserkrug, A. Gal, O. Etzion, and Y. Turchin. Efficient processing of uncertain events in rule-based systems. *IEEE Transactions on Knowledge and Data Engineering*, 24(1):45–58, Jan 2012.

[89] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[90] Jennifer Widom and Stefano Ceri. *Active database systems: Triggers and rules for advanced database processing*. Morgan Kaufmann, 1996.

[91] C. H. Wu, J.M. Ho, and D. T. Lee. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):276–281, Dec 2004.

[92] Haiqin Yang, Kaizhu Huang, Irwin King, and Michael R. Lyu. Localized support vector regression for time series prediction. *Neurocomputing*, 72(10):2659 – 2669, 2009. Lattice Computing and Natural Computing (JCIS 2007) / Neural Networks in Intelligent Systems Designn (ISDA 2007).

[93] W. Yongheng and C. Kening. A proactive complex event processing method for large-scale transportation internet of things. *International Journal of Distributed Sensor Networks*, 2014, 2014.

[94] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[95] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. *HotCloud*, 10:10–10, 2010.

[96] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 423–438. ACM, 2013.

[97] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 423–438. ACM, 2013.

[98] J. Zhang, F. Y. Wang, K. Wang, W. H. Lin, X. Xu, and C. Chen. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, Dec 2011.

[99] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 116–, New York, NY, USA, 2004. ACM.

[100] H. Zhao and F. Magouls. A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 16(6):3586 – 3592, 2012.

[101] Y. Zhou, S. De, and K. Moessner. Real world city event extraction from twitter data streams. *Procedia Comput. Sci.*, 98(C):443–448, October 2016.