

UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

Situation Recognition from Multi-Resolution Event Streams

Permalink

<https://escholarship.org/uc/item/9x37b5gd>

Author

Pongpaichet, Siripen

Publication Date

2016

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, available at <https://creativecommons.org/licenses/by/4.0/>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,
IRVINE

Situation Recognition from Multi-Resolution Event Streams

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Siripen Pongpaichet

Dissertation Committee:
Professor Ramesh Jain, Chair
Professor Michael Carey
Professor Gopi Meenakshisundaram

2016

TABLE OF CONTENTS

	Page
LIST OF FIGURES	vi
LIST OF TABLES	viii
LIST OF ALGORITHMS	ix
ACKNOWLEDGMENTS	x
CURRICULUM VITAE	xii
ABSTRACT OF THE DISSERTATION	xv
1 Introduction	1
1.1 Situation Recognition in Spatio-Temporal Data Streams	3
1.1.1 Large volume and high velocity of data streams for real-time processing	4
1.1.2 Data at different granularities in the physical world	5
1.1.3 Disparate and heterogeneous data sources	7
1.2 Goals and Contributions	8
1.3 Thesis Overview	11
2 Literature Review	13
2.1 Social Life Networks	13
2.1.1 EventShop	16
2.1.2 Personal EventShop	17
2.1.3 Personalized Alerts	18
2.2 Situation Recognition	19
2.3 Data Sources and Data Integration	20
2.4 Data Storage	21
2.5 Real-time Analytics	22
2.5.1 Analytics Toolkit	22
2.5.2 Publish/Subscribe Systems	23
2.5.3 Distributed Stream Processing Engines (Workflow System)	24
2.5.4 Geo-spatial Stream Processing	25
2.6 Summary	26

3	EventShop Fundamentals	27
3.1	General System Architecture	28
3.2	Concepts and Data Structure	30
3.2.1	Data Unification Model	31
3.2.2	Data Representations	32
3.2.3	From Heterogeneous Data Streams to Emage Streams	34
3.2.4	Situation Recognition Operators	35
3.3	Advantages	36
3.4	Limitations	37
4	Research Challenges	39
4.1	Observing Real-World Phenomena	39
4.2	Research challenges	42
4.3	Framework Overview	45
4.4	Data Model	47
4.5	From Sensor Data to Situations	51
4.6	New EventShop System Architecture	53
4.7	Graphical User Interface	54
5	Archival System	58
5.1	Data Stream Ingestor	60
5.1.1	Data Adapter	60
5.1.2	Emage Generator	70
5.2	Data Store	73
5.2.1	AsterixDB	74
5.3	Archival System Infrastructure	78
6	Multi-Resolution Processing Engine	79
6.1	Definition of Granularity	80
6.1.1	Spatial Granularity	80
6.1.2	Temporal Granularity	82
6.1.3	Emage Granularity	83
6.2	Query Processing Engine	85
6.2.1	Resolution Selection	85
6.2.2	Error Estimation on Multi-resolution Emages	87
6.2.3	Emage Stream Executor	93
6.3	Emage Clustering Operator	96
6.3.1	Density-grid based clustering on data stream	97
7	Micro-Reports	103
7.1	Limitations of Micro-Blogs	105
7.2	Micro-Reports and Beyond	107
7.2.1	Basic Characteristics	108
7.2.2	Emerging Challenges and Opportunities	109
7.2.3	Applications from Situation Recognition	110

7.3	Situation Recognition from Micro-Reports	112
7.3.1	Micro-Reports Wrapper	113
7.3.2	Visual Analytics	113
7.4	Micro-Reports Proof of Concept	115
7.4.1	Detecting Evolving Situations from Concepts of Photos	116
8	EventShop Applications	120
8.1	Healthcare Applications	121
8.1.1	Asthma Risk Recommendation	121
8.2	Disaster Management Applications	125
8.2.1	Emergency Response for Thailand Flood	127
8.3	Smart Community and Smart City Applications	133
8.3.1	Trash Bins Management	133
9	Conclusions and Future Research	138
9.1	Conclusions	138
9.2	Future Research	140
9.2.1	Scalability	140
9.2.2	Advanced Operators	141
9.2.3	Usability and User Experience	142
	Bibliography	144
A	EventShop Web Application	155
A.1	Getting Started	155
A.1.1	System Requirements	156
A.1.2	Launching EventShop Web Application	156
A.1.3	Log In and Sign Up	156
A.1.4	Application Layout	157
A.2	Register Data Sources Panel	158
A.2.1	Controlling and Searching Data Sources	159
A.2.2	Adding New Data Source	161
A.2.3	Sample of Data Sources	165
A.3	E-mage Panel	167
A.3.1	Visualizing Data Source E-mage	167
A.4	Create and Execute Query Panel	167
A.4.1	Forming Queries using Operators	167
A.4.2	Registering Query	175
A.5	Query Graph Panel	177
A.5.1	Visualizing Query Graph	178
A.6	Registered Queries Panel	178
A.6.1	Executing Registered Query	178
A.6.2	Searching Registered Query	180
A.6.3	Visualizing Output E-mage	181
A.7	Taking Actions	182

B	EventShop Rest API	184
B.1	Data Source	184
B.1.1	Register Data Source	184
B.1.2	Enable Data Source	187
B.1.3	Disable Data Source	187
B.1.4	Delete Data Source	188
B.1.5	View Data Source Metadata	188
B.2	STT Rule Engine API	189
B.2.1	Create Rlue	189
B.2.2	Get Rule	192
B.2.3	Get List of Rules	192
B.3	Search STT and Emage	193

LIST OF FIGURES

	Page
1.1 Situation recognition in cybernetics loop	2
1.2 (a) Growth of standard image size in pixels also follows Moore’s Law. (b) By year 2040, images of the Earth should be available in the resolution of 25 <i>meters</i> ² or better. [91]	6
2.1 Overview of Social Life Networks (SLNs) framework	15
2.2 Overview of EventShop framework	16
2.3 Overview of the Personal EventShop framework [86]	18
2.4 Main components in stream processing framework for situation recognition	19
3.1 From heterogeneous data sources to situation recognition	28
3.2 EventShop Web GUI	30
3.3 An example emage [119]	32
3.4 From heterogeneous STT observation to Emage	34
4.1 Observations from machine-centric sensing over California on Feb 12, 2008. [130]	41
4.2 Quadtree structure for multi-resolution Emage	51
4.3 From low level sensors’ data to high level situation semantic	52
4.4 EventShop system architecture overview	54
4.5 EventShop GUI	55
4.6 EventShop query graph	56
4.7 EventShop situation monitoring dashboard	57
5.1 Archival system and query processing engine	59
5.2 Archival system infrastructure	78
6.1 Spatial granularity in two data models [112]	81
6.2 [79] summarizes formulas to find the right pixel size for different types of data sources.	86
6.3 The interpolated results at one iteration in Monte Carlo. Three different methods are used to interpolate PM2.5 concentration values at four Emage resolutions: (left) cubic interpolation, (center) linear interpolation, and (right) nearest neighbor interpolation.	89
6.4 STD error estimation using linear interpolation method at four Emage resolutions	90

6.5	Rasterize state boundaries into Eimage	91
6.6	Localized error in quad-tree	92
6.7	Error estimation vs Eimage generation response time	93
6.8	EventShop Eimage Stream Executor	94
6.9	Situation representation in 2D space and time dimension. Situations of different scales are not necessarily localized in both time and space.	97
6.10	Flickr photos in Thailand from July 1, 2011 to June 30, 2012. The X, Y and Z axes represent latitude, longitude, and day, respectively.	101
6.11	Photos from a cluster which classified as a flooded area in Thailand on November 5, 2011.	102
7.1	Reports of events from citizens through micro-blogs technologies	106
7.2	From micro-reports to situation recognition	112
7.3	Micro-reports in E-model	113
7.4	Visual analytics tool for Micro-reports: (A) interactive timeline, (B) query control, (C) theme chart, (D) interactive map, and (E) media gallery	114
7.5	Number of photos taken in London	117
7.6	Timeline of concepts	117
7.7	Evolving concepts in Beijing in the year of 2008	118
7.8	Micro-reports of Flickr data	119
8.1	Asthma risk situation model	124
8.2	Snapshots of the asthma risk recommendation app	125
8.3	Three stages in disaster management. Planning and recovery have different levels of urgency than the response stage.	126
8.4	Flood risk situation model	130
8.5	Eimage result and alert tweets sent from EventShop	131
8.6	Flood multimedia micro-reports using Krumb's intents	132
8.7	Dashboard for citizens and city planners	132
8.8	(a) Krumb's Intent capture, (b) submitting report, and (c) Krumb's dashboard	134
8.9	Real-time trash situation from sensorized trash cans and citizen reporting using Krumb's	136
8.10	Predicting trash situations in 30 minutes at a given event's location by combining a trash prediction model based on events history and a current real-world trash situation.	137

LIST OF TABLES

	Page
2.1 Features comparison between this work and related fields of study	26
3.1 Summary of Operators.	35
4.1 Advantages and challenges between geographical-centric and human-centric sensing	42
5.1 Rule operators and value types	72
8.1 Data source configuration for asthma risk situation model	123
8.2 Data source configuration for Thai flood situation model	129
8.3 Data source configuration for trash fill level situation model in EventShop . .	135

LIST OF ALGORITHMS

	Page
1 Error Estimation Algorithm for Eimage Interpolation	89
2 Eimage Stream Source	94
3 Eimage Stream Operator	95
4 Eimage Stream Sink	96
5 Density-grid Based Clustering Algorithms	97
6 Density Eimage Clustering	99

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Professor Ramesh Jain, for believing in me from the beginning. I feel extremely grateful for his constant guidance, generous support, and continual encouragement during my Ph.D. life. He provided me countless opportunities to explore the world and connect with many incredible world-class researchers. His unbounded energy and eager enthusiasm kept me excited and inspired to solve useful and practical problems. It was a great honor working with him, a wise man who is so bold, and yet so humble.

I would like to thank my committee members, Professor Michael Carey and Professor Gopi Meenakshisundaram, for their insightful comments and challenging questions which widen and strengthen my work from various perspectives. I am extremely grateful to Mike for reading through every page of my thesis and providing a very comprehensive feedback.

I would like to thank the pioneers of EventShop, Vivek Singh and Mingyan Gao, who were also my former lab-mates at UC Irvine, for building such a great foundation of the system and introducing me to this innovative research.

I would like to thank my best friend and lab-mate at the Social Life Networks lab, Laleh Jalali. Her diligence reminded me to work harder, her care got me through difficult times, and her humor brought more laughter to my life.

I would like to thank my other lab-mates (Setareh, Hamed, Ish, Arjun, Mengfan, Pranav, Jordan, Eric, and Nitish) and research partners (Minh-son, Kim, Koji, and Amarnath) for their numerous discussions, collaborations, and supports to my work. My special thank goes to Mengfan Tang for being a great collaborator, co-author, and data analytics expert. Thanks to his hard work and commitment, we were able to publish a number of publications during the last few years of my Ph.D. program.

I also would like to thank my sisters, brothers, and friends at UC Irvine who filled my life with happiness, joy, and unforgettable memories. My deep appreciation goes to Phillip Preechakul, Nga Dang, Rosario Cammarota, and Ching-Wei Huang for their true friendship which kept me sane during up and down moments in my grad school life.

I gratefully acknowledge the funding received towards my Ph.D. program from the scholarship by the Information and Communication Technology (ICT) department, Mahidol University, Thailand.

Finally and foremost, I would like to thank my beloved parents, Ms. Tipaporn Tassaneekulkij and Mr. Sukit Pongpaichet, and my beloved brother, Mr. Paiboon Pongpaichet, for their love, support, care, and understanding through the years even they were physically 8,000 miles away. Their sacrifices allowed me to pursue on this path and came this far.

Undertaking this Ph.D. has been a truly life-changing experience for me. It gave me an opportunity to observe and understand the world from a new perspective. This would not have been possible to do without the support from all the people mentioned above.

CURRICULUM VITAE

Siripen Pongpaichet

EDUCATION

- Doctor of Philosophy in Computer Science** **2016**
University of California, Irvine *Irvine, California*
- Master of Science in Computer Sciences** **2011**
University of California, Irvine *Irvine, California*
- Bachelor of Science in Computer Sciences** **2008**
Mahidol University *Nakhon Pathom, Thailand*

RESEARCH EXPERIENCE

- Graduate Research Assistant** **2014–2016**
University of California, Irvine *Irvine, California*
- Research Intern** **2014**
National University of Singapore *Singapore*
- Research Intern** **2012**
FXPal *Palo Alto, California*
- Graduate Research Assistant** **2011**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

- Reader and Teaching Assistant** **2014–2016**
University of California, Irvine *Irvine, California*
- Fundamental Programming and Web Programming** **2008–2009**
Mahidol University *Nakhon Pathom, Thailand*

REFEREED CONFERENCE PUBLICATIONS

- EventShop: recognizing situations in web data streams** **September 2013**
WWW 2013 Companion Proceedings of the 22nd International Conference on World Wide Web
- Situation fencing: making geo-fencing personal and dynamic** **October 2013**
The 1st ACM international workshop on Personal data meets distributed multimedia, PDM@ACM Multimedia 2013
- A Real-time Complex Event Discovery Platform for Cyber-Physical-Social Systems** **June 2015**
ACM International Conference in Multimedia Retrieval (ICMR) 2014
- Observing Real-World Phenomena through EventWeb over Space, Time and Theme** **June 2015**
The 2nd International Workshop on Building Web Observatories, ACM Web Science 2014
- Geospatial interpolation analytics for data streams in EventShop** **July 2015**
IEEE International Conference on Multimedia and Expo (ICME) 2015
- Exploring spatio-temporal-theme correlation between physical and social streaming data for event detection and pattern interpretation from heterogeneous sensors** **September 2015**
IEEE International Conference on Big Data (IEEE BigData) 2015
- Using Photos as Micro-Reports of Events** **June 2016**
ACM International Conference in Multimedia Retrieval (ICMR) 2016
- Situation Recognition from Multimodal Data** **June 2016**
ACM International Conference in Multimedia Retrieval (ICMR) 2016
- A graph based multimodal geospatial interpolation framework** **July 2016**
IEEE International Conference on Multimedia and Expo (ICME) 2016
- Situation Recognition from Multimodal Data** **July 2016**
IEEE International Conference on Multimedia and Expo (ICME) 2016
- Research Challenges in Developing Multimedia Systems for Managing Emergency Situations** **October 2016**
ACM Multimedia Conference 2016
- Situation Recognition from Multimodal Data** **October 2016**
ACM Multimedia Conference 2016

SOFTWARE

EventShop

<http://slnlab.ics.uci.edu/eventshop>

A platform for situation recognition from heterogeneous spatio-temporal event streams

ABSTRACT OF THE DISSERTATION

Situation Recognition from Multi-Resolution Event Streams

By

Siripen Pongpaichet

Doctor of Philosophy in Computer Science

University of California, Irvine, 2016

Professor Ramesh Jain, Chair

The nature of data has changed from the past decades. A large volume of spatial-temporal data has been produced, collected, and shared to observe events in the physical world at different spatial scales and temporal frequencies. A multitude of data streams such as weather patterns, stock prices, social media, traffic information, and disease incidents can be used to recognize evolving real-world situations. These situations vary and affect multiple aspects of people's lives - such as traffic, flash floods, economic recession, and epidemic diseases. Detecting situations in time to take appropriate actions can help in saving lives and resources. Building upon a situation recognition and social network concept, where people can easily connect to people online, the Social Life Network (SLN) concept is introduced. The goal is to connect people with the right resources efficiently, effectively, and promptly, depending on the evolving situations. To achieve this goal, a novel platform for situation recognition is required.

In this dissertation, we propose a generic framework for situation recognition from heterogeneous event streams. First, to ingest and store massive data streams, we implement the archival system using the pub/sub messaging system, Kafka, and big data management system, AsterixDB. Second, to handle data and events at different granularities, multi-resolution processing is introduced in every step including data ingestion, query processing, and data

visualization. To aid users in finding the right resolution, statistical information about estimated error and response time for each situation model at different resolutions are provided. The system can automatically select an appropriate resolution or users can interactively select the most satisfactory resolution according to their needs. The idea that action drives design is a key component in creating a situation model. Finally, to fuse disparate data sources, a micro-reports concept is proposed. These micro-reports are compelling, universal, spontaneous, and objective. They can be used to solve problems of existing micro-blogs and replace them. The applicability of this framework is presented in a healthcare application for asthma risk management, disaster rescue for flood and hurricane mitigation, and smart city innovation for trash management in Downtown Washington D.C.

Chapter 1

Introduction

Cybernetics is about having a goal and taking an action to achieve that goal. This term comes from a Greek word meaning “an art of steering”. Knowing whether the goal is reached or is getting closer requires feedback and communication between the actor and the environment. The cybernetics term was coined by Norbert Wiener, a mathematician at the Massachusetts Institute of Technology, when he published a book “Cybernetics” in 1942 [146]. He pointed out that effective control requires communication, as shown in the sub-title of his book, “control and communication in the animal and machine”. This sub-title also explicitly stated that both animals (humans or any biological systems) and machines (non-biological or artificial systems) can have goals and operate according to cybernetic principles. Recent progress in sensors, Internet of Things (IoT), social networks, actuators, and information systems makes his ideas very relevant in today’s societal systems.

Given the advanced technology in sensor networks, mobile device infrastructures, processing powers, and participatory sensing frameworks, cybernetic principles may play an important role in designing and developing situation awareness systems. Diverse sensors are inexpensive and already being used to collect observation streams at varied locations. In addition, people

become powerful sensors, responders, and actuators in most societal situations. Equipped with smartphone technologies, people can easily collect and report real-time *in-situ* information (in texts, images, videos, or all forms), as well as receiving instructions for taking appropriate actions at the right place. The ultimate goal of these situation awareness systems is to connect people to the right resources efficiently, effectively, and promptly. Situation recognition is the most crucial part of closing the loop in the cybernetic principles. Situation recognition aims to understand and recognize an evolving environmental stage, and then send actionable information to an actuator as shown in Figure 1.1.

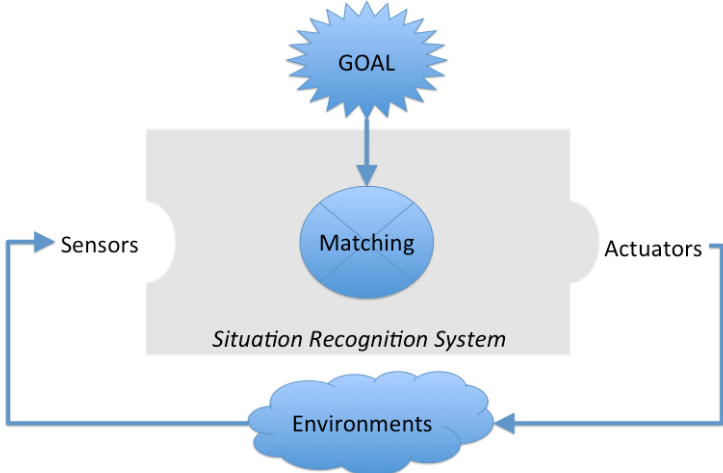


Figure 1.1: Situation recognition in cybernetics loop

In this thesis, we primary focus on situations occurring in the real physical world at the macro level across cities, states, or countries. Examples of situations are natural disasters (floods, hurricanes, wildfires), air pollution, epidemic diseases (Influenzas, Zika virus), economic regressions, political movements, terrorist attacks, and traffic situations. When disasters happen, people in different areas may suffer from different emergency situations such as being trapped in flooded houses, lacking clean water and food, losing electricity, or having insufficient medical aid. To ask for help, people quite often make phone calls or send short messages (e.g., SMS, tweets, and status updates). However, the problem is how to promptly find and connect them to the right resources and agencies who can help. On the other hand,

emergency planners encounter the challenge of finding potential victims and sending out help. To solve this problem, it is essential that we analyze all requests and reports from people who are in affected areas together with a real-time map of the disaster situation in order to take the best available approaches in providing help and preparing more resources at particular times and locations.

1.1 Situation Recognition in Spatio-Temporal Data Streams

Situation recognition is required for deriving actionable insights from heterogeneous, real-time, big multimedia data in order to benefit human lives and resources in different applications. A multitude of data streams such as weather patterns, stock prices, social media, traffic information, and disease incidents are converted into actionable information. Building upon situation recognition and the Social Networks concept, where people can easily connect to people online, the Social Life Networks (SLN) concept was introduced in 2011 [83]. The goal in SLN is to connect *people with the right resources* efficiently, effectively, and promptly, depending on the evolving real-time situations recognized from multiple data streams. To achieve this goal, a novel platform for situation recognition is required. This platform should allow domain experts to explore and build situation recognition models by abstracting high level information from a variety of low level data streams. They should be able to browse through the discovered patterns in any space, time, and theme dimension to obtain deeper insights that support their assessment and decision-making in any domain such as a smart city, public health, and emergency management. By adding different layers of data streams, they can model numerous situations such as disease incidence and severity, and emergency response and impact.

The nature of the data streams has changed over the past few decades, and exhibits several new research challenges. In this chapter, we address three essential data characteristics

and explore some research challenges inherent to building an effective situation recognition platform.

1.1.1 Large volume and high velocity of data streams for real-time processing

In recent years, a large amount of data has been generated and shared everywhere at every moment because of the lower cost of sensors and the rapid development of the ubiquitous computing power, the web, and social networks. The high rate of these available data eventually becomes massive torrent of data streams. Stock market data feeds can generate more than tens of thousands of messages per second. We expect to see a lot of things on the planet get sensor-tagged and report their state and location in real-time. In one minute on the Internet in 2016 [19], on average 1,389 Uber riders are booked, 347,222 tweets are tweeted, 527,760 images are snapped on Snapchat, and almost one million potential suitors are swiped on Tinder.

Making use of these data streams for real-time situation recognition requires very high-volume processing of feed data with very low latency. This requirement is pushing the limits of traditional data processing infrastructure. Instead of handling *archived data* like in the Database Management System (DBMS), the system has to accept and process continuously moving *data streams*. Stonebraker *et al.* [127] identified eight requirements found in most real-time stream processing applications, which include processing data in-stream without storing, supporting high-level “StreamSQL” language with built-in extensible stream oriented primitives and operators, handling stream imperfections (delayed, missing, or out-of-order data), and achieving incremental scalability. However, specialties introduced by the *spatio-temporal* data streams in the situation recognition framework entail some new

requirements, for instance, a tuple-based model may not be suitable for spatio-temporal data analytics.

1.1.2 Data at different granularities in the physical world

A large volume of spatial-temporal data have been produced and collected to observe events in the physical world at different spatial scales and temporal frequencies. In the past, data production was expensive. From visual images, to sensor readings, to telegraphed text, data were produced only when really required. Today, sensors and humans alike keep updating their status continuously once they initiate a broadcast mechanism. With low additional costs for production, there is an abundance and overflow of data. More and more data are produced today without any presumption about their usage characteristics.

At the *global level*, there is a surge of data coming from surveillance cameras, radar systems, Internet of Things, and large sensor networks. At the *personal level*, data is being generated at an accelerated pace from mobile phones, social networks, and wearable devices. Cisco predicted that 50 billion Internet of Thing devices will be connected to the Internet by 2020 [1]. This has led to an explosion in the number of available datasets. Therefore, working with disparate data at multiple scales has become common in many studies.

One of the most challenging areas in big data management is handling the spatio-temporal data collected at different spatial scales and temporal frequencies. Advances in satellite imagery and remote sensing permit scientists to access spatial data at many different resolutions. For decades, geographic information systems (GISs) have made it possible to combine spatial data at different resolutions. However, in most projects, grid resolution is selected without any scientific justification as a background. The challenge in finding the most “satisfying” scale based on data availability and user requirements still remains. This inspired us to create methodological guides to select a suitable grid resolution when designing a model

that combines heterogeneous spatial datasets. At this resolution, the model should assure that the uncertainty in the results is acceptable and the processing cost is minimum or reasonable. Typically, the resolution needs to be improved only to a level after which the model will not necessarily perform better.

The size of grid cells has decreased as computing power used to analyze them has increased. In year 2005, Lagacherie and McBratney [91] analyzed the relationship between the image resolution and processing capabilities of standard desktop over years and discovered a rough relationship between the log of the grid size and the current year as shown in Figure 1.2. The surface of the Earth is about $5.1010^8 km^2$ [147], which means that the coarsest global standard resolution in the year 2015 is about $2 km^2$. If the formula is correct, the coarsest global standard resolution in year 2040 will be about $25 m^2$.

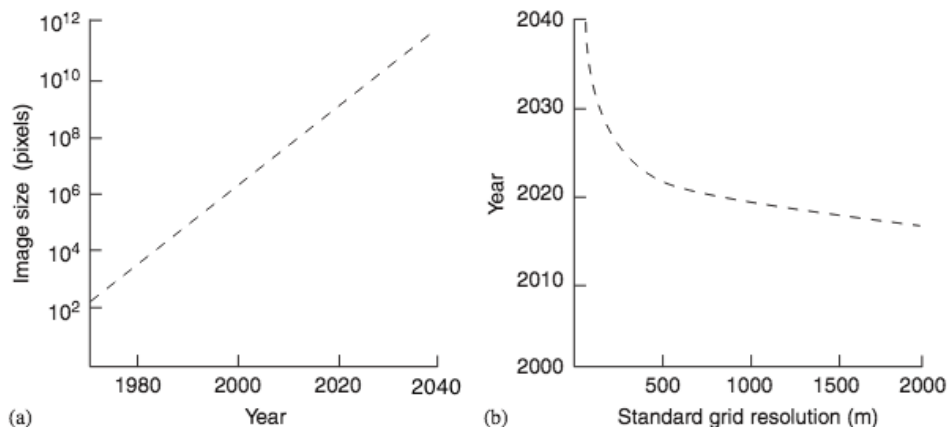


Figure 1.2: (a) Growth of standard image size in pixels also follows Moore's Law. (b) By year 2040, images of the Earth should be available in the resolution of $25 meters^2$ or better. [91]

In time-sensitive applications, the computation or processing time on a large volume of data can take too much time to be useful. For instance, a model that requires two days to complete a 24-hour forecast would not be operationally useful. In decision support systems, if the result is not available within the time requirement, it is useless. In simulation systems, if each iteration takes too much time, then it is not quite efficient. Daily reporting about

flooded risk areas during normal period is acceptable, but during heavy rain season, hourly or more frequent reporting are required to be helpful for people and city administrators who are working on to manage and control the situation.

1.1.3 Disparate and heterogeneous data sources

In the last few years, access to public data from both governmental sources such as NASA (National Aeronautics and Space Administration) and non-governmental sources such as Google has become easier than before. Many emerging applications and social media systems also make their data publicly available through their APIs (Application Program Interfaces). In observing real-world phenomena, we categorize these data sources into two categories. The first data source is traditional *physical sensor data* generated from *in-situ* and remote sensing sensors. These data always contain geo-spatial attributes and meta-data. The second data source is *user-generated content*. People with their smartphones are smart sensors and they publish data in form of text messages, photos, and videos through a variety of social media platforms. The values obtained directly from sensors may range from measured facts (i.e., traffic flow) to opinions reporting by people. In this thesis, we use a term ‘reporters’ to represent these people. The challenge is to deal with these values taken from disparate sensors and reconstruct the state by mapping sensors to attributes that are relevant in the context of computing the particular situation.

The process of extracting data from the disparate sources and making them available for later computation is not trivial. The data are not only generated in different media formats (e.g., KML, JSON, image, audio, video, table, and sensor signal), but the properties of them are also very different (e.g., measuring weather and traffic). The types of data are very diverse, ranging from heart rate signals, to text-based microblogs, to images and video in social media. While semantic web technologies and APIs are trying to make data interoperable,

there is still a lack of a generic data model which can be used to integrate heterogeneous data coming from spatio-temporally distributed web streams. As a result, we are still living in a *data rich but insight poor* world. We need to have a tool in place to bridge the semantic gap between the high level concepts of situations and the low level data streams in order to turn data and information into action.

1.2 Goals and Contributions

In this thesis, we designed and implemented an open-source system for situation recognition in real-time, called EventShop, that can solve many challenges mentioned earlier. A situation in EventShop is defined as “an actionable abstraction of observed spatio-temporal descriptors. [121]” To provide a practical and flexible base for integrating spatio-temporal data streams, we adopt the grid structure and call it Emage (an *event data based analog of image*). Each Emage represents some measure taken over a certain geographical area within a time window. The measure can be evaluated from various heterogeneous data sources, including social sources, such as Twitter, Facebook and Flickr, as well as other geographic map sources like cloud map and pollen count map.

Grid resolution is an important concept in situation modeling in EventShop. It refers to the fineness of detail that can be observed. A spatial resolution specifies how large (in degrees of latitude and longitude or in kilometers or miles) the grid cells in a model are. The amount of detail increases as the grid resolution is higher, thus the accuracy of the result also increases. A temporal resolution refers to the size of the time steps used in models; how often calculations of the various properties being modeled are conducted. This is also related to the data arrival rate.

The choice of space-time resolution in each situation model is purely relied on the knowledge and imagination of users or application developers. This can lead to the problems we mentioned in the beginning. There is no absolute ideal resolution - one size fits all. However, one should at least try to avoid using resolution that do not comply with the inherent properties of available datasets and application needs. To aid users in finding the ‘right’ resolution for their models, this proposed framework provides statistic information about estimated error and response time for each situation model at different resolutions. System can automatically select an appropriate resolution or the user can interactively select the most satisfactory resolution according to their need. One key component in designing a situation model is “action drives design”.

To support spatio-temporal data exploration for situation recognition paradigm, we propose a novel infrastructure of data management system, queries processing engine, and interactive visualization tool. A drill-down approach has been widely used by scientists and data analysts to get insight on the large data sets. Most of the data can be processed and visualized at a relatively coarse resolution, and then the ‘interesting’ subset of the data can be identified and processed at the increasingly finer resolution. We propose a generic multi-granularity platform to process large spatio-temporal data sets under the combination of AsterixDB [28] and EventShop infrastructure. Instead of relying purely on the users’ knowledge or machines’ intelligence in order to select the optimum Emage resolution, we merge these two worlds together to find the most ‘satisfied’ Emage resolution.

The contributions of this thesis are

- Generic and open-source framework for situation recognition from multimodal data streams: This work is not designed to support a specific application in situation recognition; it is a generic or domain independent system. Researchers or experts from different domains can use the system for analyzing heterogeneous geo-spatial and tem-

poral data streams of their choices and exploring various kinds of situation models related to their specific domains. Many types of situation awareness applications can be conducted, including, but not limited to, disaster management systems, smart city applications, and disease outbreak response and support systems.

- Scalable archival system: We integrate publish and subscribe messaging with an AsterixDB platform [28], a full-function scalable BDMS (Big Data Management System), in order to efficiently ingest, store and access both input data and output situations. We follow a similar principle of data integration process in database management, called Extract, Transform, and Load (ETL). Data extraction extracts space, time, and theme data from the data sources, data transformation transforms the data into the STT model data format, and data loading loads the STT data into the data storage.
- Multi-resolution data streams processing: The proposed framework has abilities to extract, ingest, and store spatio-temporal data streams in multi-resolution manner in order to process data at different granularities. It allows users to look at low resolution data sets from a larger region cheaply, before deciding to obtain more detailed and potentially more expensive data sets.
- Novel mechanism for reporting events: To overcome limitations of current micro-blogs which are subjective, ambiguous, and noisy, we introduce, in Chapter 7, a new mechanism for citizens to efficiently report events. These reports are compelling with context around multimedia contents, representing facts instead of opinions, using universal language like photos, and easy to share information.
- Interactive user interface for visual data analytics: Interactive interface allows domain experts to employ their intelligence, creativity, and background knowledge to gain insight from the large volume of data. We provide advanced visual interface where decision makers can directly interact with the data analysis capabilities in order to make well-informed decisions in complex situations.

1.3 Thesis Overview

For building a generic situation recognition platform, this thesis addresses various well-defined research challenges, explores and implements proposed solutions, and demonstrates expectable results of the real-world use cases.

In Chapter 2, we discuss the concept of Social Life Networks (SLN) in further detail. We survey several data streams processing systems from various research fields. Our survey shows that existing works are not truly capable to fulfill the requirements for processing heterogeneous spatio-temporal data streams for situation recognition.

The predecessor system of this work is described in details in Chapter 3. We describes its system architecture, fundamental design principles, data models (STT, Emage, and Emage stream) to capture the nature of spatio-temporal data streams, and primitive set of operators that can be combined to create complex situation models. The strengths and limitations of the system in its present form are also discussed.

Based on the experience of the first version of EventShop, seven research challenges in building situation recognition framework are defined in Chapter 4. We redesign the EventShop system to overcome its limitations and handle some new challenges. The overall architecture of our proposed system is presented including three main components: archival system, multi-resolution processing engine, and micro-reporting. These components are described in more detail in the following chapters.

We present the archival system of EventShop in Chapter 5. This system consists of two main components: data stream ingestor and data store. Heterogeneous data streams are ingested according to their data source's configuration. The data stream ingestor extracts space-time-theme values from the raw data, transforms them to STT data model, and aggregates them

to generate Eimage streams. The outputs from this component are stored in the data store component.

To deal with different spatio-temporal granularities of data streams, we propose multi-resolution stream processing engine. We apply error estimation methods when converting Eimage from its original resolution to a desired resolution. To detect various scales of situations from citizens' reports, a new density-grid based clustering algorithm on STT data streams is discussed. We test it with 100 millions photos from Flickr and show that city flood situations and sport events like the Olympic Games can be detected.

In Chapter 7, we discuss limitations of micro-blogs and show why micro-blog based applications, such as those using Tweets, could be dramatically improved using *micro-reports*. We extended an EventShop framework to import photo-reports as an information stream from people to detect situations and trends by combining it with other sensor data streams.

In Chapter 8, the applicability of the EventShop framework is evaluated in a healthcare application for asthma risk management, disaster rescue for flood and hurricane mitigation, and smart city innovation for trash management in Washington D.C.

Chapter 9 summarizes motivations, challenges, and contributions of this work. There are multiple opportunities for improvement and future research.

Chapter 2

Literature Review

2.1 Social Life Networks

During the first-generation of the Web (Web 1.0) in the 1990s, the focus was primarily on building the Web, and making it accessible by connecting people to online documents. In the second-generation (Web 2.0) in the 2000s, the growth of social networking sites, wikis, communication tools and folksonomies brought a new experience to our society, by connecting people to people. In addition, the emergence of the mobile Internet and mobile devices was a significant platform driving the adoption and growth of the Web. Effectively, the Web became a universal medium for data, information, and knowledge exchange. In the third-generation Web (Web 3.0), the innovation shifted toward upgrading the back-end Web infrastructure making the Web more connected, more open, and more intelligent. The Web has been transformed from a network of separately siloed applications and content repositories to a more seamless and interoperable network. The Web now can be used to establish new kinds of network called “Social Life Networks (SLNs)” [83] by connecting people to real-world (life) resources for decision-making at both individual and societal levels.

The decisions-making can range from daily life problems such as traffic, to emergency situations such as hurricane migration. With the enormous reach of mobile phones, used by more than 5 billion people, SLNs can be used to connect people with life resources even in the remote areas of underdeveloped countries. Real world phenomena are now being observed by multiple media streams which are available in real-time over the Web and increasingly the majority of these have space and time semantics. We use hurricane migration as an example: there are massive volumes of related heterogeneous data streams available on the Web such as hurricane status (NOAA.gov), weather forecasts (weather.com), population demographics (census.gov), rescue shelters (redcross.org), and traffic directions (maps.google.com). Despite the rapid increase of data available on the Web, comprehensive development tools and computational frameworks for effectively combining and processing these available heterogeneous streams are lacking. Solving particular problems, such as giving people who are currently in unsafe areas directions to the nearest shelter, is very difficult to achieve and time consuming by just manually browsing on the Web. To provide appropriate resources to individuals and organizations efficiently and promptly, the most important problem is accurately recognizing the situations happening around them.

Recognizing situations in the right place at the right moment is a key to take appropriate actions in order to save lives and resources. For instance, in the healthcare domain, approximately 250,000 people die prematurely each year from asthma attacks [54], and almost all of these deaths are avoidable. Asthma patients should be notified to avoid the places that can cause their asthma attacks when they are in critical situations. Some of the early examples of the use of SLN to recognize situations that we studied, such as flood alerts in Thailand [71], flu monitoring [120], and determining the new location of business based on product demand [119], clearly demonstrate the importance of situation recognition and the benefits of SLN.

The overview of SLNs is shown in Figure 2.1. There are three main components: (a) Geospatial Situation Detection to recognize evolving global situations, (b) Life Situation Detection to recognize evolving personal situation, and (c) Need-Resource Matcher to recommend actionable information. In our SLN lab at the University of California, Irvine, we have been developing each component as open-source projects called *EventShop*, *PersonalEventShop*, and *PersonalizedAlerts*, respectively. We will provide a brief explanation of each component here. The focus of this thesis is mainly on the EventShop system. We will discuss EventShop further in the next chapter including its original development, usages, and prior research.

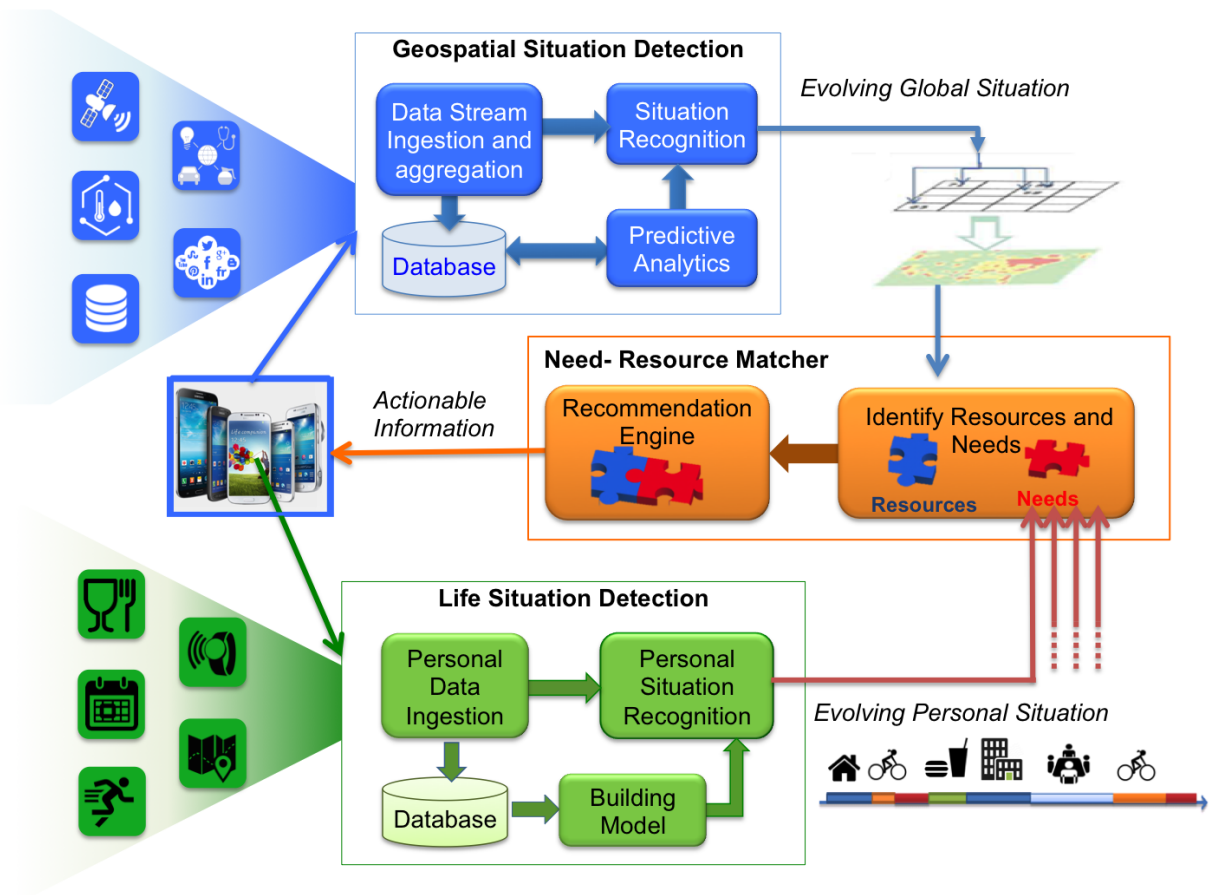


Figure 2.1: Overview of Social Life Networks (SLNs) framework

2.1.1 EventShop

We designed and developed a novel computational framework called “EventShop” to integrate and process heterogeneous Web data streams. The overall framework is shown in Figure 2.2. The data streams over the web (e.g., tweets, weather.gov feeds) are translated into a unified format and made amenable for computing in the next step. Based on application logics, multiple spatio-temporal analysis operators are formed to generate different situation recognition models. The uniform data streams are continuously fed into the models to detect and recognize real-time situations. Then, the detected situations (e.g., ‘flu outbreak’ in New England) can be combined with user parameters (e.g., ‘high temperature’ and location). Again, based on application logic, the situation based controller is constructed to send out personalized action alerts (e.g., ‘Report to the CDC center on 4th street’) to each individual. In addition, the analytic reports are also available to a central analyst who can then make large-scale (state, nation, corporate, or world-wide) decisions.

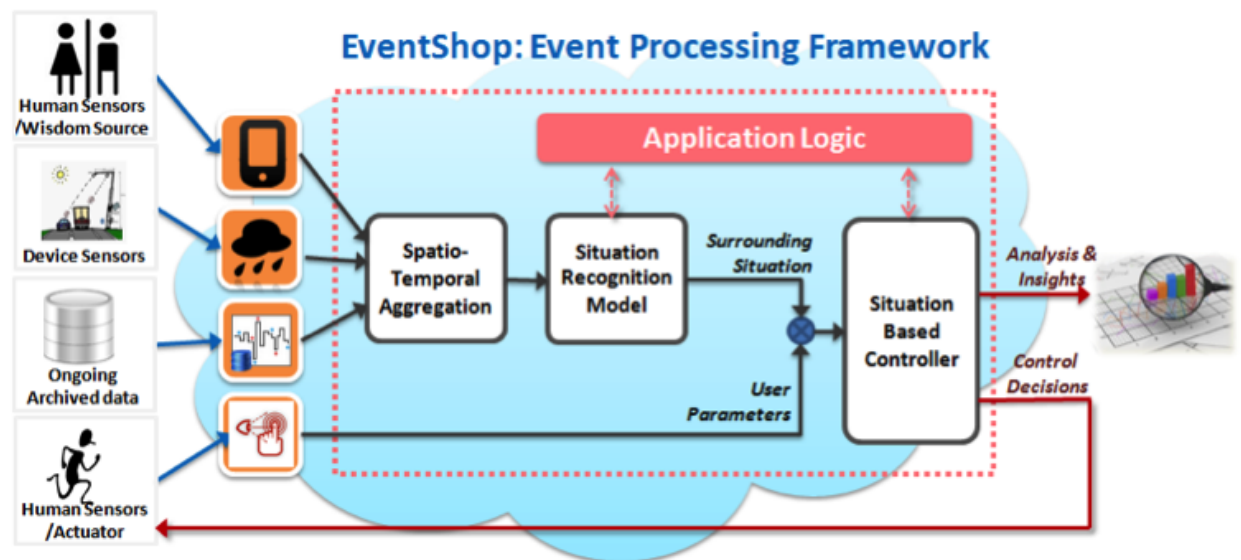


Figure 2.2: Overview of EventShop framework

2.1.2 Personal EventShop

The main goals of the Personal EventShop framework are: (a) providing an architecture to integrate, store, and analyze data from heterogeneous personal data streams; (b) detect low-level physical activities using various unobtrusive sensors embedded in a mobile phone; (c) design a hierarchical classifier that identifies high-level activities with simultaneous use of asynchronous observations consisting of GPS and accelerometer measurements to create a chronicle of life events named Personicle; (d) tools to represent Personicle for a person and derive persona, and (e) visualizing and mining personas to form societal models.

Translating low-level observation from personal data streams into high-level knowledge of a person's evolving situation is a perceptual task and a variety of enabling technologies should be incorporated. Figure 2.3 displays the architecture of a multisensory evolving personal situation recognition system. This architecture has five layers: data ingestion, data processing, visualization, repository, and client. First, the data ingestion layer is composed of various sensors and a sensor fusion component. The sensors range from smartphones, wearable activity trackers, and calendars, to physiological sensors that track vital signs (e.g., heart rate, respiration rate, skin temperature, skin conductivity, ECG). More sensors can be added to improve the performance of our system. The client layer consists an expert and the person/user herself. The user contributes to data collection and the expert's knowledge is used to predefine risky situations for different application domains. For example in case of applying this framework to the problem of asthma management, a medical expert would define a set of situations that might put an asthmatic patient at risk (e.g., a patient being vigorously active more than 15 minutes in a geographical location with a high pollen count). The system would monitor a patient continuously to detect these predefined situations. Also by analyzing the history of a patient's asthma attacks over a period of time, correlations between attacks and local and global factors would be detected.

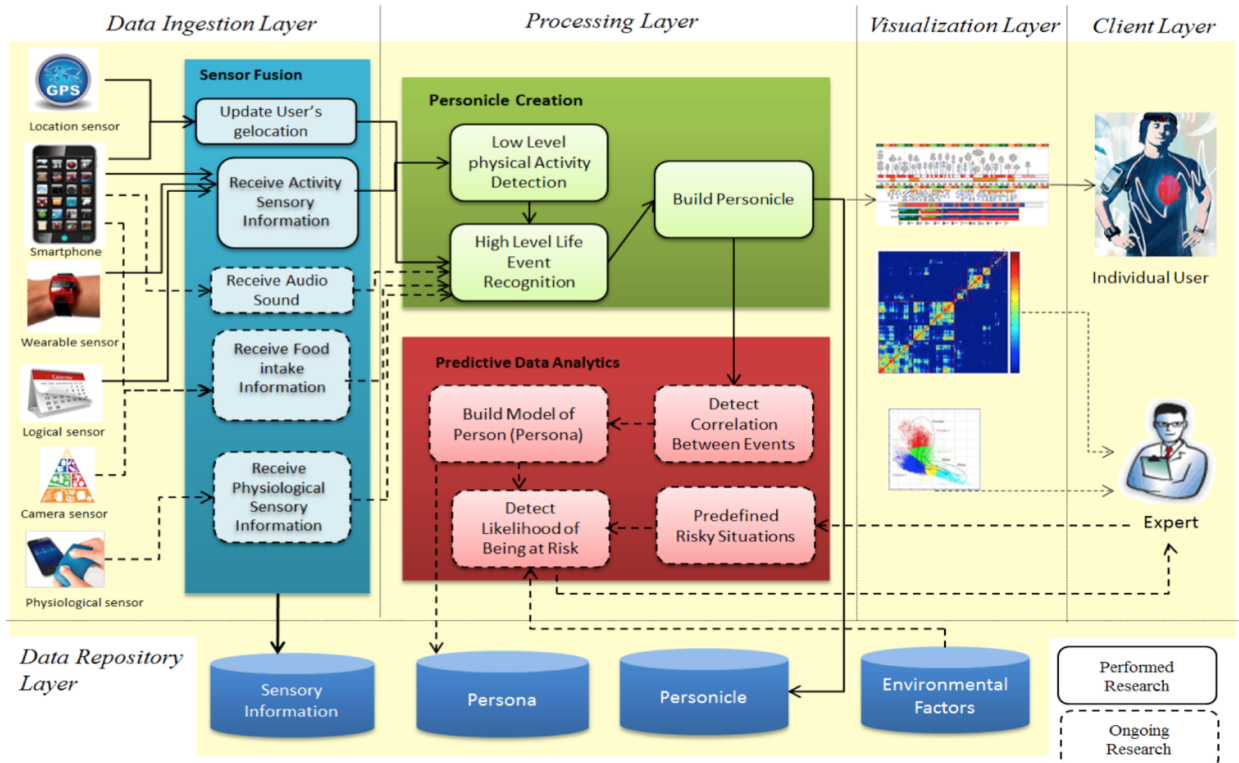


Figure 2.3: Overview of the Personal EventShop framework [86]

2.1.3 Personalized Alerts

Providing the right information, at the right time, in the right place, and to the right people is crucial. In traditional situation recognition frameworks, *macro* scale (e.g., city, state, country level) insights and decisions, such as health warning, weather alerts, and advertisements were made and *broadcasted*. Now, the capability and availability of mobile phones and personal devices allow each user to receive a *personalized* or *unicast* alert based on her specific situation. This framework individually accesses each user's inputs and combines them with her surrounding situation.

2.2 Situation Recognition

The process in situation recognition starts from observing, analyzing, and taking actions. The main components in stream processing framework for situation recognition is shown in Figure 2.4.

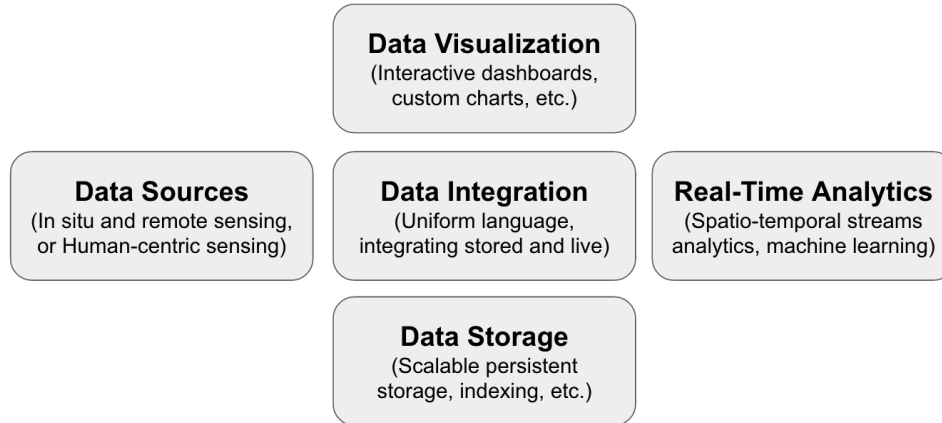


Figure 2.4: Main components in stream processing framework for situation recognition

First, data sources for observing real-world phenomena can be classified into two categories based on their characteristics: machine-centric sensing, and human-centric sensing. On the same phenomena, the former sensory data are related to physical characteristics of a phenomena happening in the real world, while the latter sensory data are associated with high semantic meaning. These two disparate perspective can be assimilated in order to generate a holistic view of the real-world situations. Second, data integration is critical to process heterogeneous data. This process has a root in database and data warehouse. Now, many unsolved research challenges emerge when dealing with data at the web scale. Third, to efficiently archive and manage big data has been an intensive active research area for the past decade. Platforms like Hadoop MapReduce [60] is a well-known name in the big data community. Next, unlike the data storage, to process the data in action, real-time analytics are required. We review some of these platforms in different fields in this section. Finally, data visualization has become a more significant tool to get insight from large scale.

Human’s brains can work much better through charts, photos, and animations. This new research area of visual analytics has gained significant interest from researchers.

2.3 Data Sources and Data Integration

We are witnessing a world of sensor data explosion. All kinds of sensors are constantly observing the state of the world. Physical sensors include, but are not limited to, thermal-based sensors, air quality measuring sensors, inductive-loop traffic detector sensors, surveillance cameras, and satellites. These sensors generate machine-centric sensing data. In the other hand, the rise of human-centric sensing or citizen sensing has changed the way we observe the world. With the technological advances in communications and mobile sensing devices, everybody carrying a mobile device becomes a smart sensor who can sense surrounding situations, express opinions, and report high level semantic information. To realize this vision, Campbell *et al.* [149] developed the MetroSense project which users are the key architectural system component for sensing, learning, and sharing information. This architecture can support a variety of applications which can be categorized into three levels: person sensing, social sensing, and public sensing. The first group focuses on personal monitoring and archiving. The second group combines data shared within a social and specific focused group. Finally, the last group uses data shared with everyone for improving public communities. Several ongoing projects are related to this vision such as CitySense [100] from Harvard, Urban Sensing [126] from UCLA, and SenseWeb [116] from Microsoft Research.

Sheth [117] introduced a system to enhance experience in situational awareness from the integration of machine-sensing and human-sensing. Using Flickr and Twitter, citizens can share their views of the events such as terrorist attacks. The location information in conjunction with semantic annotations from these data provide a rich description of the situation and create trails of various events. In the same direction, Wang [143] proposed a unified frame-

work to assimilate real-time data streams from surveillance cameras and social networks in order to facilitate event detection. To fuse data, visual concept detectors are applied on the raw camera feeds. The detected concepts are constructed as “camera tweets” posted regularly. These tweets are represented by a probabilistic spatio-temporal (PST) data structure. Then, the topics discussed from the geo-located social media data are integrated with the camera tweets to create comprehensive situation descriptions.

2.4 Data Storage

From conventional Database Management Systems (DBMSs), Data Stream Management Systems (DSMSs) are specially designed to manage continuous data streams. DSMSs provide a flexible query processing similar to DBMSs. However, unlike the one-time queries in tradition DBMSs, the queries in DSMSs are *continuous* or *standing* queries [136]. One of the main challenges is to handle infinite data streams using a fixed amount of memory. Two different approaches are proposed including *synopses* and *windows*. The former approach compresses the data points in the data streams to form a synopsis. The algorithms to create synopses ranging from sampling methods for selecting some of data points to summarization using histograms, wavelets, or sketchings. The downside of this approach is that the results from the processing on these synopses can be inaccurate. The second approach is to only process on a portion of the data stream by applying windows boundary into it. There are different kinds of windows such as sliding windows, tumbling windows, and punctuation windows. These windows can be either element-based windows or time-based windows.

To formulate queries, most DSMSs use declarative languages extended from SQL in DBMS such as Continuous Query Language (CQL) [33] from Stanford, StreamSQL from StreamBase [17] (a commercialization of the Aurora project [24]), and Event Processing Language (EPL) from iSpheres [27]. Another approach is a graphical model using boxes and arrows. The

processing step is represented with boxes, and the processing flow is expressed by arrows between the boxes. In DSMSs, the operators are similar to the operators of the relational algebra such as selection, projection, join and set operations. Examples of DSMSs are STREAM (for STanford stREam datA Manager) [32], AURORA [33], Telegraph CQ [49], NiagaraCQ [51], and Odysseus [31]. The limitation of DSMSs is that they usually deal with tuple-based streams. Therefore, Heterogeneous geographic patterns along with the multi-dimensionality of the data in the streams requires more complex operations than traditional ones in relational DSMSs.

2.5 Real-time Analytics

2.5.1 Analytics Toolkit

Several analytic toolkits are available both as open-source software and commercial products. We list some popular tools in data and geospatial data analytics below.

GIS Softwares: The most widely used commercial GIS software for working with maps and geographic information is ArcGIS [4] from ESRI. The most popular open-source GIS software is QGIS (previously Quantum GIS) [15] and GRASS (Geographic Resources Analysis Support System) [10].

Spatial Statistical Tools: MATLAB, a commercial software, provides Mapping Toolbox and several spatial statistical toolboxes. A free software, called ‘R’ project, provides many packages for spatial and spatiotemporal analysis such as ‘spatstat’ for point analysis, ‘gstat’ and ‘geoR’ for geostatistics, and ‘spdep’ for areal data analysis.

SMOA [58]: Scalable Advanced Massive Online Analysis (SMOA) is an open-source software for online data mining on massive evolving data stream. SMOA is both a platform and

library. It has implementations of classification, regression, and clustering algorithms for distributed machine learning on streams. It also provides pluggable feature to port SAMOA to new execution engines via a minimal API.

HStreaming¹: HStreaming is a complete analytics platform built on top of Hadoop and MapReduce. It consists of two main components: data acquisition and real-time analytics. The former collects data in near real-time with ETL ability, while the latter process unstructured and structured data on HDFS in a real-time fashion. In April 2014, HStreaming is acquired by Adello, a mobile ad platform.

2.5.2 Publish/Subscribe Systems

Publish/subscribe (or pub/sub, in short) is a well-know communication paradigm for exchanging information (also called events, or messages) through distributed systems. Pub/sub systems [103, 103, 67] contain, *publishers* that submit information to the systems, and *subscribers* that express their interest and listen to specific channels of information. The main advantages of these systems are (a) anonymity: the systems' participants (publishers and subscribers) do not need to know each other during communication, (b) decoupling in time: they do not need to be up at the same time to communicate, and (c) decoupling in flow: the data processes do not block other participants. Pub/sub systems can be classified into three main categories: *topic-based*, *content-based*, and *type-based* systems [65]. In the topic-based systems such as TIBCO Rendezvous [20], publishers and subscribers exchange the messages through many-to-many distinct (and fixed) logical channels which are created from the set of predefined topics. This concept is easy to understand but it is not flexible and has limited expressiveness on the topics. In the content-based systems such as SIENA [48] and Gryphon [128], on the other hand, allows subscribers to consume information based on the actual content of the considered messages. The subscription scheme represents the

¹<https://gigaom.com/2013/02/14/hstreaming-ready-to-show-the-world-its-real-time-hadoop/>

specific constraints using name-value pair, comparison operators ($=, <, \leq, >, \geq$), logical operators (and, or, etc.), and event correlation [37]. The dynamic channels can be created based on those subscription patterns. Lastly, the type-based systems [66] are inspired by many event-based applications that adopt pub/sub interaction paradigm. In stead of looking inside the content, the *type* of events are used in the basic subscription criteria. The filters can be applied to event types and attributes.

These traditional pub/sub systems have been used in many stream processing applications. However, the expressiveness of the system is very limited.

2.5.3 Distributed Stream Processing Engines (Workflow System)

To address the memory and bandwidth issues in data stream processing on a single machine, *distributed stream processing engines* are introduced. These engines combine the scalability from distributed computing technologies with the efficiency of stream processing algorithms as discussed in many surveys [80, 87]. We mention well-known systems here.

Apache S4 [101]: Apache S4 is designed specifically for managing data streams. S4 apps are designed to combine streams and process elements in real time.

Apache Storm²: Apache Storm is an open-source distributed computation system for data-intensive streaming developed by Nathan Marz from Twitter. It can be used for real-time analytics, machine learning, and numerous applications especially those of high data velocity. These applications are designed as directed acyclic graphs.

Apache Spark³: Apache Spark is an in-memory computing platform. The most important feature of Spark is speed. Unlike MapReduce, Spark operates on the data set in one round

²<http://storm.apache.org/>

³<http://spark.apache.org/>

and produced results in real-time or near real-time. It also supports both acyclic and cyclic data flow.

Apache Samza⁴: Apache Samza is a distributed stream processing framework based on Apache Kafka [3] and Hadoop YARN [141]. Unlike other stream processing frameworks, Samza does not have its own protocol for transferring data between operators. It adapts publish/subscribe messaging service from Kafka. Topics in Kafka are used as inputs and outputs for Samza jobs.

2.5.4 Geo-spatial Stream Processing

Spatial Database Management Systems: Many commercial databases support spatial data such as Oracle Spatial and DB2 Spatial Extender. PostGIS is a widely used open source spatial database management system.

Spatial Big Data Platform: To support scalable spatial analysis, many spatial big data platform have been developed such as ESRI GIS on Hadoop, Hadoop-GIS [26], and Spatial-Hadoop [63]. These provide distributed systems for geometric-data (e.g., lines, points and polygons) including geometric indexing and partitioning.

Geospatial Image Stream Processing: GEOSTREAM [114] provides scientific work-flows for change detection in environmental geospatial image streams. SHAHED [64] is a MapReduce-based system for querying and visualizing spatio-temporal satellite data.

Most of these systems usually process on homogeneous type of spatial-temporal data. To detect the real-world phenomena, we need to deal with heterogeneous data.

⁴<http://samza.apache.org/>

2.6 Summary

For the literature review in this chapter, we finalize seven key features that are required for detection real-world situations. They are (1) processing on heterogeneous data, (2) supporting social sensors, (3) having real-time analysis, (4) handling data at multi-resolution scales, (5) having an efficient reporting mechanism, (6) having scalable data storage, and (7) providing interactive data visualization and analytics. We found that existing works are incapable to achieve all of these features. Table 2.1 compares and summarizes works from various domains. In this thesis, we present EventShop, a generic platform for situation recognition from heterogeneous event streams. It provides all the key features and more.

Table 2.1: Features comparison between this work and related fields of study

Area	Heterogeneous Data	Social Sensors	Data Streams	Multi-resolution Processing	Reporting Mechanism	Historical Data	Interactive Data Analytics
Situation Awareness	✓	–	○	–	○	✓	–
Complex Event Processing	✓	–	✓	–	○	✓	○
GIS	✓	○	–	✓	–	✓	○
Spatial Statistical Tools	○	✓	–	–	–	✓	–
Data Stream Processing	✓	✓	✓	–	–	○	✓
Map Visualization	○	○	–	○	–	✓	–
<i>This Work</i>	✓	✓	✓	✓	✓	✓	✓

Note: ✓ = full support, ○ = partial support, – = not available

Chapter 3

EventShop Fundamentals

The EventShop system was built in 2011 as an academic research project. The pioneers of this system were Vivek Singh and Mingyan Gao. With support from technology companies such as HCL¹ and ABOVE², EventShop has become an open-source software since 2013. Several research collaborations have occurred since then, including Cyber-Physical Cloud Computing systems [56, 85, 57] with National Institute of Standard Technology³ (NIST) and National Institute of Information and Communication Technology⁴ (NICT).

In this chapter, we provide a fundamental background of the EventShop System. We review its key advantages comparing to other systems. Then, we address some limitations that we found after using EventShop to detect real-world situations in many applications. A complete explanation about the system is available in a book, called “Situation Recognition Using EventShop” [122].

¹<http://www.hcltech.com/>

²<http://www.above-inc.com/>

³<http://www.nist.gov/>

⁴<http://www.nict.go.jp/en/>

3.1 General System Architecture

There are four main components in the EventShop framework which are **data ingestor**, **stream processing engine**, **personalized alert**, and **front-end user interface**. A workflow of moving from heterogeneous raw data streams to actionable situations is shown in Figure 3.1. We argue that real world entities and events can be easily described along three dimensions: spatial, temporal, and thematic. An example of an event includes “the wildfire near east of Los Angeles grows out-of-control on August 8, 2016.” The spatial dimension described *where* the event occurred (Los Angeles), the temporal dimension describes *when* the event occurred (August 8, 2016), and the thematic dimension described *what* occurred (out-of-control wildfire).

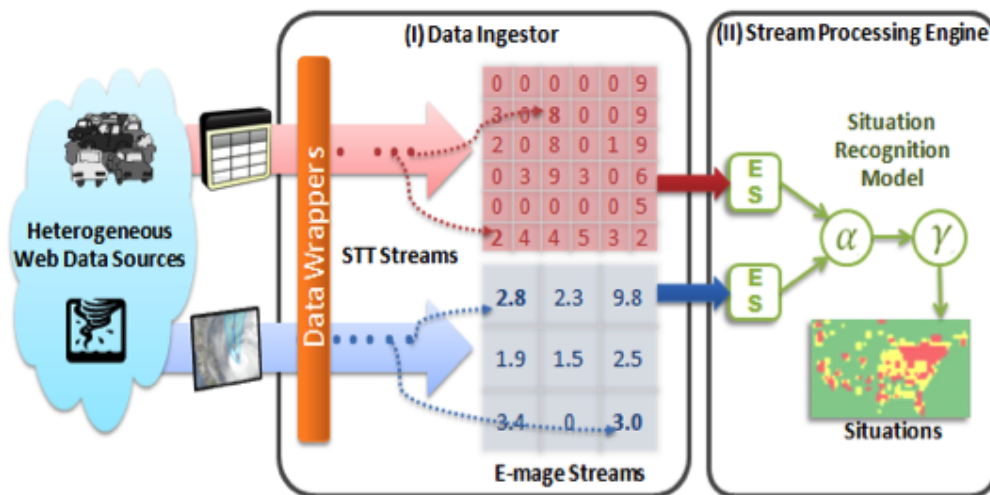


Figure 3.1: From heterogeneous data sources to situation recognition

In the data ingestor component, original raw spatio-temporal data are translated into unified STT (Space-Time-Theme) format along with their numeric values using an appropriate data wrapper. Based on users’ defined spatio-temporal resolutions, the system aggregates each STT stream to form an *Emage* stream which we will describe in more detail later in this Section. *Emage* (from *Event Image*) was first introduced in [119]. These Emage streams are then transferred to the stream processing engine component for processing. Based on

situation recognition model determined by the domain expert, appropriate operators are applied on the Emage streams to detect situation. The final step is a segmentation operation that uses a domain knowledge to assign appropriate class to each pixel on the Emage. This results in a segmentation of an Emage into areas characterized by the situation there.

Once we know the situation, appropriate actions can be taken. If the state of individual is recognized, these actions can also be tailored and sent to individual through the personalized alert unit. For example, in the flood rescue application, system can continuously monitor flood risk situation and automatically send an alert to people who are in the area with the high flood risk level. This alert is also personalized based on the current location of the people to include the nearest shelter location around them.

Inspired by the user-friendly GUI of PhotoShop which allows users to experiment on their photos, the front end interface of EventShop allows application developers and domain experts to explore available data streams, register new data stream sources, formulate queries by combining a rich set of built-in operators, and set up personalized actionable alerts. Figure 3.2 shows the web front end GUI of EventShop which consists of five main components as follow:

- (a) **Data Source Panel:** displays a list of data sources that registered with the system. Users can click on each data source to view its description and its recent Emage created by the system. Users can also register a new data source into the system.
- (b) **Operators Panel:** allows users to form a query by applying different operators on any of the data sources, as well as allow them to configure action control to send out alerts. The built-in spatio-temporal operators are categorized into seven groups as mentioned in Section 3.2.4.
- (c) **Intermediate Query Panel:** shows a textual representation of the intermediate query currently being composed by the user.

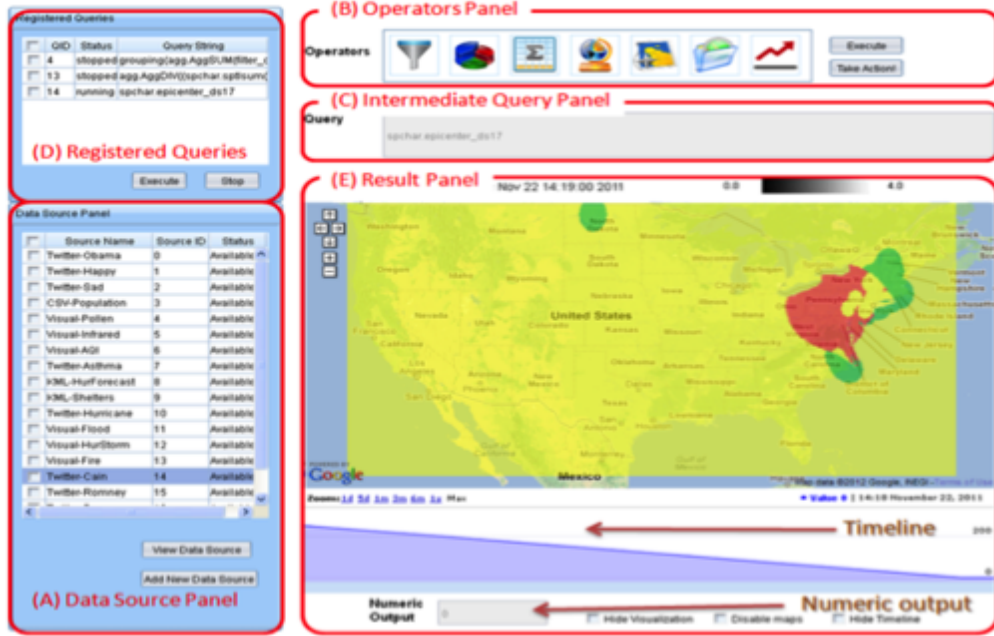


Figure 3.2: EventShop Web GUI

- (d) **Registered Queries Panel:** displays a list of standing queries that registered with the system. It allows users to start and stop the query processes.
- (e) **Result Panel:** displays the recent output of the running query, Emage and stel, which can be presented on a map, timeline, numeric value output, or a combination of them.

3.2 Concepts and Data Structure

The fundamental nature of the data streams observing real-world events is always associated with a time and geo-spatio metadata. This commonality between different streams motivates our new data model. There are many existing approaches for modeling these data; for example, using Semantic Web data models [118, 106], Spatiotemporal Database [142], and GIS Spatio-Temporal Modeling [148, 104]. Given the geo-spatial continuity, we believe that a spatial grid structure is naturally suitable for representing various geo-spatial data, where each cell of the grid stores value of certain measurement taken from the corresponding geo-

location. We adopt the grid structure, and call it *Emage* (an event data based analog of image). We believe that this generic data model can be used to integrate heterogeneous data coming from spatio-temporally distributed web streams.

For handling data streams, special attentions need to be paid to the semantics of aggregated data. Since the data streams are unbounded, combining such a data to find simple aggregated value such as summation and average is unclear. This problem is normally resolved by introducing windows which transform an unlimited sequence of data streams into overlapped or non-overlapped windows of data. In this work, we use tumbling [46] window that splits data streams into non-overlapped contiguous windows.

3.2.1 Data Unification Model

We introduce a new data structure called *Emage*. Each *Emage* represents some measure taken over a certain geographical area in a time window, and the measure can be evaluated from various heterogeneous data sources, including social sources, such as Twitter, Facebook and Flickr, as well as other geo-graphical sources. For example, we collect tweets about a given topic, e.g., “Obama”, from Twitter. Then for each time window, an *Emage* can be constructed, each cell of which represents aggregated number of tweets regarding this topic coming from the corresponding geo-location. An example *Emage* representing interest level amongst users across mainland US in terms of number of tweets containing the term “iphone” on 11th Jun 2009 is shown in Figure 3.3.

Although the use of a single number in each cell to represent a measure seems a very simple model, combining multiple such *Emages* that characterize different measures of an event could potentially generate very insightful results and discover interesting phenomena hidden behind. This approach can capture the semantics and the notion of spatial neighborhood very elegantly, and geography-driven-joins [81] between data reduce to simple overlaying of grids.

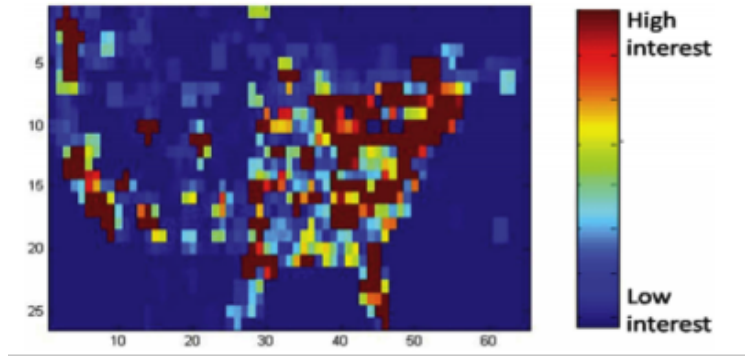


Figure 3.3: An example emage [119]

Humans are quite used to seeing satellite images and GIS data on similar interfaces, so they can understand such data much better than any text-centric representation. Emage allows intuitive visualization and hence aids situation awareness for a human user. In addition, computation on grid-like structure can efficiently speed scale up in several ways, including parallel computing with GPU, and spatial index (i.e., R-tree [76], quad-tree [68] and KD-tree [40]).

3.2.2 Data Representations

All observed spatio-temporal data stream coming into the EventShop framework are converted to, and represented in a common *Space, Time, and Theme (STT)* format. A STT Observation is represented as:

$$STTObservation = \langle latitude, longitude, timestamp, theme, value \rangle$$

A flow of STT Observation becomes a STT Stream:

$$STTStream = \{STTObservation_0, \dots, STTObservation_i, \dots\}$$

The space is captured in *latitude* and *longitude*. The time is *timestamp*. The *theme* is a description of what this data observed. The *value* is a single numeric value. Values in STT Observations collected in a particular time window over STT Stream can be aggregated to form a two-dimensional data grid. The data grid together with related STT information is called Emage, represented as:

$$Emage = \langle SWCoord, NECoord, latUnit, longUnit, timestamp, theme, 2DGrid \rangle$$

SWCoord and *NECoord* are the southwest and northeast spatial coordinate to which the value at the bottom-left and the top-right cell in the 2D grid corresponds. *latUnit* and *longUnit* specify the actual spatial granularity distance, such as 40 miles or 0.1 unit of the width and height of each cell in the grid. The *timestamp* of an Emage is the end time of the time window over which the STTs are collected, which is normally equal to the timeStamp of the last STT in the time window. A flow of Emages forms an Emage Stream:

$$EmageStream = \{Emage_0, \dots, Emage_i, \dots\}$$

Building block of the 2D grid in an Emage is a single cell. The cell together with STT information is called *stel* (spatio-temporal element),

$$stel = \langle SWCoord, NECoord, latUnit, longUnit, timeStamp, theme, value \rangle$$

A stel is a special Emage, where the size is 1 by 1. Some operators take normal Emage stream as an input and generate an output of this special Emage stream, called stel stream:

$$stelStream = \{stel_0, \dots, stel_i, \dots\}$$

3.2.3 From Heterogeneous Data Streams to Emage Streams

Apart from various media formats of the data, we can classify spatio-temporal input streams into three categories based on their behavior. First, the simplest and the most common type is *1D (or point) STT Observation*. Each observation measures a property in the real-world at a particular point in time and geo-location. Example of these data points are tweets and *in-situ* sensor signals. For each time window, the number of STT observations is collected to generate an Emage. Each STT Observation is spatially mapped to a cell in the Emage based on its spatial coordinate and the Emage's resolution as shown in Figure 3.4. The value at the cell is normally the sum of values of all the STT Observations mapped to the cell. Depending on the applications, different aggregation functions can be applied such as count, min, max, and average.

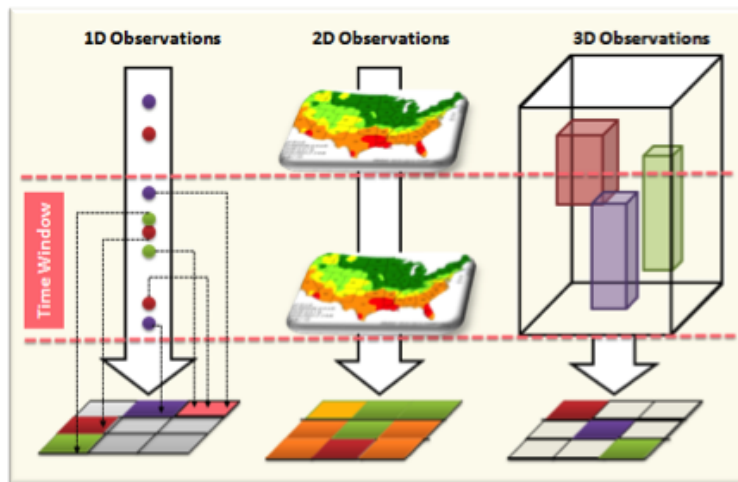


Figure 3.4: From heterogeneous STT observation to Emage

Second category is *2D STT Observation*. The values of the observations are already aggregated over time and space. This type of data is generally found on the geo-images (i.e., pollen count from <http://pollen.com/images/usamap.gif>), maps, and most of the KML data format. In this case, the value at a cell in an Emage is computed from the original or normalized values at the pixels of the original geo-image that are projected to this cell. Computation of projected area depends on the geo-coordinate projection system.

The last data type is *3D STT Observation*. Unlike 1D STT, each observation captures the real-world property for a period of time at a region (not a point). Thus, each cell in an Emage often consists of multiple values from many STT Observations. The interpolation and convolution techniques to constructing the cell from the set of data observations are required [107].

3.2.4 Situation Recognition Operators

We analyzed situation recognition models across multiple domains and defined the initial set of operators, which are generic enough to capture most of the common requirements. We expect to keep enriching this set as this framework gets applied to more applications. Note that these operations are used to find out interesting Emage through describing their characteristics, rather than to manipulate them directly. Therefore, these operations are more declarative than the underlying operations in image processing. The original operations include *selection*, *segmentation*, *aggregation*, *spatialcharacterization*, *spatialpatternmatching*, *temporalcharacterization*, *temporalpatternmatching*.

The summary of supported operators is shown in Table 3.1. More details about the operators and how to use them can be found in the [122] book.

<i>Operators</i>	<i>Input</i>	<i>Output</i>
Selection	Emage Stream	Emage Stream
Segmentation	Emage Stream	Emage Stream
Aggregation	K * Emage Stream	Emage Stream
Spatial Characterization	Emage Stream	Stel Stream
Spatial Pattern Matching	Emage Stream	Stel Stream
Temporal Characterization	Stel Stream	Stel Stream
Temporal Pattern Matching	Stel Stream	Stel Stream

Table 3.1: Summary of Operators.

3.3 Advantages

EventShop was the first framework that allows assimilating multiple geospatial-temporal data streams in order to recognize real-world situation or phenomena. The first prototype of EventShop showed its potential in handling heterogeneous data streams by using a novel Emage data structure. By discretizing data streams into the same spatial and temporal resolution (Emage parameters), the heterogeneity in data sampling of the original data streams could be handled. To recognize complex situations, several Emage operators were introduced and implemented. Many applications from various domains have used EventShop to recognize situations in several countries including wildfire detection in USA, flu epidemic in USA, asthma and allergy awareness in USA and Japan, and disaster rescue in Thailand, India, and USA. We summarize its functionality and advantages below:

- A data ingestor to obtain data from disparate sources, and transform it to the Emage format.
- A set of operators to select, segment, aggregate, characterize, spatially and temporally filter, and cluster spatio-temporal data streams.
- A query processing engine to process ‘standing queries’[108] created using stream processing operators.
- Map visualizations of input and output streams.
- A rule based action engine that can execute tasks depending on detected situations.

3.4 Limitations

After using EventShop in several real-world applications, we found a number of limitations. This motivated us to enhance the system and build a new generation of EventShop presented in this dissertation. Some of the challenges are:

- Ad-hoc data ingestion: In the data ingestion process, some of the data wrappers were restricted to a specific data scheme. For example, the data in CSV form must contain only three columns representing latitude, longitude, and value respectively. The order of the columns is significant. So many times we had to manipulate the structure of the data input before sending to EventShop or to create an ad-hoc data wrapper for specific type of data source.
- No archival system to store raw data streams and results: EventShop was designed to process on real-time and current data stream. Data streams are temporarily stored within a given time window then they are discarded. In many cases, historical data are necessary in building more accurate model. There is a need to access old data, therefore, an archival system is required.
- One resolution per data source: Each data source in EventShop is discretized into only one Emage stream with a specific resolution. This resolution is defined by the users when they register the data source. This resolution has to be the same across all data sources used in the situation model. However, different application may require different granularity of the same data source. Without multi-resolution support, it is difficult to share data sources on different situation models.
- Only support numeric value in data streams: In the STT data model, the value has to be single numeric value. In reality, data sources may contain sophisticated theme

value in text such as keywords, categories, or tags. One data source can also measure a list of values both in structure and semi-structure format.

- Emage resolution is usually selected without any application-specific justification. Creation of methodological guides to select a suitable grid resolution when designing a model to combine heterogeneous spatial data sets for a specific application is a new research problem. At this resolution, the model should assure that the uncertainty in the results is acceptable and the processing cost is minimal or reasonable.

Chapter 4

Research Challenges

We are living in a Big Data era where data can be generated, collected, and exchanged by anything and anybody - anytime and anywhere. Although, a large number of data are available, a small number of knowledge and insight can be derived from them. This leads to a need of developing a new tool and framework in order to deal with emerging challenges. In this chapter, we first discuss the characteristics and sources of sensory data observing real-world phenomena. Then, we address essential research challenges for recognizing situations from these data streams. Finally, the proposed framework is introduced.

4.1 Observing Real-World Phenomena

A very important aspect of the real-world is that it is dynamic. It is always evolving. Most of the new data is collected to capture the dynamic nature of the world. The dynamic data about the past can be used to model the world. One can then use those models to understand the world and more importantly predict future phenomena. Collectively, we are creating a

global data warehouse that involves massive number of heterogeneous data streams that must be analyzed in real-time for recognizing evolving situations and managing them.

In this section, we describe different types of data sources for observing real-world phenomena and the challenges in handle these data sources. We classify the data sources into two categories based on their characteristics: machine-centric sensing, and human-centric sensing. On the same phenomena, the former sensory data are related to physical characteristics of a phenomena happening in the real world, while the later sensory data associated with high semantic meaning. These two disparate perspective can be assimilated in order to generate a holistic view of the real-world phenomena. In other world, we believe that by finding spatio-temporal-theme correlations between these two different sensory data types, situations can be recognized more accurately in real-time [110].

Machine-centric Sensing

Machine-centric sensing can be categorized into two groups; *in-situ* and remote sensing. *In-situ* sensing is the acquisition of information about an object or phenomenon from “on site” or “in position” observation stations. In contrast, remote sensing does not require a physical contact with the object such as signals from aircraft or satellites. The observation from these data sources comes from a large sensor networks. In traditional sensor networks, the emphasis has been on autonomous system operations with the limitation of humans interaction. The role of human is limited to being end-consumers who are using the data and information for making decision. Some examples data sources are shown in Figure 4.1.

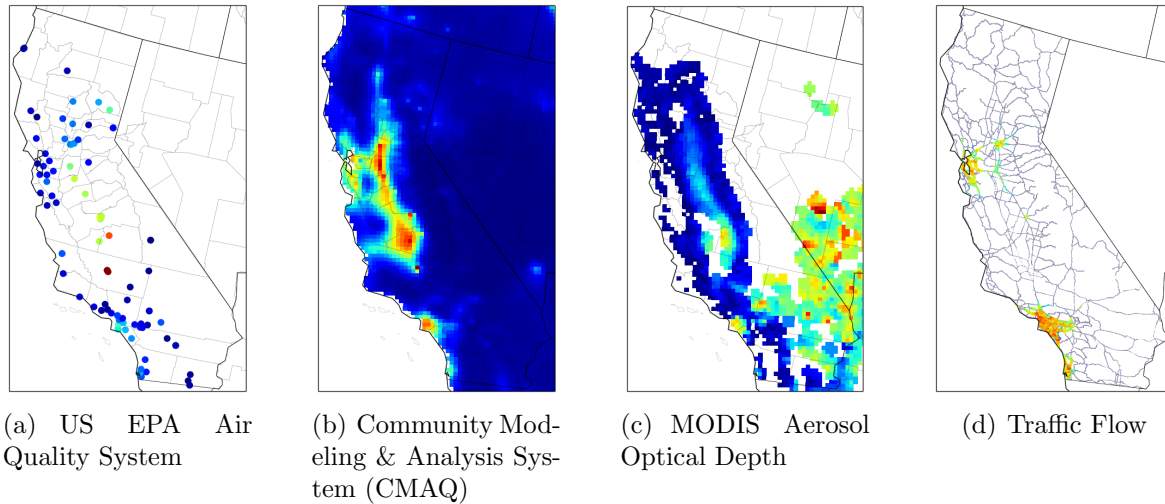


Figure 4.1: Observations from machine-centric sensing over California on Feb 12, 2008. [130]

Human-centric Sensing

With the availability of advance mobile computing technology, fast Internet connectivity, and many popular social web services, humans have now become one of the most important sensors in observing real-world phenomena. Nowadays, millions of users can generate information with attaching photos, videos, or short messages in real-time. Human involvement is particularly useful in *sensing* various processes in complex personal, social, and urban space where traditional embedded sensors are lacking. In the GIS research community, Goodchild [73] introduced the term *Volunteered geographic information (VGI)* in 2007 and described citizens as a network of human sensors with over “6 billion components, each an intelligent synthesiser and interpreter of local information.” Integrating data from human as sensors with crisis maps has been a powerful tool in humanitarian assistance and disaster relief. Several projects have been developed to integrate these data; for example, OpenStreetMap[78, 77], GeoChat[9], Ushahidi [22], Crowdmapping[7], Sahana[16], and Tomnod[21].

The advantages and challenges for processing these two types of data sources are summarized in the Table 4.1.

<i>In-situ</i> and Remote Sensing		Human-centric Sensing	
Advantages	Challenges	Advantages	Challenges
Good at symbolic processing	Extract high level events semantic	Good at perception, common sense, and experience	Information propagation, data reliability, and trust assessment
Good at continuous and long-term sensing	Handle gaps in spatio-temporal coverage	Humans can contextualise, discriminate, and filter	Participant recruitment challenge
		Cover the local area at a finer resolution	Privacy challenge

Table 4.1: Advantages and challenges between geographical-centric and human-centric sensing

4.2 Research challenges

Based on experience of users from National Institute of Standard Technology¹ (NIST) and National Institute of Information and Communication Technology² (NICT) in using the first version of EventShop and our study of the real use cases, we identify essential seven research challenges of building situation recognition as shown below:

- 1 Massive Geo-spatial Stream Processing:** Traditional data processing techniques considered data streams as a sequence of data items. Increasingly, Data is collected by sensors with geo-spatial attributes and meta-data. These heterogeneous geo-spatial data streams must be combined to detect emerging situations. The processing engine has to be able to handle fast data flowing into the system as continuous streams. Unlike traditional database or batch processing engine, EventShop needs to deal with standing query over continuous streaming data. The spatio-temporal data streams provided by different data sources are not only heterogeneous in terms of media, but also often

¹<http://www.nist.gov/>

²<http://www.nict.go.jp/en/>

available at different spatio-temporal bounding boxes and granularities. Approaches that can handle such differences among streams need to be designed.

2 Heterogeneity of Data Sources: The value obtained from sensors may range from measured facts to expressed opinion. Methods to transform data to information and the reliability of information could be entirely different for different sensors. Physical sensors provide either direct measurements (as in a thermometer) or indirect measurement (as in pollen concentration level: high, medium, and low). People with their smartphones used as smart sensors in participatory sensing usually provide opinions, not measurement. The challenge is to deal with these values taken from disparate sensors and reconstruct the state by mapping sensors to attributes that are relevant in the context of computing the particular situation.

3 Situation Recognition Model from Historical Data: Cameras and sensors are deployed at different locations to continuously monitor target conditions, for example, traffic flow, air pollution, and temperature. When such data streams are combined to provide actionable information, discovering the spatially correlated and frequently evolving patterns is challenging: First, informative patterns are often flooded by trivial fluctuations. Second, the pattern search space is extremely large. A pattern can contain an arbitrary number of sensors, and the matching time intervals are at different scales. This results in an exponential pattern search space, calling for novel and efficient pattern search methods. This data may offer a novel challenge for current machine learning techniques. Situation recognition may very well be the next major research challenge for machine learning after recent great progress in concept recognition in images.

4 Data Quality: In the real world, sensors are spatially sparse, preferentially located and often lack complete measurements in targeted temporal resolution. Quality of data affects the quality of situation detection. Reliance on these measurements not

only restricts the geographical region of situation recognition, but can also result in temporal uncertainty. There is a great need for spatial and temporal interpolation to increase the availability of these data across space and time. One of the challenges is to combine data sources that are at different resolutions. For example, one data stream is provided as grid cells, while other data streams are only available at point locations. One data stream is available in hours, while the other is in days. Data missing and misalignment offer another research challenge. The research challenge is how to effectively associate data at different spatial and temporal resolutions and use them to create dense Emage representations. By improving quality and density of data, the quality of situation detection can be improved.

5 Human in the Loop as Sensors and Actuators: Participatory sensing utilizes the eyes and ears of citizens to collect information about a situation. Twitter introduced micro-blogging that has been widely used as participatory sensing. However, Twitter is too broad and subjective for most applications [38]. For the changing demographics of the society and now significantly more powerful smartphones, it is possible to create more powerful multimedia participatory sensing approaches using multiple sensors and other contextual information. A very interesting and potentially transformative research challenge is to use computer vision and contextual reasoning to capture powerful micro-reports that will rapidly identify people's need in given situations and help them get to resources. Citizens not only report micro-events which are fed to the situation recognizer, but they also receive actionable information from the system to help them making decisions. How to communicate successfully and instantly with potential actuators is also challenging.

6 Graphical User Interface (GUI): Big data has created significant interest around analytics and visualization of data. Advanced technology in “visual analytics” research supports analytical reasoning in big data by closely coupled human and machine analy-

sis. To be able to reach conclusions from a combination of evidence and assumptions, human judgments are required. The challenge is to provide a graphical user interface (GUI) that is user-friendly and encouraging for end users to easily experiment with their data.

7 Predictive Analytics: Recognition of situations in real-time is good, but it is much better if the situations are predictable especially in life threatening situation. One example is the use of multimodal data and multimedia modeling tools to predict an area that might face future flooding [12]. An accurate prediction of the severity and disastrous flooding will significantly reduce the damages.

In this thesis, we address all the challenges, but minimally discuss Challenge (7). The first two Challenge (1) and (2) are explored in this chapter and the following chapter. To solve Challenge (3), we proposed a scalable distributed archival system described in Chapter 5. Then, multi-resolution processing is proposed in Chapter 6 to deal with Challenge (4). Issues and solutions of Challenge (5) are discussed in Chapter 7. We also present an early user interface for solving Challenge (6).

4.3 Framework Overview

For the research challenges mentioned above, we redesigned and re-implemented EventShop to handle those new challenges. EventShop framework has emerged from a marriage between the several erstwhile areas of event-based computing, Geographical Information Systems (GIS), web service choreography, semantic web, and mashup tools. Multiple recent efforts have looked at analyzing social and sensor data for understanding the relevant situations. Data Stream Management System (DSMS) and Complex Event Processing (CEP) systems, e.g., Rapide [96], provide sophisticated support for data stream analysis but typically focus

on data in cyber space only. The main applications of these systems have a different focus from our system. The Geographical Information System (GIS) provides handful operators on spatial analysis, but offers limited temporal analysis on streaming data.

There also has been a growing interest in web service choreography [34] and semantic web services [97]. However, most of these efforts focus on well-structured (typically ontological) data. We, on the other hand, make the unified representation extremely simple (space-time-theme: STT), which allows many web streams to integrate easily. Semantic web technologies have been trying to make content on the Web meaningful. Their analytics tools have primarily focused on theme or entity relationships, but offer little support for the analysis of spatial and temporal relationships which are often the critical components for recognizing situations.

To extract data from web pages, several techniques have been practically used, including screen scraping (e.g., HTML Parser) and the deep web [41]. The former aims to extract the content from the surface web pages, while the latter tries to access the content on invisible or dynamic web pages. Mashup and domain specific mashup tools (e.g., MashArt [55], <http://pipes.yahoo.com>) also try to integrate data from multiple web contents. However, most web mashups remain very shallow (only visual integration) and do not provide sophisticated (e.g., geo-temporal) analysis capabilities. Efforts like Yahoo Query Language, Google Tables, Yahoo Pipes [23], and MashArt provide easy, modular computational tools for users to integrate and analyze web data. Following a similar lead, our ultimate goal is to provide a computational framework for sophisticated spatio-temporal analysis of heterogeneous web streams to undertake situation based control.

4.4 Data Model

In the previous chapter, we explained the original data model used in the first generation of EventShop. In this section, we redefine the existing data model to support more variety of data sources. We also introduce a multi-resolution Emage data structure in order to process data at different levels of granularity. Here are definitions of the new data model used in this thesis.

STT Observation

The STT data has three dimensions, and each input data record is defined within the space

$$S = S_{space} \times S_{time} \times S_{theme},$$

where S_i is the definition space for the i^{th} dimension. We partition each dimensional space S into **density grids**. Suppose for each dimension, its space S_i , $i = lat, lon, time$, and $theme$ is divided into p_i partitions as

$$S_i = S_{i,1} \cup S_{i,2} \cup \dots \cup S_{i,p_i},$$

then the data space S is partitioned into $N = p_{lat} \times p_{lon} \times p_{time} \times p_{theme}$ Emage.

Emage

An Emage g is a function $g : X \rightarrow V$ from some point set X to some value set V . A V valued Emage on X is an element of V^X . Here we used grid as the data structure representation of the Emage. So $g = \{(x, v(x)) | x \in X = \mathbb{N}^2\}$, and $v(x) \in V = \mathbb{R}\}$, and each $(x, v(x))$ is called a stel (spatio-temporal element). We use $|g|$ to indicate the size of an Emage, which is (number of rows, number of columns) in the Emage.

Spatial Dimension S_{space}

- **Point Geometry**

Point is the fundamental two-dimensional building block for spatial dimension. It consists of *latitude* and *longitude* value in range $[-90.0, 90.0]$ and $[-180.0, 180.0]$ respectively. For example, the location of traffic sensor can be modeled as

$$S_{space} = point(33.646458, -117.831130).$$

- **Line Geometry**

Line consists of two points that represent the start and end points of a line. For example, a road segment can be modeled as

$$S_{space} = line(33.647655, -117.835314 \quad 33.646056, -117.835196).$$

- **Grid Geometry**

Grid consists of two points that represent the southwest (bottom left) and northeast (upper right) corner of a grid. For example, a bounding box of Orange County in California is

$$S_{space} = grid(33.3869, -118.1174 \quad 33.9473, -117.4127)$$

- **Polygon Geometry**

Polygon consists of N points that represent the vertices of a closed polygon. For example, the active wildfire area is

$$S_{space} = polygon(37.830, -122.377 \quad 37.830, -122.377 \quad 37.830, -122.377 \quad 37.830, -122.377)$$

Temporal Dimension S_{time}

- **Time Instants**

Time instant is a particular time in T at the time granularity G . We model the time point at which an event occurs by an instant timestamp ts along with its finest time granularity g . $S_{time} = \langle ts, g \rangle$ For example, an hourly weather report at 2PM GMT on January 31, 2016 is modeled as $S_{time} = \langle 2016-01-31T12:00, \text{'hours'} \rangle$

- **Time Periods**

A period is a contiguous subset of the time domain in which the start time and end time are known. A period of granularity G , encoded with the start time st , end time et , and finest time granularity g . $S_{time} = \langle st, et, g \rangle$ For example, a meeting started from 2:30PM to 3:30PM GMT on May 23, 2016 is modeled as $S_{time} = \langle 2016-05-23T14:30, 2016-05-23T15:30, \text{'minutes'} \rangle$. We assume that both start time and end time have the same granularity

Thematic Dimension S_{theme} Thematic dimension defines what topics are being measured by data sources. A single data source may generate a data stream that contains one or more topics. For example, an air quality sensor may be used to measure Ozone concentration, temperature, and wind speed at the same time. A data type of the theme's value can be number (Theme Double), text (Theme String), or list of values (Theme List) as shown below.

- **Theme Double**

For example, a temperature in Irvine is 76 Fahrenheit which can be modeled as

$$S_{theme} = \langle \text{"temperature"}, \text{"value": 76} \rangle$$

- **Theme String**

For example, a tweet related to asthma can be modeled as

$$S_{theme} = \langle \text{"asthma"}, \text{"value": "I almost had an asthma attack."} \rangle$$

- **Theme List**

For example, a list of concepts on a Flickr photo can be modeled as

$$S_{theme} = \langle \text{"concepts"}, \text{"value": ["outdoor", "sky", "water", "beach"]} \rangle$$

Emage Stream

An Emage stream is represented as $ES = (t_0, g_0), \dots, (t_i, g_i), \dots$ where t_i is the time when

Emage g_i arrives. If g_i in ES consists of only one stel, ES becomes a special type of stel stream $SS = (t_0, p_0), \dots, (t_i, p_i), \dots$ where p_i is a stel.

Multi-resolution Emage

Multi-resolution Emage is a new concept introduced in this thesis. It allows one data stream to be processed at different levels of detail. An objective is to deal with data scalability issue when processing and visualizing large Emage. A drill-down approach has been widely used by scientists and data analysts to get insight on the large data sets. Most of the data can be processed and visualized at a relatively coarse resolution, and then the ‘interesting’ subset of the data can be identified and processed at the increasingly finer resolution. Representing data using multi-granularity model is a valuable tool for interactive exploration of very large data sets.

Figure 4.2 represents a quadtree structure for multi-resolution Emage. Each Emage Stream, ES , has a multiscale sequence M_L, M_{L-1}, \dots, M_0 of Emages, where M_L and M_0 correspond to the coarsest and finest resolution Emages, respectively. We assume that the finest scale Emage M_0 has a resolution of $\delta \times \delta$, and $i \times j$ stels. Each coarser resolution Emage M_z has a resolution of $(2^z \delta) \times (2^z \delta)$, and $(i \div 2^z) \times (j \div 2^z)$ stels. Each stel in Emage M_x corresponds to four “child” stels in Emage M_{x-1} . This indicates that quadtree is natural for the mapping. Each node on the tree is associated with one of the stels $M_z(a, b)$ corresponding to stel (x, y) in Emage M_z . The finer resolution Emage (lower level) is mapped to successively coarser resolution Emages (higher levels). The mapping strategies includes sum, max, min, avg, and majority.

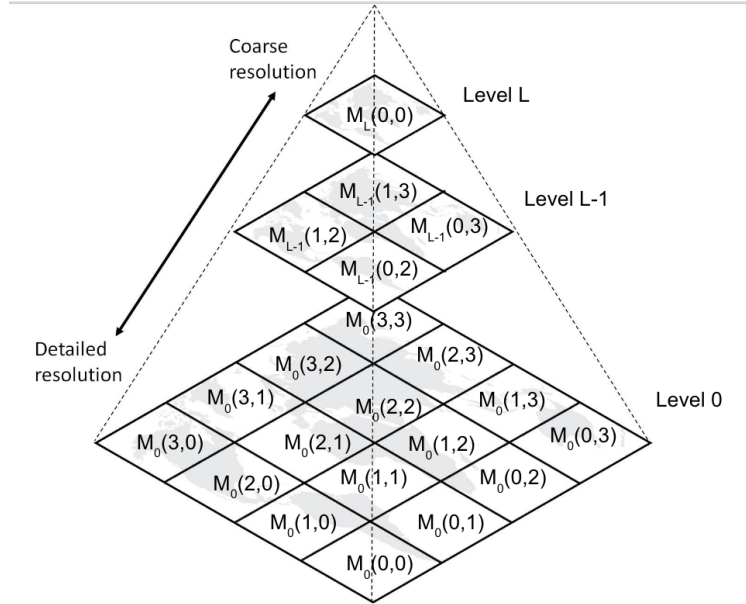


Figure 4.2: Quadtree structure for multi-resolution Emage

4.5 From Sensor Data to Situations

Data are influenced by characteristics or purposes of those people who create them. Therefore, they exist in heterogeneous data format, reflect knowledge from diverse domains, and exchange and propagate in different platforms or protocols. Data streams may be of different media types (e.g., free text, numeric value, categorical, or rich media types like images, audio, and videos). The format of observation streams can be different such as structured (records, XML, array), semi-structured, and unstructured. Since a situation can only be achieved by assimilating a set of data streams not only by a single data stream, the heterogeneity of data streams causes many challenges. The general procedure to derive high level situation semantic from low level sensors data is shown in Figure 4.3.

- (a) **Data Ingestion:** At this step, both physical and human sensor data streams are ingested into the EventShop. Users have to register the data sources first before start collecting data.

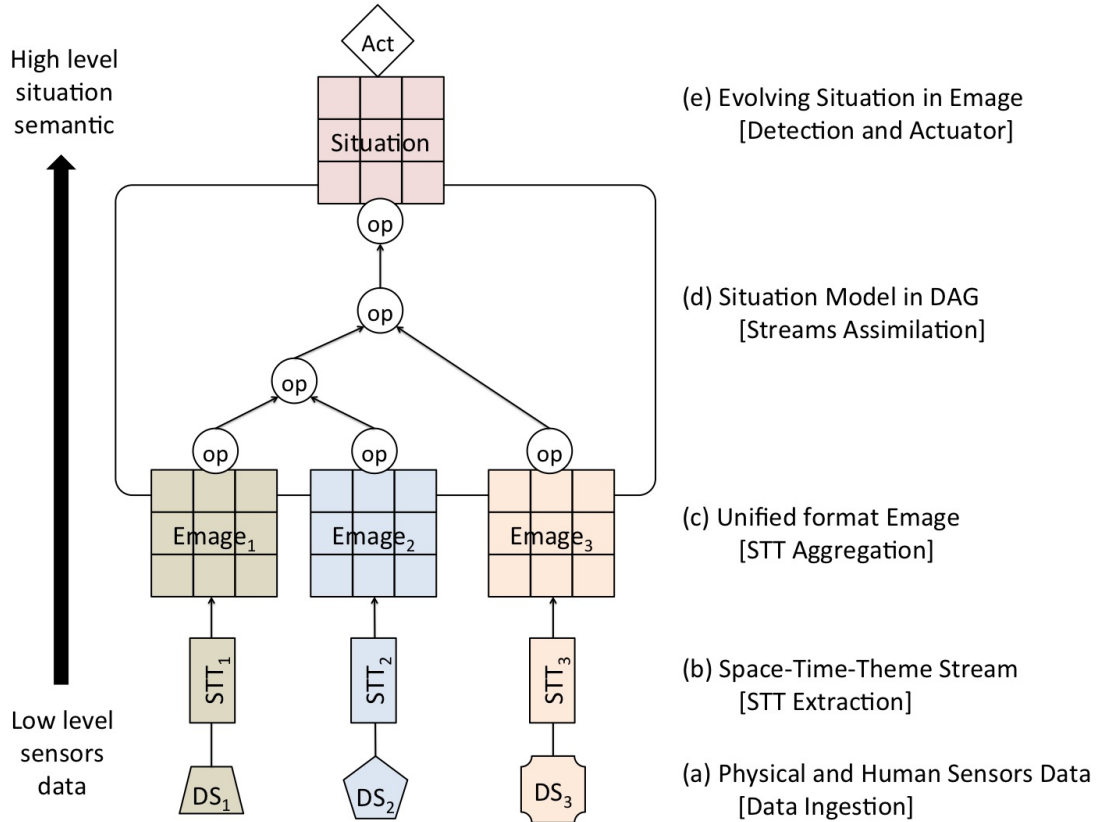


Figure 4.3: From low level sensors' data to high level situation semantic

- (b) **STT Extraction:** According to the data source configuration in the first step, space-time-theme information are extracted from the raw data streams. Users can define addition parameters in order to filter elements in each STT observation streams.
- (c) **STT Aggregation (Emage Generator):** Each STT Observation stream from the previous step is aggregated to generate an Emage Stream.
- (d) **Emage Stream Processing:** Users can create a situation recognition model by combining Emage operators (mentioned in the previous chapter). The Emage streams are continuously fed into the stream processing engine to create the final result of macro-situations in Emage.

- (e) **Situation Recognition and Taking Action:** Once the situation are recognized, appropriate information is sent to potential actuators according the user defined action control rules.

4.6 New EventShop System Architecture

In this section, we discuss a new EventShop system architecture which is more flexible, extensible, and scalable. We divided EventShop architecture into four layers, as shown in Figure 4.4, including user interface layer, application program interface (API) layer, processing layer, and data management layer.

In the first layer, EventShop interface is a web-based user interface that allows users to easily use EventShop system. To communicate between front-end and back-end, the REST API services are provides. Next is the processing layer. There are three main components: data ingestor, query processing engine, and alert/output unit. According to the data source registered by a user, data ingestor instances are created and run to collect raw data from outside either from physical data source or human generated data source. These raw are transformed into unified format STT (space-time-theme) data model presented above. The output from the data ingestor can be archived in the data store and passed to the query processing engine. In contrast to traditional one-time queries issued in database management system, a query in EventShop is a ‘standing query’. A standing query needs to be registered and instantiated in the system before relevant data streams flowing into the system and getting processed. The final outputs from the query processing engine are then sent to the alert or output unit to create personalized alerts and show in the dashboard for future decision making process.

Next section, we describe the user interface of EventShop. Then, in the next chapter, we discuss more details of the following three components; data ingestor, data store, and query processing engine. We combine data ingestor and data store together to create an archival system.

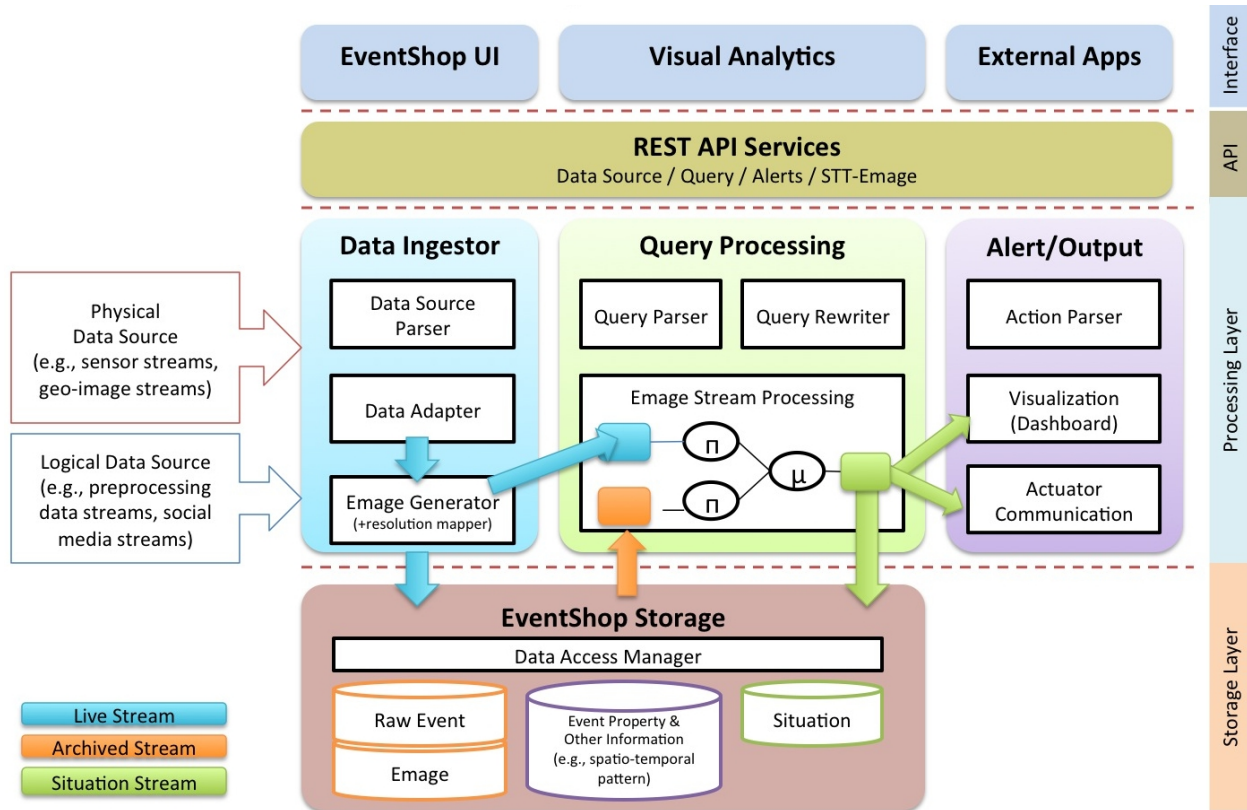


Figure 4.4: EventShop system architecture overview

4.7 Graphical User Interface

The EventShop back-end services and front-end UI is designed to be loosely coupled using client-server architecture. To communicate between back-end and front-end, we provide some basic REST API described in more detail in Appendix B. APIs allow the system to be evolved more easily in the future. The back-end is free to change the exposed resources at

will. One can modify the existing GUI or develop her own GUI without effecting the back-end processes. She only needs to know the initial URI and required parameters.

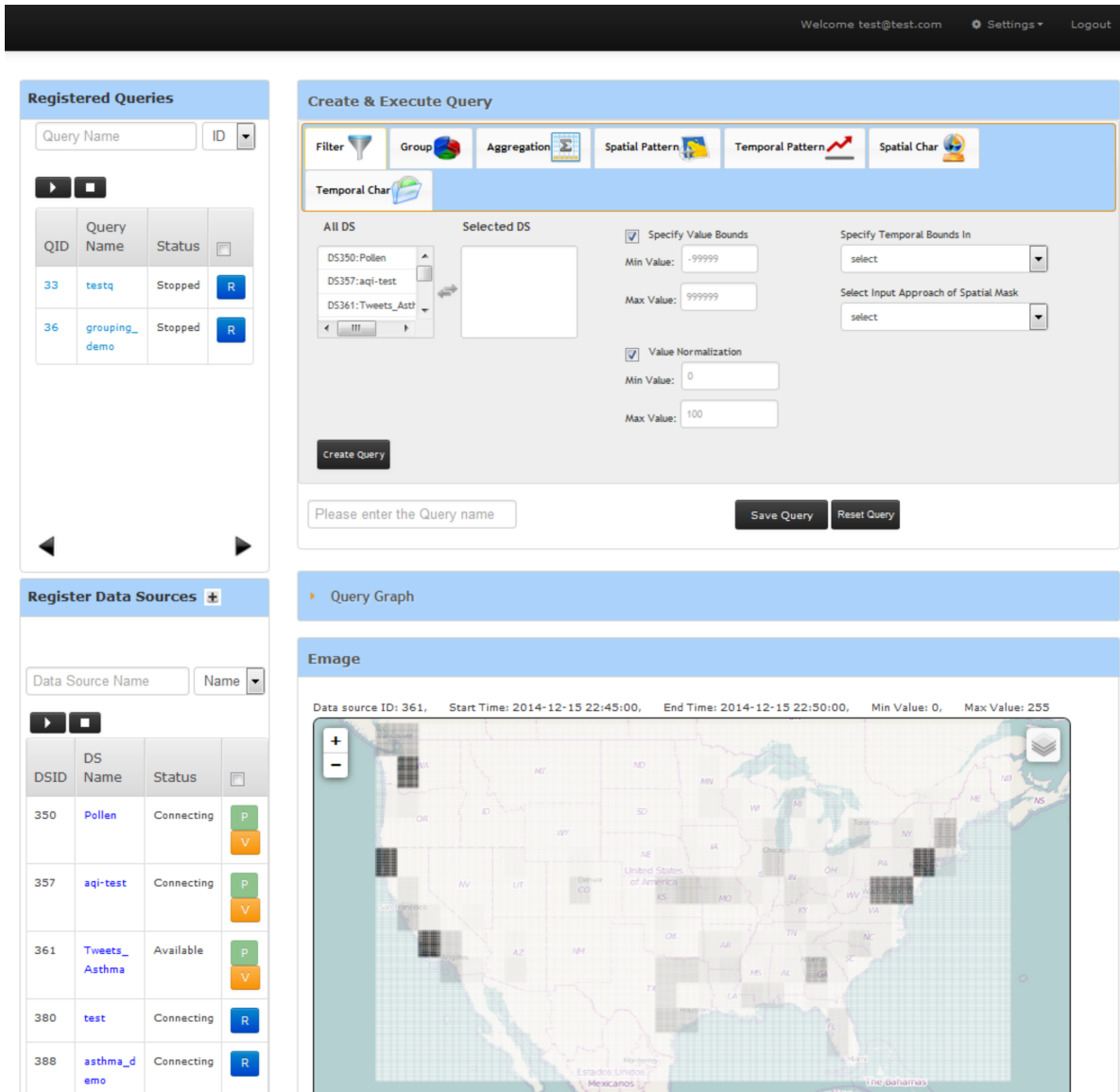


Figure 4.5: EventShop GUI

A layout of the EventShop main page is divided into a series of panels as show in Figure 4.5. The basic components are:

- (a) **Register Data Sources:** to display a current list of data sources registered with the system, to register new data sources into the system, and to control and visualize data sources
- (b) **Create and Execute Query:** to create and register query by applying different operators to the registered data sources
- (c) **Query Graph:** to graphically visualize the intermediate query currently being composed by the user
- (d) **Registered Queries:** to display a current list of queries registered with the system and to control the query processes
- (e) **E-mage:** to visualize an E-mage of a data source

Each components and its functionality are described in more detail in Appendix A.

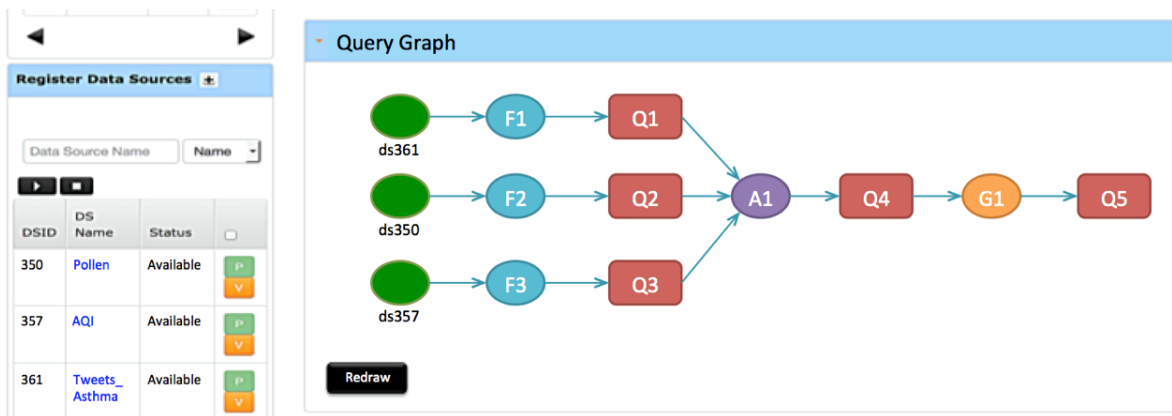


Figure 4.6: EventShop query graph

The query graph panel displays a query graph to aid users in forming query. Nodes in the tree represent data source, operator, or intermediate/finalized query. Its root node is the finalized query, leaf nodes are the data sources, and other nodes are either operators or intermediate queries. Each query node is represented in a rounded rectangle shape, while each data source and operator node are represented by an ellipse shape. A direction in the tree show how the data flow from end to end.

An example of query graph is displayed in Figure 4.6. Three data sources and three operators are used to create one finalized query. First, a filter operator is applied on each data source - “ds361”, “ds350”, and “ds357”. The intermediate queries - “Q1”, “Q2” and “Q3” are generated respectively. Second, we apply an aggregation operator on those intermediate queries and generate another intermediate query, called “Q4”. Finally, a grouping operator is applied on the query “Q4” to form a complex situation model.

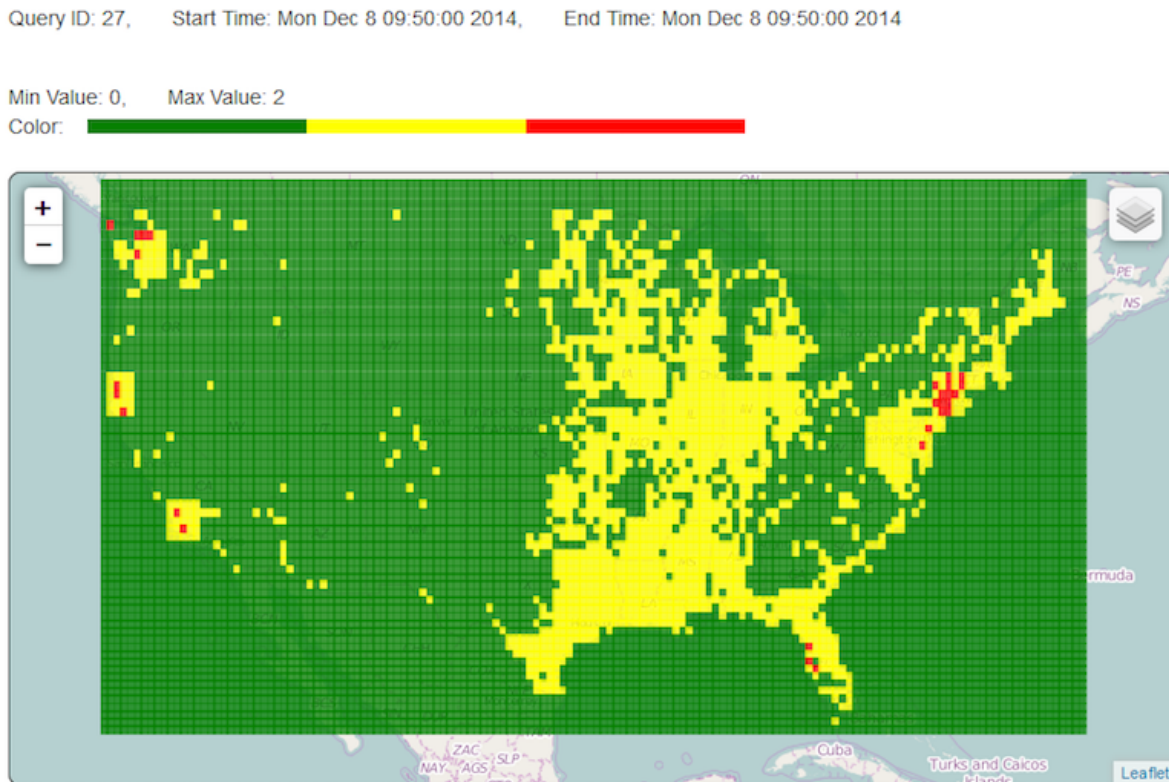


Figure 4.7: EventShop situation monitoring dashboard

A situation monitoring dashboard is shown in Figure 4.7. On the top of this page, information about the registered query are presented including *Query ID*, *Start Time*, *End Time*, *Min Value*, and *Max Value*. We present the query’s output E-mage on the map using Leaflet³ and D3⁴ JavaScript packages.

³<http://leafletjs.com/>

⁴<https://d3js.org/>

Chapter 5

Archival System

From the research challenges discussed in the previous chapter, we designed and implemented an archival system and query processing engine to detect situations from heterogeneous spatio-temporal data streams. In the archival system, these data streams are ingested, extracted, and transformed to a unified data format, Emage. Both raw data and Emages are stored in the database for use in later historical data analysis. Based on a situation model, the query processing engine continuously accesses Emage streams from the archival system and performs stream processing to generate an Emage showing real-time situations.

Figure 5.1 shows the overall architecture of the archival system and query processing engine. The archival system consists of two building blocks, a data stream ingestor and a data store. Their functionality is explained below. The query processing engine also consists of two components which are a situation model designer and an Emage stream executor. We explain the archival system in this chapter and the query processing engine in the next chapter.

The archival system in EventShop handles heterogeneity in disparate data sources and makes data available for historical data analysis by end users. We follow a similar principle of

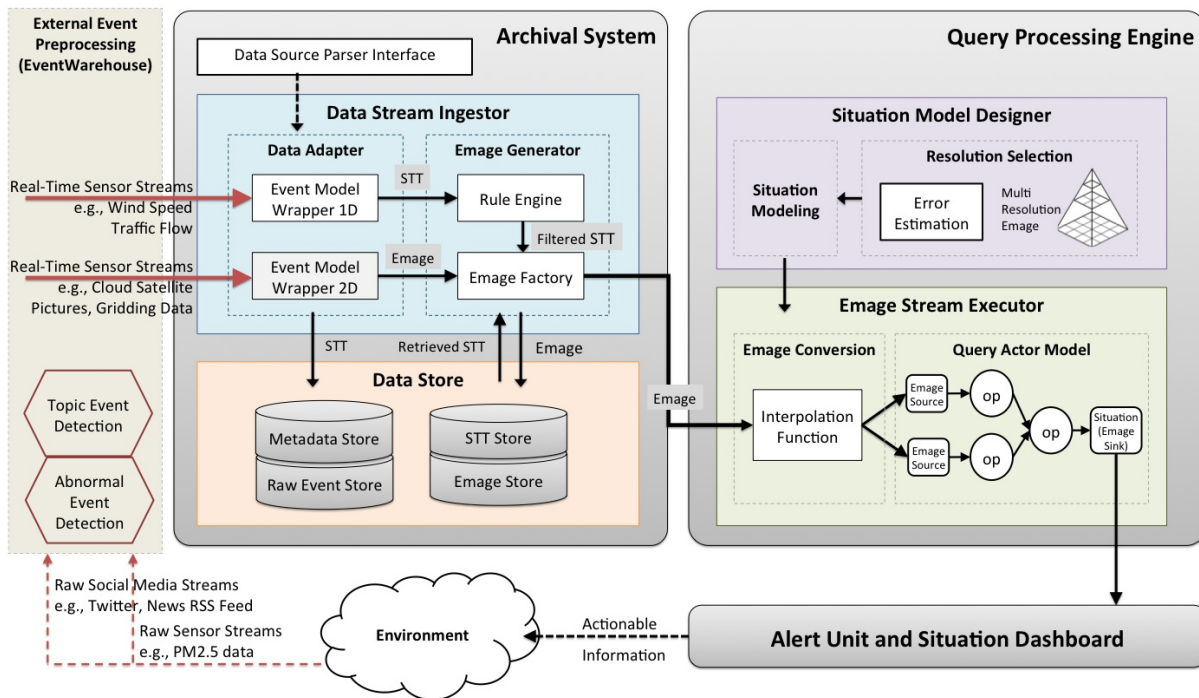


Figure 5.1: Archival system and query processing engine

data integration process in database management, called Extract, Transform, and Load (ETL). Data extraction extracts space, time, and theme data from the data sources, data transformation transforms the data into the STT model data format, and data loading loads the STT data into the data storage. To support data stream ingestion, EventShop archival system has two additional steps which are configuring data source, and generating Emage streams.

Several current technologies and tools are selected to create a reliable and scalable archival system such as Kafka [3], MySQL [14], MongoDB [13], and AsterixDB [5]. We explain the integration of these technologies along with their functionality at the end of this chapter. The EventShop archival system has two functional entities: a data stream ingestor and data store. The input of the archival system is the heterogeneous data streams and the output is the unified format data streams, Emage streams.

5.1 Data Stream Ingestor

The data stream ingestor gathers heterogeneous data from the cloud and transform them into a unified format. The processing pipeline in the data stream ingestor is divided into two steps: first, the extraction of space-time-theme values from raw data and second, the aggregation of extracted data to form an Emage. As shown in Figure 5.1, these two steps are handled by two main components: the data adapter and the Emage generator.

5.1.1 Data Adapter

Data adapter component pulls raw data from the external data sources via various out-of-the-box wrappers. These wrappers can connect data from Twitter and Flickr APIs, CSV files, JSON files, and XML/KML files into the unified STT format. Data from several sources such as PM2.5 measuring stations, satellite Aerosol Optical Depth observations, loop detectors on the freeway can be easily ingested into the system. Currently, these wrappers can connect to three different types of data sources including data stream, rest endpoint, and message queue types. Each data source require different parameters. We define a specification language to describe data adapter parameters. This specification is conducted on the system front-end interface and used internally in the system. JavaScript Object Notation (JSON) standard is chosen because it is lightweight and efficient in object serialization and message passing.

The data adapter specification language consists of two objects: *data_source* and *data_syntax* as shown in Code 5.1. The *data_source* object describes parameters or meta data of the data source such as url, data type, and data format. The *data_syntax* object identifies a pattern for extracting a space, time, and theme semantic from the raw data. We explain each object in further details below.

```
1 # The '#' character at the beginning of a line indicates a comment.
2 {
3     # describe data source metadata
4     "data_source": {
5
6     },
7     # describe data syntax to extract space, time, and theme values
8     "data_syntax" : {
9
10    }
11 }
```

Code 5.1: Data adapter specification language

Data Source Object

The parameters in the `data_source` object includes:

- **name:** a name of the data source identified by users. For instance, the name could be `pm25_concentration`, `tweets_allergy`, and `shelter_locations`.
- **url:** a unique resource location where a data stream can be retrieved. For example, Twitter provides a stream and search API to query tweet streams, AirNow has an rest endpoint linked to a specific air quality map, and Krumbs [11], a micro reporting platform, produces a message queue which allows users to consume micro reports.
- **data_type:** a type of the data source's access protocol. An appropriate data wrapper is required to connect to different types of data source including data stream, rest endpoint, and message queue protocol.
- **data_format:** the data format of the raw data streams. Data sources provide raw data in variety of formats and structures. For example, values for physical sensors could be presented in CSV (comma-separated values), JSON, XML (Extensible Markup Language), or KML (Keyhold Markup Language) format. Results from popular APIs like Twitter search API are usually available in both JSON and XML format.

- **keyword:** a list of keywords. It is required only when data stream wrapper is used. For example, to collect tweets associated with particular topic, users can define a bag of words as a query in Twitter search API.
- **time_window:** size of time window within every which data points of a data source are collected. The unit is second. For example, traffic sensors deployed by PeMS (Caltrans Performance Measurement Systems) generated new data every five minutes. The time window of this data source could be set to 300 seconds to collect all updated data.
- **bounding_box:** spatial bounding box of the area which data source covers. Bounding box consists of two points that represent the southwest (bottom left) and northeast (upper right) corner. For example, a bounding box of Orange County in California is `[[33.3869,-118.1174], [33.9473,-117.4127]]`

The following section presents example of data source object from three different data types.

(i) Data Stream Type: For this data type, the `data_type` parameter is “stream”. We currently support Twitter API, Flickr API, and data simulator, so users can set the `data_format` parameter to be one of these three values ‘Twitter’, ‘Flickr’, or ‘Simulation’. An example of Twitter data source object is shown in Code 5.2. In this data source, Twitter search API is used to find all tweets related to asthma and allergy based on the specified keywords (allergy, asthma, and pollution) across USA every 6 hours (21,600 seconds).

```

1 {
2   "data_source": {
3     "name": "twitter-allergy",
4     "url" : "https://api.twitter.com/1.1/search/tweets.json",
5     "data_type" : "stream",
6     "data_format" : "Twitter",
7     "keyword" : "allergy, asthma, pollution",
8     "time_window" : 21600,
9     "bounding_box" : "24.9493,-125.0011,49.5904,-66.9326"
10  }
11 }
```

Code 5.2: Twitter stream data source object

(ii) REST Endpoint Type: The REST endpoint is a connection point where files or active server pages are exposed. The data source must make the data available through a specific URL. The data type parameter is “rest”. The data format parameter can be selected from “csv_array”, “csv_fields”, “xml”, “kml” or “json”. The data in the csv_array format is represented in 2D matrix of values or image’s pixels such as data from satellite. On the other hand, the csv_fields format presents data in table-like structure where each row is one record with the same numbers of columns in each row. An example of this data source object is shelters’ location during flood event in Thailand as shown in Code 5.3. The data from this data source are available in the KML format. They are collected every hour (3600 seconds).

```
1 {
2   "data_source": {
3     "name": "flood-shelter",
4     "url" : "http://shelter.thaiflood.com/webservice/request.kml",
5     "data_type" : "rest",
6     "data_format" : "kml",
7     "keyword" : "",
8     "time_window" : 3600,
9     "bounding_box" : "5.6126,97.3440,20.4647,105.6369",
10  }
11 }
```

Code 5.3: Shelters’ location REST endpoint data source object

(iii) Message Queue Type: Message queue is widely used for communication between different systems or applications. Unlike technologies such as SQL, ODBC, and FTP which require both sender and receiver to be participating at the same time in order to move data from one system to another system, message queue provide asynchronous communication protocol. The message queue is similar to an office mail slots. Employees in the office have their own mailbox with their name on it. To get a document to someone, one has to put it in recipients’ slot. The recipients can pick up all their documents later. Message queues

enable asynchronous processing, which allows a data producer to put a message on the queue until a data consumer explicitly asks for the message. In this case, the data source is the data producer, and out data adapter is the data consumer. Example of message queue are Amazon Simple Query Service (SQS)¹, ActiveMQ², and RabbitMQ³.

An example of data source object with message queue type is a report about floods in Chennai, India from reporters using Krumbs [11]. Krumbs send each report to Amazon SQS in JSON format. Therefore, the data type parameter is “queue”, and data format parameter is “json” as shown in Code 5.4. The data adapter is set to retrieve reports in the queue every 5 minutes (300 seconds) interval.

```
1 {
2   "data_source": {
3     "name": "krumbs-flood",
4     "url" : "https://sqs.us-west-2.amazonaws.com/636414996916/io-krumbs-sdk-
      mediajson_flood",
5     "data_type" : "queue",
6     "data_format" : "json",
7     "keyword" : "",
8     "spatial_feature": "point",
9     "time_window" : 300,
10    "bounding_box" : "12.9672,80.1846,13.1513,80.3046"
11  }
12 }
```

Code 5.4: Krumbs micro reporting message queue data source object

Data Syntax

The second object in the data source specification is Data Syntax. The *data_syntax* object first describes how the ingested data has to be transformed (into STT data model) and stored in database. Second, it has to indicate the mapping between the raw data scheme and the STT data scheme. This object includes following parameters:

¹<https://aws.amazon.com/sqs/>

²<http://activemq.apache.org/>

³<http://www.rabbitmq.com/>

- **stt_scheme:** The `stt_scheme` declare the final STT scheme that will be stored in the database. The field name and field type are configured as a JSON. The field type could be NUMBER, STRING, TIMESTAMP, or DATETIME. The basic STT point observation is shown in Code 5.5 and 5.6.

```

1 {
2   "data_syntax" : {
3     # an example of STT point observation
4     "stt_scheme" : {
5       "space" : { "point" : { "lat" : "NUMBER", "lon" : "NUMBER" } }},
6       "time" : "TIMESTAMP",
7       "theme" : { "aqi" : { "value": "NUMBER" } }
8     },
9
10    # an example of mapping for CSV_fields data format
11    "stt_mapping" : {
12      "datasource_type" : "point",
13      "space.point.lat_index" : 2,
14      "space.point.lon_index" : 1,
15      "time_index": 0,
16      "time_format" : "yyyy-MM-dd'T'HH:mm:ss",
17      "theme.aqi.value_index": 3
18    }
19  }
20 }

```

Code 5.5: Data syntax specification language for CSV data format

```

1 {
2   "data_syntax": {
3     # an example of STT point observation with multiple theme value
4     "stt_scheme": {
5       "space": { "point": { "lat": "NUMBER", "lon": "NUMBER" } }},
6       "time": "TIMESTAMP",
7       "theme": {
8         "speed": { "value": "NUMBER" },
9         "deg": { "value": "NUMBER" }
10      }
11    },
12    # an example of stt_mapping for XML, KML, and JSON data format
13    "stt_mapping": {
14      "isList": true,
15      "rootElement": "",
16      "tokenizeElement": "values",
17      "space.point.lat_path": "coord.lat",
18      "space.point.lon_path": "coord.lon",
19      "time_path": "dt",
20      "time_format": "Long",
21      "theme.speed.value_path": "wind.speed",

```

```
22     "theme.deg.value_path": "wind.deg"
23     }
24 }
25 }
```

Code 5.6: Data syntax specification language for JSON data format

- **stt_mapping:** Different data formats requires different mapping scheme. Extracting STT values from the raw data is not a trivial task. Instead of manually creating an ad-hoc code for parsing data, we generalize the common characteristics of the raw data and come up with basic parameters. These parameters have to be declared by users. In this section, we provide example of the data input in two different data formats, CSV_fields and JSON, and explain how to extract STT values from them.

```

1 20160704T13:00,37.7,-121.8,79
2 20160704T13:00,37.8,-121.8,5
3 20160704T13:00,37.7,-122.2,78
4 20160704T13:00,37.8,-122.2,13
5 20160704T13:00,37.8,-122.3,109
6 20160704T13:00,37.7,-122.3,11
7 20160704T13:00,37.8,-122.3,8
8 20160704T13:00,39.8,-121.8,153
9 20160704T13:00,39.9,-121.8,16
10 20160704T13:00,38.2,-120.7,58
11 20160704T13:00,38.8,-120.7,7
12 20160704T13:00,39.0,-122.3,19
13 20160704T13:00,39.9,-122.3,16
14 20160704T13:00,37.9,-122.0,57
15 20160704T13:00,37.7,-122.0,10
16 20160704T13:00,37.8,-122.0,14
17 20160704T13:00,38.0,-122.4,76
18 20160704T13:00,38.8,-122.4,16
19 20160704T13:00,36.8,-119.8,171
20 20160704T13:00,36.6,-119.8,3
21 20160704T13:00,36.8,-119.8,19
22 20160704T13:00,36.8,-119.8,173
23 20160704T13:00,36.6,-119.8,12
24 20160704T13:00,36.8,-119.8,6
25 20160704T13:00,36.6,-120.4,83
26 20160704T13:00,36.6,-120.4,13
27 20160704T13:00,36.8,-120.4,4
28 20160704T13:00,32.7,-115.5,109
29 20160704T13:00,32.2,-115.5,4
30 20160704T13:00,32.4,-115.5,13
31 20160704T13:00,32.6,-115.5,9
32 20160704T13:00,32.8,-115.5,11
33 20160704T13:00,32.7,-115.5,123
34 20160704T13:00,32.2,-115.5,11
35 20160704T13:00,32.4,-115.5,7
36 20160704T13:00,32.6,-115.5,4
37 20160704T13:00,32.8,-115.5,4
38 20160704T13:00,32.7,-115.5,128
39 20160704T13:00,32.2,-115.5,16
40 20160704T13:00,32.4,-115.5,8
41 20160704T13:00,32.6,-115.5,11
42 20160704T13:00,32.8,-115.5,17
43 20160704T13:00,36.5,-117.9,16
44 20160704T13:00,36.6,-117.9,14
45 20160704T13:00,36.8,-117.9,11
46 20160704T13:00,35.1,-118.1,11
47 20160704T13:00,35.5,-118.1,4
48 20160704T13:00,35.6,-118.1,8
49 20160704T13:00,35.8,-118.1,12
50 20160704T13:00,35.4,-119.0,166

```

Code 5.7: Raw Data in CSV Field

```

1 {
2   "count": 10,
3   "values": [
4     {
5       "id": 495260,
6       "name": "Shcherbinka",
7       "coord": {
8         "lon": 37.559719,
9         "lat": 55.499722
10      },
11      "dt": 1435731186,
12      "wind": {
13        "speed": 2,
14        "deg": 360,
15        "var_beg": 290,
16        "var_end": 90
17      }
18    },
19    {
20      "id": 564517,
21      "name": "Dubrovitsy",
22      "coord": {
23        "lon": 37.486698,
24        "lat": 55.43969
25      },
26      "dt": 1435731486,
27      "wind": {
28        "speed": 2,
29        "deg": 360,
30        "var_beg": 290,
31        "var_end": 90
32      }
33    },
34    {
35      "id": 570578,
36      "name": "Butovo",
37      "coord": {
38        "lon": 37.57972,
39        "lat": 55.548328
40      },
41      "dt": 1435731486,
42      "wind": {
43        "speed": 2,
44        "deg": 360,
45        "var_beg": 290,
46        "var_end": 90
47      }
48    }, ...
49  ]
50 }

```

Code 5.8: Raw Data in JSON

(i) CSV Field Data Format

- **[field’s name in STT]__index**: The column index (starting from 0) in the raw data is used to assign each value into each field in the STT model. In the sample of raw data in CSV fields (Code 5.7), there are four columns starting from date time, longitude, latitude, and value. The specification for this raw data set is shown in Code 5.5.
- **[field’s name in STT]__format**: To convert DATETIME data type in the raw data into TIMESTAMP data type in STT, the format of the original DATETIME is required. It can specify in JSON using `*_format SYNTAX`. For example, the date time value in the raw data is “2016-07-04T13:08”. Then, in the `stt_mapping`, we can set `"time_format": "yyyy-MM-dd' T' HH:mm"`.

(ii) JSON, XML, and KML Data Format

These data formats share the same parameters. An example of `data_mapping` for the raw data in JSON (Code 5.8) is shown in Code 5.6.

- **isList**: is a Boolean (true or false) data type. If the input data is a list of several data points, the value of this parameter will be “true”. From the sample data in Code 5.8, this data set provides an array of data points, so `"isList": true`.
- **rootElement**: is used to specify the root element’s name in JSON, XML, and KML data to parse through. The root element is the highest level object in the JSON data. In the example above, the name of the root element does not exist.
- **tokenizeElement**: indicates a name of a list element that contains all data point elements. In the sample data above, the name of the list is “values”, so we can set `"tokenizeElement": "values"`. The content inside that list is tokenized into several elements. Each element represents one data observation point.
- **[field’s name in STT]__path**: is similar to column index mentioned in CSV fields data format above. We assign each value in the raw data to each STT

field based on its path name. For nested object in JSON, sub-elements can be accessed using dot (.) notation. In the sample JSON data (Code 5.8), a “coord” is a JSON element with two sub-elements: “lon” and “lat”. To access the value of “lat” element, this following syntax “coord.lat” can be used. Therefore, we can set “space.point.lat_path” : “coord.lat” as shown in Code 5.5. Note that the dot notation is also used when nested object in *stt_scheme* is identified. For instance, to access wind speed value from the sample data, a “wind.speed” path can be used. Assigning that value to the “theme” field in STT can be specified as `"theme.speed.value_path":"wind.speed"`.

For XML and KML data format, the path to an attribute in an element should begin with ‘@’ character. If the element has no child element, the path to a its text is the same as the node or tag name. If the element has one or more child elements, the path to a child element is parent-node-name/child-node-name. For example, the raw data in XML is

```
<city id="495260" name="Shcherbinka">
  <coord>
    <lat> 55.499722 </lat>
    <lon> 37.559719 </lon>
  </coord>
  <temperature avg="34" min="27" max="36" unit="celsius" />
</city>
```

The path to access longitude value is “city/coord/lon”, and the path to access average temperature of this city is “city/temperature/@avg”.

- **[field’s name in STT]__format:** is the same as the parameter mentioned in CSV data format.

STT Data from Data Adapter

The output of data adapter is data stream in STT data model. An example of a STT point, transformed from the sample data in CSV using the `data_syntax` above is shown in Code 5.9.

```
1 {
2   "stt_id" : "80003c2a-02bf-4a05-bbfe-1fab0dcd3c65",
3   "space" : {
4     "point" : { "lon" : 37.7, "lat" : -121.8 }
5   },
6   "time" : "1467662400000",
7   "theme" : {
8     "aqi": { "value" : 79 }
9   },
10  "raw_data" : "20160704T13:00,37.7,-121.8,79"
11 }
```

Code 5.9: Sample STT point

5.1.2 Emage Generator

An Emage generator represents a step in accessing a STT data stream from the specified data source, and aggregating or converting them into an Emage stream. One STT stream can be used to generate multiple Emage streams based on the filter conditions, Emage resolutions, and aggregation functions. The rule based engine is used to form filter conditions. The STT points that satisfied all the conditions are sent to an Emage factory to generate Emage streams corresponding to user defined parameters. For example, a STT stream of PM2.5 concentration across USA can be used to generate two Emage streams; one is PM2.5 concentration in the east coast of USA and another one is PM2.5 concentration in the west coast of USA.

Rule Engine

A rule engine in Emage generator filters the STT data streams and extracts only STT points that satisfy the user defined rules. Users can filter STT on any space, time, and theme dimension. For example, user can retrieve STT points with an air quality index value higher than 50 and an geo-location inside USA. A sample rule engine specification is shown in Code 5.10. The rule parameters includes:

```
1 {
2   "source" : "ds3",
3   "extractFields" : "space,time,theme",
4   "rules" : [
5     {
6       "dataField" : "theme.aqi",
7       "ruleOperator" : ">",
8       "ruleParameters" : "50" },
9     {
10      "dataField" : "space.point",
11      "ruleOperator" : "coordinates",
12      "ruleParameters" : "24.0,-125.0,50.0,-66.0"
13    }
14  ]
15 }
```

Code 5.10: Rule engine specification

- **Source ID:** The data source ID of the STT stream being accessed. Note: when the data source was registered (in data adapter step), the data stream ingestor automatically generates a unique ID for each data source.
- **Extract Field:** The fields required as an output should be specified in the Extract Field. Each field is separated by comma.
- **Rules:** A list of rules is similar to a WHERE clause in the SQL language. It is used to filter only those STT points in the STT stream that fulfill specified criteria. Each rule has three parameters: data field name, operator, and value parameters. The operators can be selected depend on the data type specified in the data_syntax object when

configuring the data source. The operators are supported for the corresponding data type as shown in Table 5.1.

Data Type	Operator	Value Type
Number	>, <, =, !=	Double number
String	equal	String
String	regex	String regex
Location	radius	Double longitude, Double Latitude, Double radius
Location	address	String
Location	coordinates	Bottom longitude, Bottom Latitude, Top longitude, Top latitude (all values are double number)
Timestamp	before, after	Long number (in millisecond)
Timestamp	between	Start time, End time (long number in millisecond)

Table 5.1: Rule operators and value types

Emage Factory

To allow a creation of Emage, STT points from STT streams have to be aggregated in certain spatio-temporal boundary and resolution. A set of variable frame parameters as shown in Code 5.11 includes:

```

1 {
2   "frame_parameters" : {
3     "time_window" : "3600",
4     "lat_unit" : 0.1,
5     "lon_unit" : 0.1,
6     "bounding_box" : "24.0,-125.0,50.0,-66.0",
7     "aggregation" : "avg"
8   }
9 }
```

Code 5.11: Emage factory specification

- **“time_window”** is the size of time window within every which STT points are aggregated to create an Emage. The time window used is tumbling window where two continuous windows are non-overlapped with each other and the second window starts immediately after the first window ends.

- “**bounding_box**” is the spatial bounding box of Eimage represented in southwest (SW) and northeast (NE) geo-coordinate points. The format is a string of four numbers separated with comma in the following order: SW_LATITUDE, SW_LONGITUDE, NE_LATITUDE, and NE_LONGITUDE”.
- “**lat_unit**” and “**lon_unit**” are the spatial granularity of Eimage. STT points from a geo-spatial area contribute to the value of the corresponding cell in Eimage.
- “**aggregation**” indicates how to aggregate STT points in each cell of Eimage. The choice of aggregation depends on the context of the data source and the STT type (point, grid, polygon). For example, STT points from Twitter could be aggregated using *count* function. STT points about traffic flow on a road section could be aggregated using *average* function.

All out-of-the-box functions include aggregation functions (i.e., *sum*, *count*, *average*, *min*, and *max*), interpolation functions (i.e., *linear*, *nearest neighbor*, and *cubic*), and rasterization functions (i.e., *center within polygon*, and *majority*).

5.2 Data Store

Data store is the second building block of the data stream ingestor. This component receives an output from the data adapter and stores it in a database. Currently, we support two database systems: MongoDB and AsterixDB [28]. AsterixDB is an open source big data management system that provides scalability like other Big Data platforms, with a full functionality of data management. Because of its flexible data model, spatial-temporal query support, and built-in data feed ingestion, AsterixDB well serves our requirements.

5.2.1 AsterixDB

In this section, we explain the Asterix’s data model (ADM), query language (AQL), and some features that we use for storing STT data [5]. In the ADM, the top level of data organizing concept is *dataverse* (from “data universe”). It is similar to a *database* concept in a relational DBMS. To store data in AsterixDB, we need to create a dataverse and use it to create and manage the *datatypes* and *datasets*. The *datatype* is a type of data instances. It is prescribed in the datatype definition. A *dataset* is a collection of data instances of a datatype. Before storing any data instances into the dataset, AsterixDB checks the datatype of each data instance with the datatype of the dataset. AsterixDB makes sure that all data instances in the dataset have the same and correct datatype. To support semistructured data, a datatype can be defined as an *open* datatype. Instances of open datatypes are allowed to have additional content, as long as they at least contain all required fields prescribed in the datatype definition.

To use AsterixDB, we first create a dataverse called “EventShop”. This dataverse is the place in which to create and manage all datatypes and datasets for EventShop. ADM can support a primitive type such as uuid (universal unique identifier), string, int16(short), int32(int), int64(long), double, date, time, datetime, point, rectangle, and polygon. We, however, need to create a new datatype for STT data instances. Each STT instance contains a space, time, and theme value. A space and time datatype can be a primitive type, but a theme datatype is a complex type. Therefore, we have to create a new datatype for a theme as well. Code 5.12 below shows an example of ADM syntax for creating dataverse and datatype used in EventShop. The ADM syntax is an extended JSON structure with more data types and additional data modeling constructs borrowed from object databases. A question mark symbol (?) after datatype indicates an optional field. For example, the “ThemeDouble” datatype has one mandatory field (‘value’), and two optional fields (‘uncertainty’ and ‘unit’). Since this datatype is not an *open* datatype, it does not allow additional contents.

Once a data source is registered, the data store process the following steps:

1. Parse “data_syntax” and find “stt_scheme” of a registered data source.
2. Create a new theme datatype, and name it with this pattern “STT[data_source_name]ThemeType”.
The space and time datatype are a primitive type so new datatype is not needed. For example, the “stt_scheme” in Code 5.6 above shows that the space datatype is a POINT and the time datatype is a TIMESTAMP (or int64). The theme datatype is created as shown in Code 5.12 below.
3. Create a new STT datatype for this data source. Besides space, time, and theme fields, STT datatype has a mandatory field to store a unique ID and an optional field to store raw data of each STT instance. An example is shown in the bottom of Code 5.12. The name of this datatype is “STT[data_source_name]Type”.
4. Create a new dataset for storing STT instances with the STT datatype in the previous step. The primary key is the unique ID which is auto generated by AsterixDB. The name of this dataset is “STT[data_source_name]”. The DDL for creating a dataset is shown in Code 5.12

```
1 # Create EventShop dataverse
2 # -----
3 drop dataverse EventShop if exists;
4 create dataverse EventShop;
5
6 use dataverse EventShop;
7
8 # Create ThemeDouble datatype
9 # -----
10 drop type ThemeDouble if exists
11 create type ThemeDouble{
12   value: double,
13   uncertainty: double?,
14   unit: string?
15 }
16
17 # Create ThemeString datatype
18 # -----
```

```

19 drop type ThemeString if exists
20 create type ThemeString{
21   value: string,
22   uncertainty: double?,
23   unit: string?
24 }
25
26 # Create a new theme datatype of a new data source
27 # by combining ThemeDouble and ThemeString.
28 # Name format: STT[data_source_name]ThemeType
29 # i.e., the theme type of "sample" data source is "STTSampleThemeType"
30 # -----
31 create type STTSampleThemeType as open{
32   speed: ThemeDouble,
33   deg: ThemeDouble
34 }
35
36 # Create a new STT datatype of a new data source
37 # Name format: STT[data_source_name]Type
38 # i.e., the STT type of "sample" data source is "STTSampleType"
39 # -----
40 create type STTSampleType as close{
41   stt_id: uuid,
42   stt_space: point,
43   stt_time: int64,
44   stt_theme: STTSampleThemeType
45   raw_data: string?
46 }
47
48 # Create a new STT dataset
49 # Name format: STT[data_source_name]
50 # i.e., a dataset of "sample" data source is "STTSample"
51 # -----
52 use dataverse EventShop;
53
54 create dataset STTSample(STTSampleType)
55 primary key stt_id autogenerated
56
57 # Insert new STT point into the STTSample dataset
58 # -----

```

Code 5.12: Creating dataverse, datatypes and dataset for EventShop in AsterixDB

Once STT outputs from the data stream ingestor arrive at the data store, they are inserted into the respective dataset. Code 5.13 shows an insert statement of one STT instance. At run time, a set of STT instances can be inserted at once. Example query statements are presented below. A complete explanation about Asterix Query Language (AQL) can be found at [18].

```

1 use dataverse EventShop;
2
3 insert into dataset STTSample(
4   {
5     "stt_space": point("139.7646713735534, 35.67087548164055"),
6     "stt_time": datetime("2015-10-09T07:18:01.000Z"),
7     "stt_theme": {
8       "theme1": {"value": 10},
9       "theme2": {"value": "this is a value"}
10    }
11  }
12 )
13
14 # Query all STT points in the STTSample dataset
15 # -----
16 use dataverse EventShop;
17
18 for $stt in dataset STTSample
19 return $stt;
20
21 # Query STT points and aggregate them into an Eimage
22 # -----
23 use dataverse EventShop
24 let $region := rectangle(33.3869,-118.1174 33.9473,-117.4127)
25 let $swpoint := point(33.3869,-118.1174)
26 let $start := 1420102800000
27 let $end := 1420189200000
28 for $t in dataset STTSample
29   where spatial-intersect($t.stt_space, $region)
30   and $t.stt_time >= $start and $t.stt_time <= $end
31   group by $c := spatial-cell($t.stt_space, swpoint, 0.1, 0.1) with $t
32 let $y := for $x in $t return $x
33 return { \"cell\" : $c, \"count\" : count($y), \"values\" : $y}

```

Code 5.13: Inserting STT data instance and querying STT data instances

We do not describe how to use MongoDB in this data store here since the steps mentioned above are similar; except that MongoDB is a schemaless database and does not have a datatype concept. Dataverses, datasets, and data instances in AsterixDB are comparatively equivalent to databases, collections, and documents in MongoDB respectively.

5.3 Archival System Infrastructure

The archival system treats data input from external data sources as a message. We use Apache Kafka [3] as an underneath infrastructure for passing messages in the system. The main reasons are (a) supporting publish-subscribe messaging model, (b) high throughput, (c) scalability with distributed infrastructure, (d) reliability through replication, and (e) ability to re-generate STT stream facilitated by data retention policy. Kafka maintains feeds of messages in categories called topics which are akin to STT datasets of AsterixDB. Kafka's messages are STT data instances that will be inserted into a specific AsterixDB's dataset according to the Kafka's topic. A data flow from heterogeneous data stream to Emage streams inside EventShop archival system is shown in Figure 5.2.

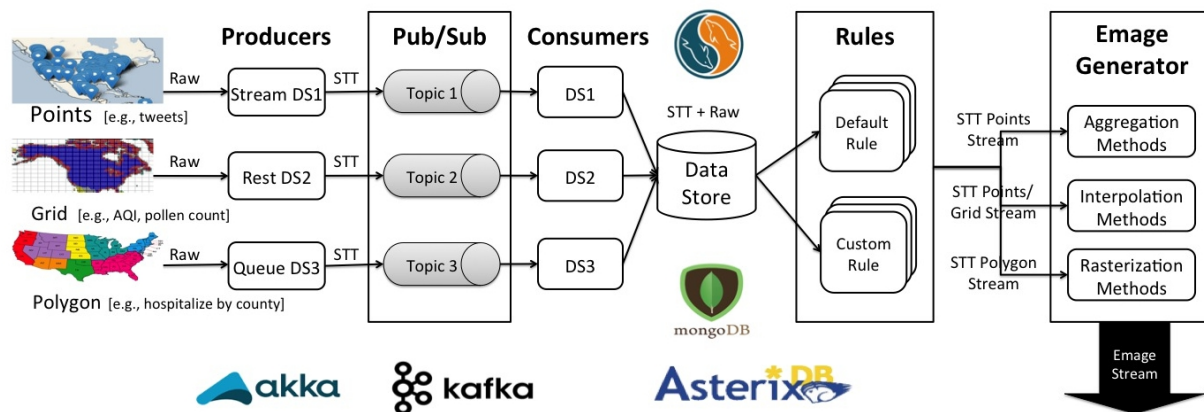


Figure 5.2: Archival system infrastructure

We use an actor model in describing message producers and message consumers. The actor model adopts the philosophy that “everything is an actor”. Actors are computational entities that can act upon receiving messages. They can concurrently send messages to other actors, spawn new actors, and designate the behavior to be used when it receive the next messages. To create such actor model, Akka [2] libraries and interfaces are used. The actor model uses asynchronous message passing for communication between actors.

Chapter 6

Multi-Resolution Processing Engine

As shown in the previous chapter's Figure 5.1, the query processing engine has two stages; building a situation model and executing it at runtime. A situation model is designed by a domain expert or situation-aware application developer who can define how to assimilate a number of data sources using a set of operators in order to detect a situation (e.g., city flood situation).

To deal with data streams at different granularities, we enhance the original Eimage data structure so that it can represent Eimage at multiple resolutions. In this chapter, we start with the definition of spatial, temporal, and Eimage granularity. Then, we present a process in building situation models that includes how to select the 'right' Eimage resolution for each application. We provide statistical information about the estimated error and response time for each situation model at different resolutions. This information can help users during the situation modeling process. After the model is built, the Eimage stream executor parses it and generates a physical query plan to be executed at runtime. Finally, we present Eimage clustering operator.

6.1 Definition of Granularity

In general, a *granularity* is a mapping from the set of integers or coordinates, called *index set*, to the subsets of a domain of interest. While the original domain can be discrete, dense, sparse, or continuous, the granularity is a countable set of disjoint *granules*. Spatial and temporal granularity has been broadly used in Geographical Information Systems (GIS), databases, and data mining and reasoning. In this section, we provide a history of the granularity definition, and the data model for processing and manipulating spatial, temporal, and Emage granularities.

6.1.1 Spatial Granularity

Intuitively, a *spatial granularity* may represent any partition of a space domain (e.g., \mathbb{R}^2) in disjoint regions, called *granules*. Typically, spatial data can be represented in two ways using a *raster* or *vector* data model. For the vector model, Belussi *et al.* [39] defined spatial granularities using a two-level model. The lower level contains geometrical information about the spatial domain, and the higher level is an index structure used to access and manage granules. To represent the relationship between granules, multidigraphs are used as index sets as shown in Figure 6.1(a). This graph is a directed graph with multiple labelled edges. Each node represents a spatial granule, and is mapped to its vector representation in the lower panel. Each edge represents a relationship between granules (e.g., direction-based and distance-based relationship), and is labelled with the name of the relationship.

In the raster model, the space dimension is partitioned into equal-sized square areas or cells, and represented as a raster map. The size of areas depends on the resolution of the map. Areas in the map are defined by a Cartesian coordinate system in which each area corresponds to a unique pair of integers. In [112], the spatial granularities of raster maps

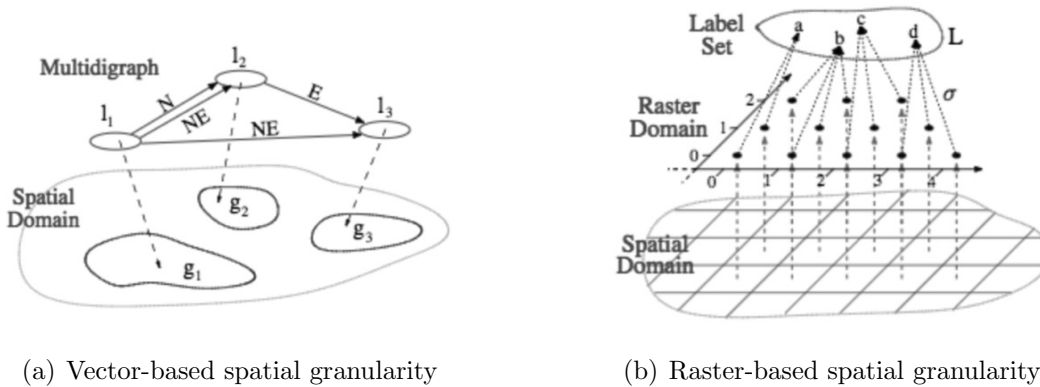


Figure 6.1: Spatial granularity in two data models [112]

represent the partitioning of cells and areas in the space, according to their associated values (or labels) as shown in Figure 6.1(b). Thus, the spatial granularity σ is defined as a total function from the two-dimensional coordinates in \mathbb{Z}^2 to the label set L , $\sigma : \mathbb{Z}^2 \rightarrow L$.

The relations between spatial granularities allow one to compute granules that belong to different granularities. These relationships are defined as:

- **GroupsInto(G,H):** each granule in H is equal to the union of granules in G ;
- **FinerThan(G,H):** each granule in G is contained in a granule of H ;
- **Subgranularity(G,H):** for each granule in G , there exists a granule in H with the same spatial extent;
- **Partition(G,H):** G groups into and is finer than H ;
- **CoveredBy(G,H):** image of G is contained in the image of H ;
- **Disjoint(G,H):** images of G and H are disjoint;
- **Overlap(G,H):** images of G and H overlap.

6.1.2 Temporal Granularity

A *temporal granularity* can be defined as a sequence of time granules, each one containing a set of time instants. This concept is a well-known notion used to temporally partition or aggregate data. Bettini *et al.* [43] defined temporal granularity and granules as follows:

A *granularity* is a mapping G from integers to subsets of the time domain, $G : \mathbb{N} \rightarrow 2^{\mathbb{N}}$, such that:

1. if $i < j$, for any $n \in G(i)$ and $m \in G(j)$, then $n < m$;
2. if $i < k < j$ and $G(i)$ and $G(j)$ are non-empty, then $G(k)$ is non-empty.

The domain of a granularity G is called an *index set*, and an element of G is called a *granule*. The first condition states that granules in a granularity do not overlap and their index order is the same as their original time domain order. Second, the subset of the index set that maps to non-empty subsets of the time domain is contiguous.

From the existing literature, the proposed framework for representing and reasoning about time granularity can be classified into *algebraic* and *logical* approaches. In the algebraic (or operational) framework, a bottom granularity is assumed, and new granularities are created from existing granularities using a finite set of calendar operators. In [92], Leban, McDonald, and Forster propose the *temporal interval collection formalism*. A collection of intervals is a structured set of intervals, and is essentially denoted by a set notation (except for the understanding that the order of elements is maintained). The order of the collection is a measure of the structure depth. An order 1 collection is an ordered list of intervals, and the order n collection, with $n > 1$, is an ordered list of collections of order $n - 1$. For example, the collection of months where each month is represented by a collection of the days in that month is an order 2 collection. The *dicing* operator is used to divide each interval of a

collection into another collection, while the *slicing* operator is used to select intervals from those collections. However, this model can represent only a set of finite no-gap granularity.

Bettini and Desibi [42] significantly extend the collection formalism to capture gap, quasi-periodical, and bi-periodic granularities. The gap granularities have granules which are composed of a set of noncontiguous time points in the basic time line (e.g., *BusinessMonths* is the set of business days in a month which is a gap granularity on the timeline of days). The quasi-periodical granularities are irregular periodical granularities. The bi-periodic granularities are represented by two sets of repeating granules.

Finally, Ning, Wang, and Jajodia [102] design the symbolic representation called the *calendar algebra*. This algebra consists of two kinds of operations: *grouping-oriented* and *granule-oriented* operations. The former combines granules of a granularity to form the new granules of a new granularity. The latter operation does not change the granules of a granularity, but rather selects some granules to remain in the new granularity. Each granule of a granularity that is generated by a calendar algebraic operation consists of one or more granules of at least one operand granularity, for example, $\text{minute} = \text{Group}_{60}(\text{second})$ and $\text{Sunday} = \text{Select} - \text{down}_{7}^1(\text{day}, \text{week})$.

6.1.3 Emage Granularity

We define Emage granularity by advancing a raster model approach in spatial granularity and an algebraic approach in temporal granularity. The *relations* and *operators* for the Emage data model are discussed below. The granularity of Emage depends on its frame parameters which include spatial resolution (latitude and longitude unit), spatial bounding box (southwest and northeast coordinate), and temporal resolution (size of time window). In this chapter, we interchangeably use the following terms: Emage granularity, Emage resolution, and resolution.

The relations between Emage granularities allow one to process Emages of different granularity. The relations are defined as:

- **FinerThan (G,H):** Emage resolution G is finer than Emage resolution H if $G.latitude_unit \leq H.latitude_unit$ and $G.longitude_unit \leq H.longitude_unit$ and $G.bounding_box$ contains $H.bounding_box$
- **CoarserThan (G,H):** Emage resolution G is coarser than Emage resolution H if Emage resolution H is finer than Emage resolution G
- **Combinable (G,H):** Emage resolution G is comparable with Emage resolution H if $G.latitude_unit = H.latitude_unit$ and $G.longitude_unit = H.longitude_unit$ and $G.bounding_box$ overlaps $H.bounding_box$ and $G.timewindow = H.timewindow$
- **MoreFrequentThan (G,H):** Emage resolution G is more frequent than Emage resolution H if $G.timewindow \leq H.timewindow$
- **LessFrequentThan (G,H):** Emage resolution G is less frequent than Emage resolution H , if $G.timewindow \geq H.timewindow$

The operators below are used to create a new Emage granularity from the existing one.

- **Coarse2Fine:** If $CoarserThan(G,H)$ exists, we can convert an Emage at the resolution G to an Emage at the resolution H using the following methods: nearest-neighbor interpolation, linear interpolation, bilinear interpolation, and split uniform.
- **Fine2Coarse:** If $FinerThan(G,H)$ exists, we can convert an Emage at the resolution G to an Emage at the resolution H using the following strategies: sum, max, min, avg, and majority.

- **Low2High:** IF `LessFrequentThan(G,H)` exists, an Emage stream at the resolution G can be converted to an Emage steam at the resolution H using the following methods: repeat or split uniform.
- **High2Low:** IF `MoreFrequentThan(G,H)` exists, an Emage stream at the resolution G can be converted to an Emage steam at the resolution H by using an aggregation function such as sum, max, min, avg, and majority, or by selecting every n^{th} Emage, first arrival Emage, or last arrival Emage.

6.2 Query Processing Engine

The query processing engine is divided into two phases: situation model designer and Emage stream processing at runtime. In the first phase, `EventShop` provides a front-end user interface to help users in design and build a situation model. We explain this process in Appendix A. In addition to building a model, users have to define the frame parameters of the final Emage. Selecting the right resolution for detecting a situation is not trivial. `EventShop` offers an error estimation function to aid users in performing this task. The second phase is to execute a situation model at runtime. This model acts like a standing query in the stream processing system. We discuss this component later in this section.

6.2.1 Resolution Selection

The quality of the situation recognition model in `EventShop` describes how well the actual situations are detected or predicted. The choice of spatial resolution of a given data stream determines a location accuracy and precision. An observation location can be at a very high degree of detail (precise) but may not be in the right place (accuracy). For example, a GPS sensor in a mobile phone measures precise geo-coordinates but it might be too far

away from the actual location by 10-15 meters. On the other hand, a weather forecast of a given city can be locally accurate but it is not very precise. The main challenge is to find the right grid resolution for each data input stream. Principles from general statistics and information theory such as the Nyquist frequency concept from signal processing and equations for estimating the probability density function can be closely related to the process of selecting the grid resolution.

We categorize data sources into categories and apply the equation in Figure 6.2 to find the three standard grid resolutions that can be derived for each data source: coarsest legible grid, finest legible grid, and recommended grid resolution.

Summary equations to select grid resolution: SN is scale factor, r_E is positioning error, \bar{r}_E is average positioning error, \bar{a} is average size of delineations, a_{MLD} is area of the minimum legible delineation, w_{MLD} is width of narrowest legible delineation, A is surface of study area, N is number of sampled points in study area, h_{ij} is spacing between closest point pairs, \bar{h}_{ij} is average spacing between closest points, h_R is range of spatial dependence, m is number of point pairs within range of spatial dependence, and l is total length of contours

Aspect	Coarsest legible resolution	Finest legible resolution	Recommended compromise
Working scale	$\leq SN \cdot 0.0025$	$\geq SN \cdot 0.0001$	$= SN \cdot 0.0005$
GPS positioning error	$\leq 1.8 \cdot r_{E(P=99\%)}$	$\geq \bar{r}_E \cdot \sqrt{\pi}$	$= 1.8 \cdot r_{E(P=95\%)}$
Size of reference objects	$\leq \frac{\sqrt{\bar{a}}}{4}$	$\geq \frac{\sqrt{w_{MLD}}}{2}$	$= \frac{\sqrt{a_{MLD}}}{4}$
Inspection density	$\leq 0.1 \cdot \sqrt{\frac{A}{N}}$	$\geq 0.05 \cdot \sqrt{\frac{A}{N}}$	$= 0.0791 \cdot \sqrt{\frac{A}{N}}$
Distance between points	$\leq \frac{\bar{h}_{ij}}{2}$	$\geq h_{ij(P=5\%)}$	$= 0.25(0.5) \cdot \sqrt{\frac{A}{N}}$
Spatial dependence structure	$\leq \frac{h_R}{2}$	$\geq h_{ij(P=5\%)}$	$= h_R \cdot m^{-\frac{1}{2}}$
Complexity of terrain	$\leq \frac{A}{\sum l}$	$\geq \frac{w_{MLD}}{2}$	$= \frac{A}{2 \cdot \sum l}$

Figure 6.2: [79] summarizes formulas to find the right pixel size for different types of data sources.

- **Coarsest legible grid resolution:** the largest resolution that should be used. Any resolution coarser than this coarsest legible resolution means that too much information will be lost.
- **Finest legible grid resolution:** the smallest grid resolution that represents the most (95%) of spatial objects or topography. This is the finest meaningful resolution which

ensure maximum location accuracy. Any finer resolution is probably just a waste of memory.

- **Recommended grid resolution:** a compromised resolution, usually set as the intermediate number between the coarsest and finest resolutions.

We explain some formulas from Figure 6.2 below:

- **Grid on GPS positioning:** A significant parameter is “confidence radius”, which is the radius of a circle where we expect most (P=95%) of the points to appear. The grid resolution area should not be smaller than this confidence circle.
- **Grid on remote sensing system:** To represent the smallest object and narrowest object, at least 4 pixels and 2 pixels are required respectively. The formula to classify the narrow polygon is $r = \sqrt{a/\pi}$, a = area of polygon, $S = p/(2 \cdot \pi \cdot r)$, p = perimeter of polygon. If $S > 3$, then the polygon is narrow.
- **Grid on point samples:** In the regular sampling situation, the coarsest grid size is about half of the average spacing of the closest pair, while in the random sampling situation, the coarsest grid should be about two times finer.

6.2.2 Error Estimation on Multi-resolution Emages

Errors in the output of a situation model may come from many factors such as data uncertainty in the data streams, data loss during data aggregation, and errors during Emage conversion. Many data streams often have uncertainty issues resulting from a missing value, network failure, or trustworthiness of the source. Numerous approaches try to handle this data uncertainty by using probability theory. To generate an Emage from data streams, we need to aggregate them over space and time windows. As a result, some of the original

data or attributes may be discarded. In addition, data streams are provided at a particular rate which may not match the users' requested rate for situation detection. The interpolation techniques are needed to convert data from one resolution to another or from one data structure to another. Errors during conversion are unavoidable but we can minimize them by selecting an appropriate Emage resolution for each situation. We present a process to estimate the error after the interpolation and rasterization process. This information can help users chose the right Emage resolution.

Interpolation

Spatial interpolation techniques are influenced by Waldo Tobler's first law of geography which says "everything is related to everything else, but near things are more related than distant things." Therefore, the correlation between data points that are closer is stronger than for those that are farther away. Several techniques such as inverse distance weighting, nearest neighbor, spline, and Kriging are widely used.

To create dense Emages from sparse Emages, an interpolation method is needed. Typically, a situation model with a finer Emage resolution (smaller grid's size) generates better results, but may not necessarily perform better after certain levels. A finer Emage resolution takes a longer time and more resources than the coarser one. We aid users in choosing the 'right' resolution by providing an estimated error at each Emage resolution. A method to estimate errors when using an interpolation on sparse Emage is shown below.

In the following experiment, we try to interpolate PM 2.5 concentration values obtained from the air quality measuring stations across California. This data source provides a sparse geo-spatial dataset. There are about 125 stations generating data every hour. To estimate errors from the interpolation method, we define the base resolution as 0.001 latitude unit x 0.001 longitude unit. Since we use a square grid, we will later use a single value to represent the

Algorithm 1 Error Estimation Algorithm for Emage Interpolation

- 1: Use historical Emages to find mean and standard deviation value at each Stel containing an actual value
 - 2: Use a Monte Carlo method to repeatedly simulate more Emages based on the statistic values found in previous step (note that these simulated Emage are still a sparse Emage like the original Emages)
 - 3: Find the finest and coarsest legible grid resolution (mentioned in the previous section)
 - 4: Define the target Emage resolutions based on the finest and coarsest legible grid in the previous step
 - 5: Define the finest legible grid resolution a base Emage resolution (ground truth)
 - 6: Apply an interpolation method on the simulated Emage to create dense simulated Emage at the target Emage resolutions
 - 7: At each Stel of the ground truth Emage, find an absolute error between its value and the interpolated value at the target Emages.
 - 8: Finally, for each target Emage resolution, calculate an estimated error from the standard deviation of the absolute errors found in the previous step
-

resolution, such as 0.001 unit. The target resolutions are 0.1, 0.2, 0.4, and 0.8 units. Figure 6.3 shows interpolated Emage results at different target resolutions. The Emage results using cubic interpolation, linear interpolation, and nearest neighbor interpolation method are presented from left to right.

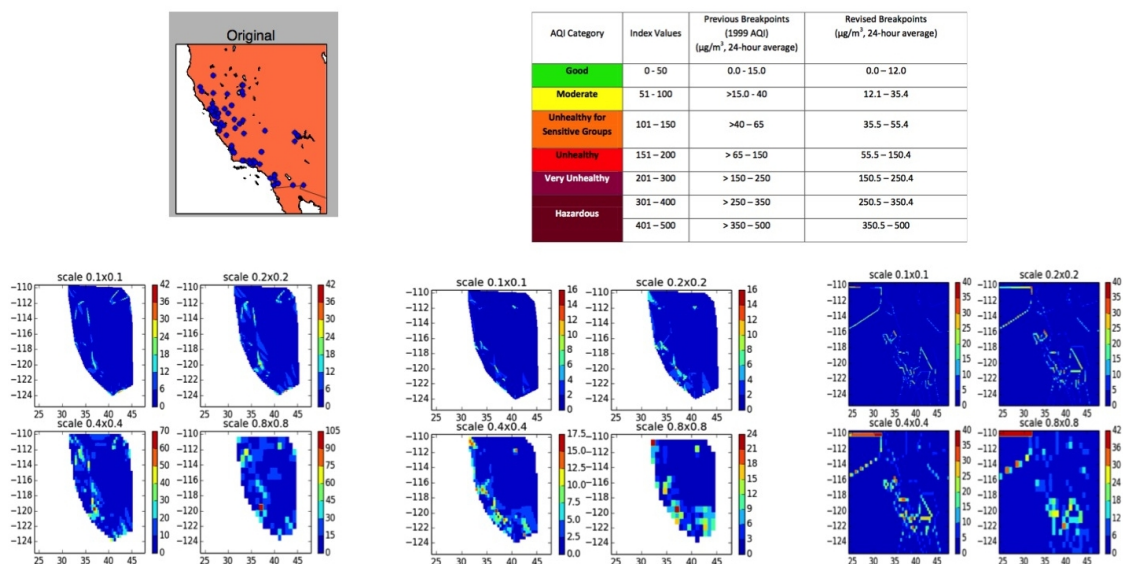


Figure 6.3: The interpolated results at one iteration in Monte Carlo. Three different methods are used to interpolate PM_{2.5} concentration values at four Emage resolutions: (left) cubic interpolation, (center) linear interpolation, and (right) nearest neighbor interpolation.

We repeatedly simulate an Emage for 1,000 iterations and apply the nearest neighbor interpolation technique to generate Emages at the base resolution and target resolutions. In each iteration, we calculate the standard deviation of the absolute errors between the base resolution Emage and the target resolution Emage, and plot the results in the graph as shown in Figure 6.4, which shows that Emages at the coarser resolutions provide more errors than Emages at the finer resolutions.

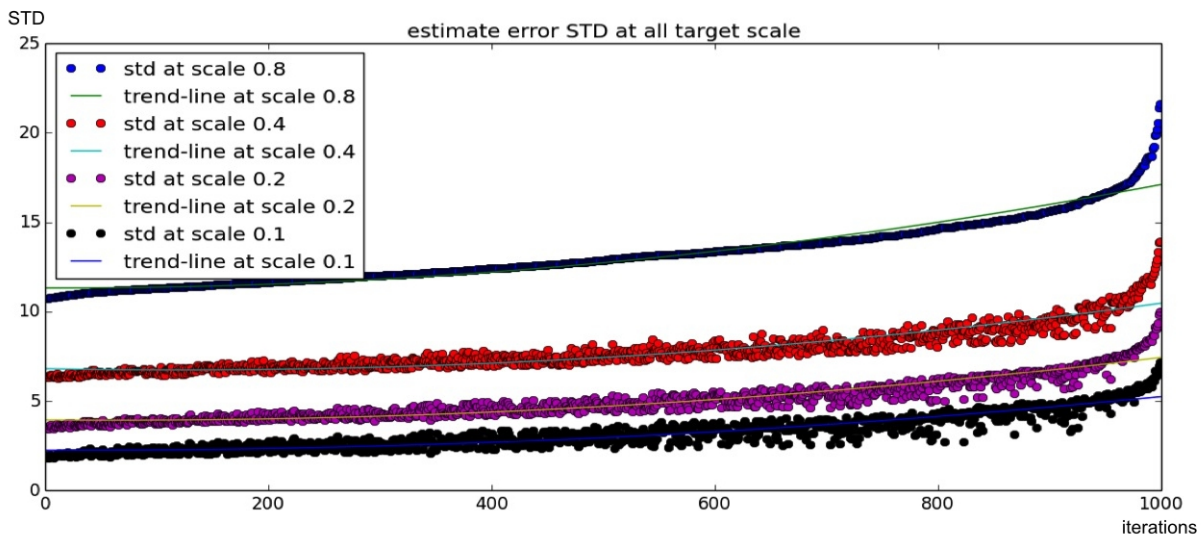
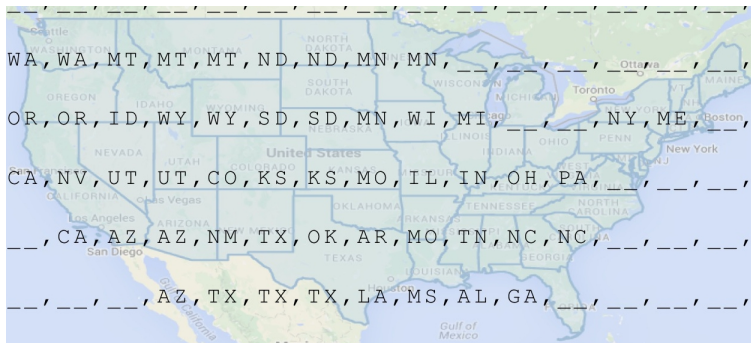


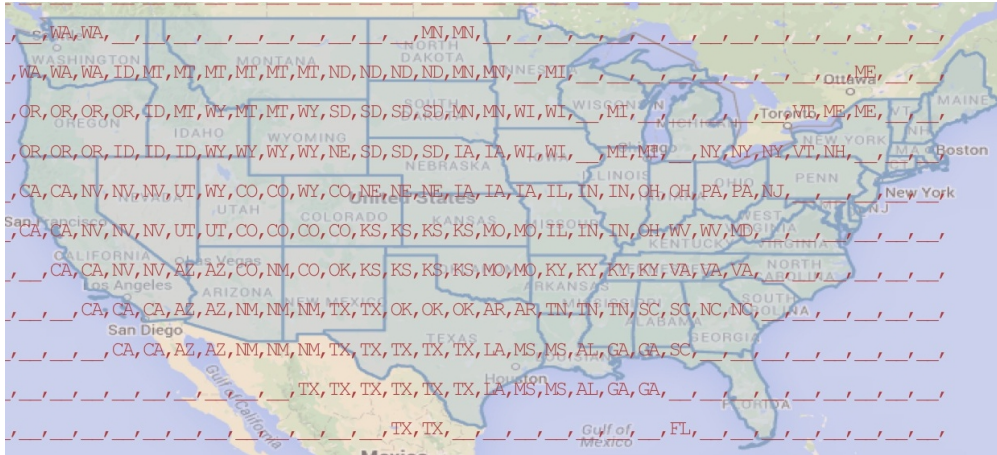
Figure 6.4: STD error estimation using linear interpolation method at four Emage resolutions

Rasterization

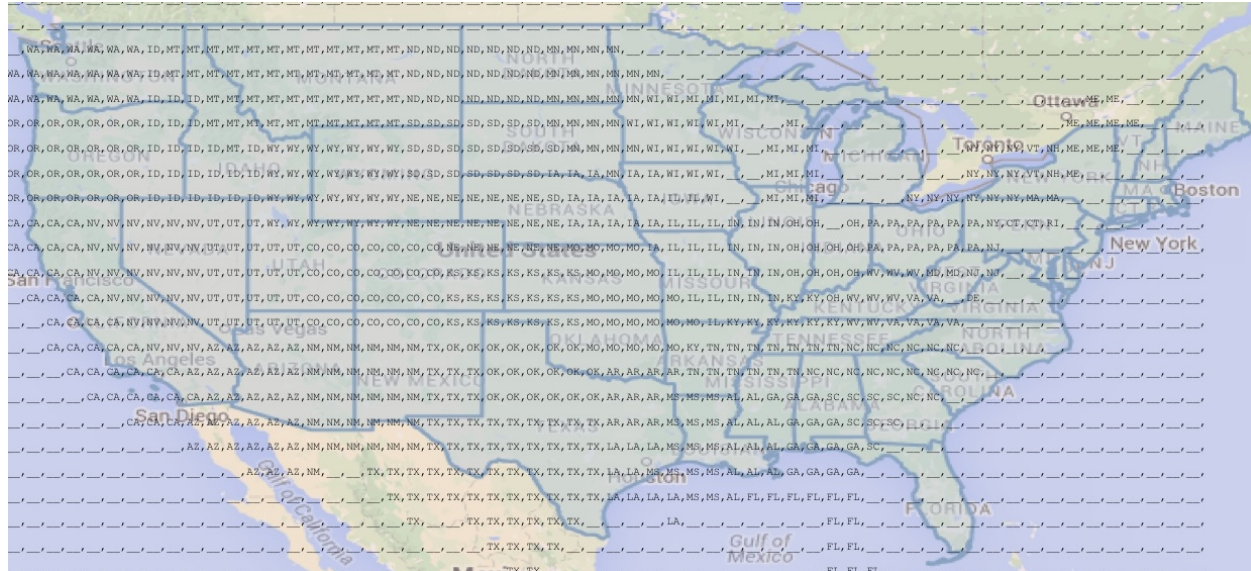
A rasterization method is used to convert polygon data type into a uniform structure such as grid or triangle. Data from data streams may represent values of the areas instead of a point coordinate. These data are, for instance, number of asthma patients in each county, flood affected areas, and election polls for each state. Figure 6.5 shows the results of rasterizing state boundaries into Emage at three resolutions: 4, 2, and 1 unit from left to right. Each Stel in Emage is assigned to a specific state whose the center point of Stel belongs to.



(a) at level 2 (unit = 4)



(b) at level 1 (unit = 2)



(c) at level 0 (unit = 1)

Figure 6.5: Rasterize state boundaries into Emage

Unlike interpolation method above, the errors of the rasterization method refer to the number of miss classified Stel. We use quad-tree data structure to store the localized errors as shown in Figure 6.6. Quad-tree is a tree data structure in which each node has exactly four children. It is suitable for this task since multi-resolution Emage has similar nature. Multi-resolution Emage partitions each Stel into four sub Stels to create a lower level Emage and aggregates four neighbor Stels to form a single Stel at the higher level.

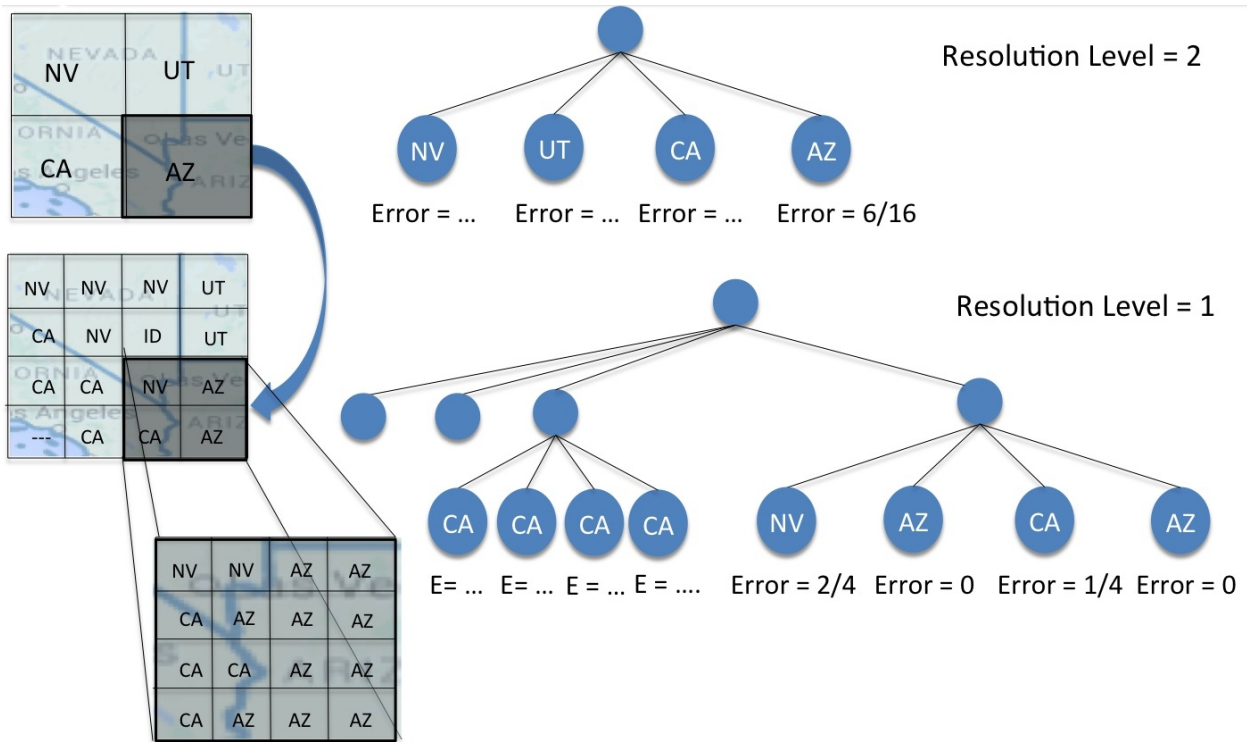


Figure 6.6: Localized error in quad-tree

Processing data at very high resolution may take too much time and resources. Users have to be able to identify the optimal resolution which has reasonable response time and small errors for detecting a situation. For example, we want to find the optimum resolution to raster US state boundaries. We define eight target Emage resolutions from 8, 4, 2, 1, 0.5, 0.25, 0.125, and 0.0625 unit. Figure 6.7 shows the graph between error estimation and Emage generation response time. We can see that the most optimum resolution is somewhere between 0.5 and 0.25. We have to mention that the resolution is not a one-size-fit-all resolution. It also depends on the characteristics of polygons and size of the areas as mention in [93]. For

example, the counties in the east coast of USA are much smaller in size compare to the counties in the west coast. Therefore, at the same resolution, the errors from a rasterization method on the east coast are higher, so that a finer resolution is needed to improve the Emage quality.

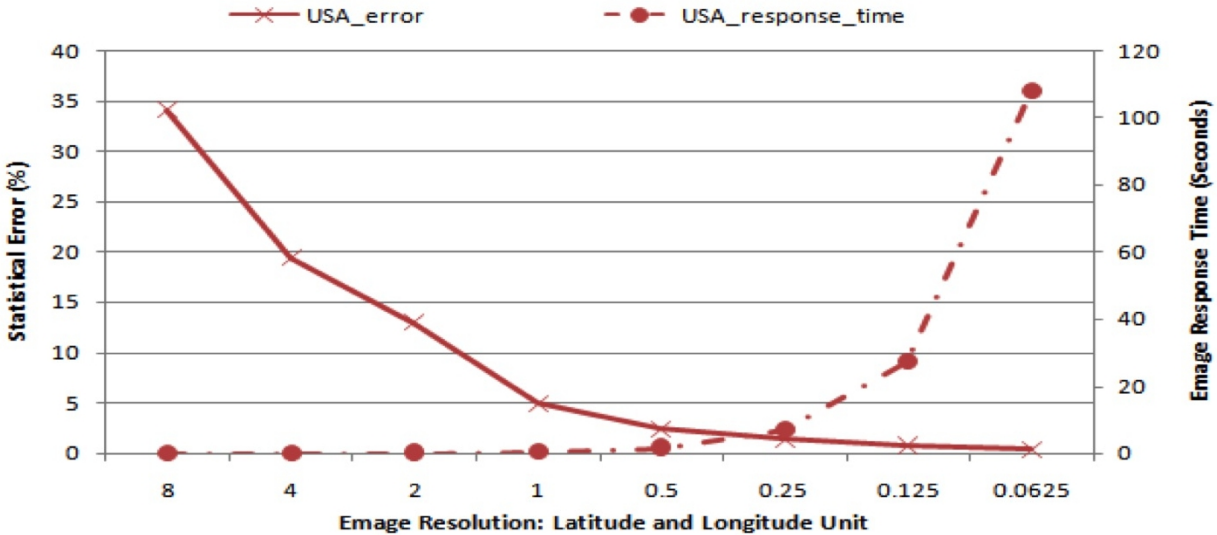


Figure 6.7: Error estimation vs Emage generation response time

6.2.3 Emage Stream Executor

Emage stream executor contains two steps: Emage conversion and query actor model as shown in Figure 6.8. When the Emage resolution of the Emage streams from the archival system are not combinable with the required Emage resolution of the situation model, an Emage conversion process is required. As mentioned in the Emage granularity section, some conversion methods are implemented such as sum, avg, min, max, and interpolation. The output Emages at the desired resolution are sent to the Emage Source in the query actor model. Similar to the archival system, we use an actor model to process the query. Emage sources, operators, and Emage Sinks are combined as a single query actor model. This model is continuously executed when the new Emage is arrived at the Emage sources as shown in Algorithm 2.

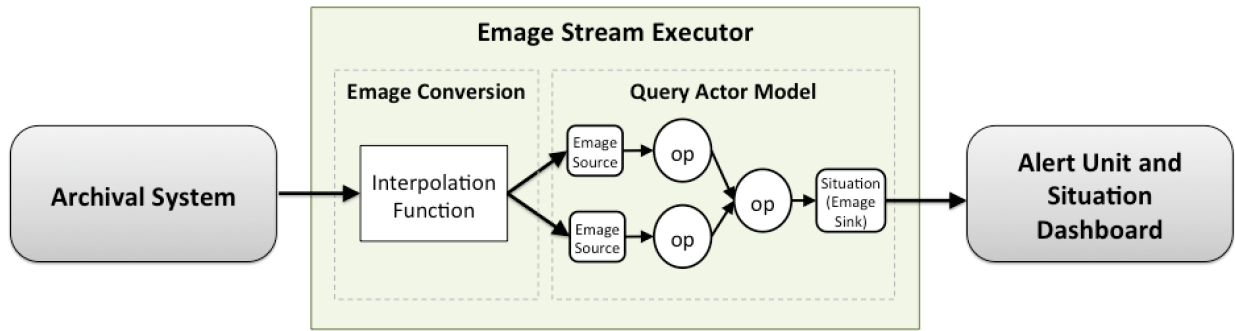


Figure 6.8: EventShop Emage Stream Executor

Emage Source

An Emage source is an actor that takes exactly one input and produce exactly one output Emage. It emits an Emage to an appropriate operator actor through a messaging pipeline.

Algorithm 2 Emage Stream Source

- 1: initialize()
 - 2: **while** true **do**
 - 3: $a \leftarrow get_next_emage()$
 - 4: output(a)
 - 5: **end while**
 - 6: *Note: $get_next_emage()$ represents the specific mechanism for obtaining the next Emage for the stream
-

Emage Stream Operator

An Emage stream operator performs a concrete operation when it receives input Emages and generates an output Emage. The input of each operator can be one or more Emage, but the output is single Emage stream. An output Emage is sent to the next operator as described in the situation modeling tree, except at the last operator in the situation model. The output from the last operator, which is the Emage of detected situation, is sent to a Emage sink to store, create alerts and visualize on a dashboard. Emage stream operators in

EventShop are inspired by image algebra operators and GIS operators. We describe some operators below. For more detail, please refer to this book [122].

Algorithm 3 Emage Stream Operator

```

1: initialize()
2: while true do
3:    $a_1 \leftarrow \text{input}(1)$ 
4:   .
5:   .
6:   .
7:    $a_n \leftarrow \text{input}(n)$ 
8:    $A \leftarrow \text{operate\_emages}(a_1, \dots, a_n)$  ▷ do operation
9:   output(A) ▷ generating result Emage
10: end while
11: *Note: operate_emages() represents a specific mechanism for processing input Emage(s)

```

- **Transformation (1 Emage \rightarrow 1 Emage):** takes an Emage from any Emage stream and transforms it into a new target Emage. For example, a value transformation (i.e., rebelling, normalization, interpolation).
- **Filtering (1 Emage \rightarrow 1 Emage):** takes an Emage from any Emage stream and filters all *stel* inside the Emage that satisfy certain predicates. User can filter Emage by its values or regions. To filter by time, the selection operator is used.
- **Aggregation (M Emages \rightarrow 1 Emage):** takes multiple Emages from two or more Emage streams ES_0, \dots, ES_n as input, and generate a new combined Emage stream ES' as an output. This is an stel-by-stel aggregation. Aggregate on the value.
- **Classification (1 Emage \rightarrow 1 Emage):** classify each stel in a given Emage based on a function and label it by a specific class accordingly. Classify by value (i.e., threshold, K-means) or region (i.e., states in USA).
- **Clustering (1 Emage \rightarrow 1 Emage)** groups similar stels in a given Emage together based on a clustering algorithm. Unlike classification operator, the number of groups are unknown and the classes are not predefined.

Emage Sink

An Emage sink receives an Emage of detected situation and performs final tasks such as situation logging, alerts, and visualization. They allow EvenShop to be interoperable with external systems. The algorithm is shown in Algorithm 4.

Algorithm 4 Emage Stream Sink

```
1: initialize()
2: while true do
3:   if has_new_image() then
4:      $a \leftarrow \textit{get\_next\_emage}()$  ▷ obtain next Emage
5:     sink_emage(a) ▷ send Emage to the sink destination
6:   else
7:     wait()
8:   end if
9: end while
10: *Note: sink_emage() represents a specific mechanism for sending output to the destination.
```

6.3 Emage Clustering Operator

Real-world situations vary in space and time scales. Some situations may occur over a long period of time and affect people in a large area such as air pollution, floods, hurricanes, and epidemic diseases. On the other hand, situations like traffic congestion, protests, and terrorist attacks may occur over a shorter period in either large or small area. Figure 6.9 shows that situations of different scales are not necessarily localized in time and space. Therefore, to detect situations from human reports (such as tweets, social media, and Flickr photos), the most common technique is clustering.

In the following section, we first review existing density-grid based clustering on data streams, and then introduce a new density Emage based clustering technique to detect evolving situations in EventShop.

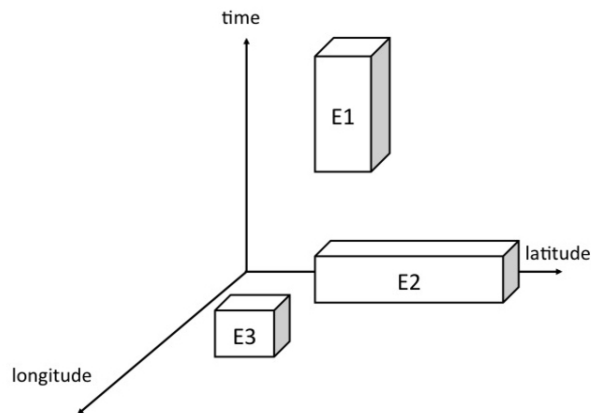


Figure 6.9: Situation representation in 2D space and time dimension. Situations of different scales are not necessarily localized in both time and space.

6.3.1 Density-grid based clustering on data stream

The unbounded nature of streaming data makes the traditional cluster algorithms on large but static data sets impractical. The synopsis concept was introduced to consolidate incoming data objects and capture current trends in data streams. Two commonly used structures are ‘micro-clusters’ and ‘grid structure’. We focus on the latter. By combining the density-based clustering with a grid structure, the data are mapped into a grid, and the grids are clustered based on density as shown in Algorithm 5.

Algorithm 5 Density-grid Based Clustering Algorithms

- 1: Define a set of grid-cells
 - 2: Assign objects to the appropriate grid cell and compute the density of each cell.
 - 3: Eliminate cells whose density is below a certain threshold t .
 - 4: Form clusters from contiguous (adjacent) groups of dense cells (usually minimizing a given objective function)
-

The well-known frameworks of density-grid based clustering on data streams are discussed below.

DUCstream [70]: Gao *et al.* developed an incremental single pass clustering algorithm for data streams. DUCstream processes the stream in a sequence of equal-sized chunks. Each chunk has to fit in the main memory, and contains a number of data objects. This framework

partitions the data space into a finite number of grids, and keeps only the grids that contain a high number of data objects inside. Three main parameters are required, including the size of a data chunk, grid resolution, and density threshold. If the number of data objects in the grid is higher than the density threshold, it is considered to be a dense grid. Then for clustering, DUCstream identifies the cluster as a connected component of a graph in which vertices represent the dense grid, and the edges are related to common attributes between two vertices. A depth first search algorithm is used in the graph.

D-Stream [52]: Chen *et al.* proposed a density-grid based approach for clustering data streams. This approach has both online and offline phases. In the online phase, a new arrival data instance is mapped into a density grid and a characteristic vector of that grid is updated. In the offline phase, dense neighboring grids are merged together in order to form clusters. In addition, to capture dynamic change in the data stream, the *decay factor* is introduced by assigning more weights on recent data instances without removing any historical information.

D-Stream II [140]: Tue *et al.* extended the D-Stream approach and introduced *attraction* of grids concept. This concept uses the position of the data object in each grid to calculate attraction between two grids. Before merging any two neighboring grids to form a cluster, D-Stream II checks the attraction of two grids. If their grid attraction in both directions is higher than a given threshold, these two grids are strongly correlated and they will be merged. The attraction of two grids is asymmetric. At a given time t , two grids g and h are strongly correlated if both $attr(g, h, t)$ is greater than a threshold and $attr(h, g, t)$ is greater than a threshold.

DENGRIS-Stream [30]: This is a first density grid-based clustering for a data stream over a sliding-window. It introduces the expired grid concept to detect the grids whose time stamps are not in the sliding window. These expired grids are removed before any processing takes place. As a result, the processing time and memory are decreased.

Excc (Exclusive and Complete Clustering) [44]: This algorithm can handle heterogeneous data streams that contain both numeric and categorical data. It has online and offline phases. The online phase creates a synopsis in the grids and the offline phase forms the clusters. The numerical attributes are mapped to the grid directly. The categorical attributes are assigned granularities according to distinct values.

In this thesis, we propose a density Emage clustering operator in order to recognize or detect evolving situations from STT stream. The algorithm is shown in Algorithm 6 below.

Algorithm 6 Density Emage Clustering

```

1: DEFINE target event using 'bag of concepts' (BOC)
2: SET density threshold (number or %)
3: SET similarity threshold (%)
4: SET signal-to-noise ratio threshold (%)
5: for each Emage do
6:   Find significant Stels (its density  $\geq$  density threshold)
7:   Sorting the Stels according to their densities
8:   Identify cluster centers
9:   Traversal of neighbor cells to create clusters
10:  for each cluster do
11:    Identify which clusters are associated with the target event
12:    calculate the similarity between each STT in the cluster and BOC
13:    if similarity value  $\geq$  similarity threshold, STT is associated with the event then
14:      if the ratio between no. of associated STT and total STT in the cluster  $\geq$ 
SNR threshold then
15:        This cluster represents the target event
16:      end if
17:    end if
18:  end for
19: end for

```

First, we define the target event using ‘bag of concepts’ (BOC) and set some parameters including density threshold, similarity threshold, and signal-to-noise ratio (SNR) threshold. The density threshold can be an actual number of STT points or percentage. Then, for each Emage in an Emage stream, we find all significant *Stels* whose density value are higher than the density threshold. For example, if the density threshold is 200 STT points and the Stel contains 250 STT points, then this Stel is a significant Stel. We sort the significant

Stels according to their densities from high to low and identify the cluster centers. To create clusters, we traverse to the cluster centers' neighbor cells. For each cluster, we identify whether this cluster is associated with the target event or not. We calculate the similarity value between each STT points in the cluster and BOC using cosine similarity function. If the similarity value is higher than the threshold, the STT point is associated with the target event. Finally, if the ratio between the number of associated STT points and the number of all STT points in the cluster is higher than the threshold, this cluster represents the target event.

Experiment: Detecting City Floods from Flickr Photos

We apply the density Emage clustering technique explained above to detect emergency situations such as city floods. The dataset in this experiment is the Yahoo Flickr Creative Commons 100M (YFCC100M) dataset [137]. This dataset contains a list of photos and videos under the Creative Commons license. A deep-learning approach was used to detect 1,570 concepts in the photos (such as people, animals, nature, buildings, sky, and scenery). We believe that an informative reports about real-world situations can be obtained from these photo concepts along with photo taken time and locations. EventShop ingests these photo dataset, generates STT data and Emages, and analyzes them to detect situations.

First, we define the flood event model using the bag of photo concepts. From the historical data and human knowledge, the concepts of 'outdoor', 'water', 'road', and 'car' are highly co-occurring together during a city flood. In this experiment, we try to detect a city flood in Thailand between July 1, 2011 and June 30, 2012. The number of Flickr photos taken during this period are shown in Figure 6.10.

Then, we apply clustering method mentioned above to identify which clusters are associated with a flood event by monitoring the frequency of its photos and measuring the similarity

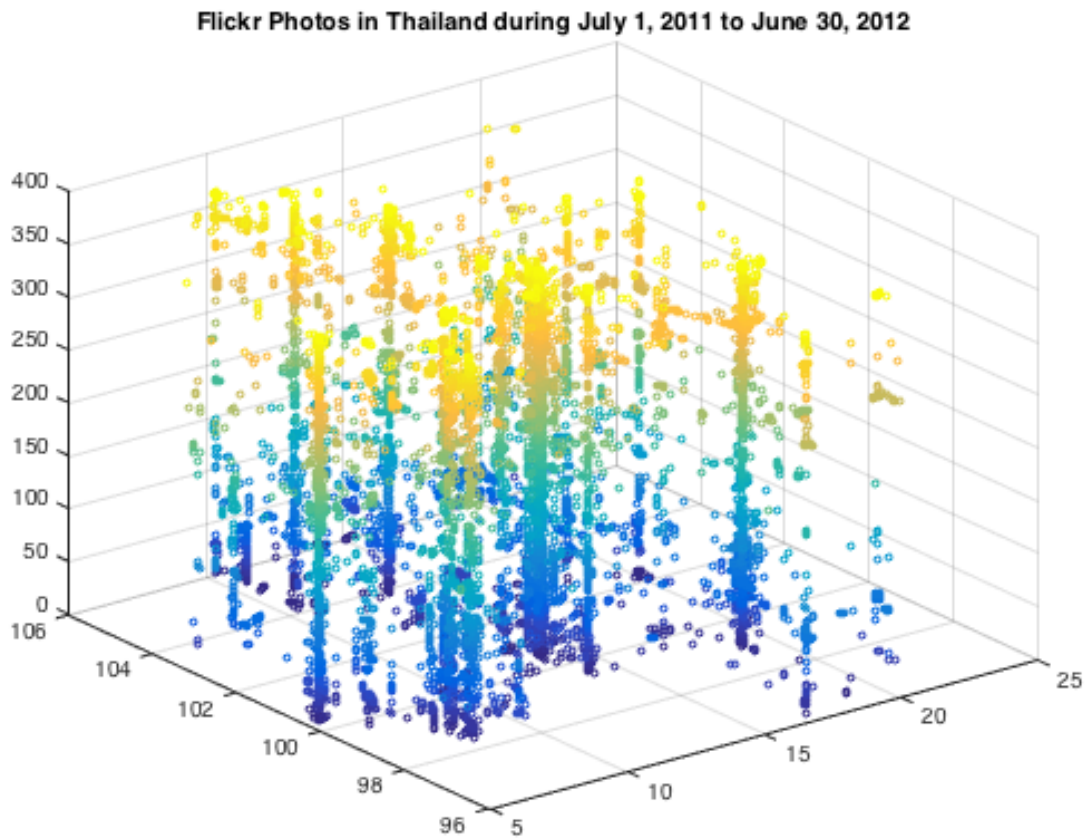


Figure 6.10: Flickr photos in Thailand from July 1, 2011 to June 30, 2012. The X, Y and Z axes represent latitude, longitude, and day, respectively.

between photo concepts and the flood event model. Finally, the cluster is identified as a flood event when the ratio between flood photos and all photos is higher than the SNR threshold. Figure 6.11 presents a set of photos in one cluster which is classified as city flood event. The location of this cluster is near Don Meuang international airport. The photos were taken on November 5, 2011. The length of edges in the graph represents a similarity between a flood event model (rectangle) and a photo (circle) considering its visual concepts. A shorter length means a higher similarity. The figure shows that this cluster contains a significant number of photos related to flood. This area is actually affected by flood as shown in the flood-map

across Thailand¹. The proposed algorithm can successfully detect several city flood events in Bangkok from November 4, 2011 to November 23, 2011.

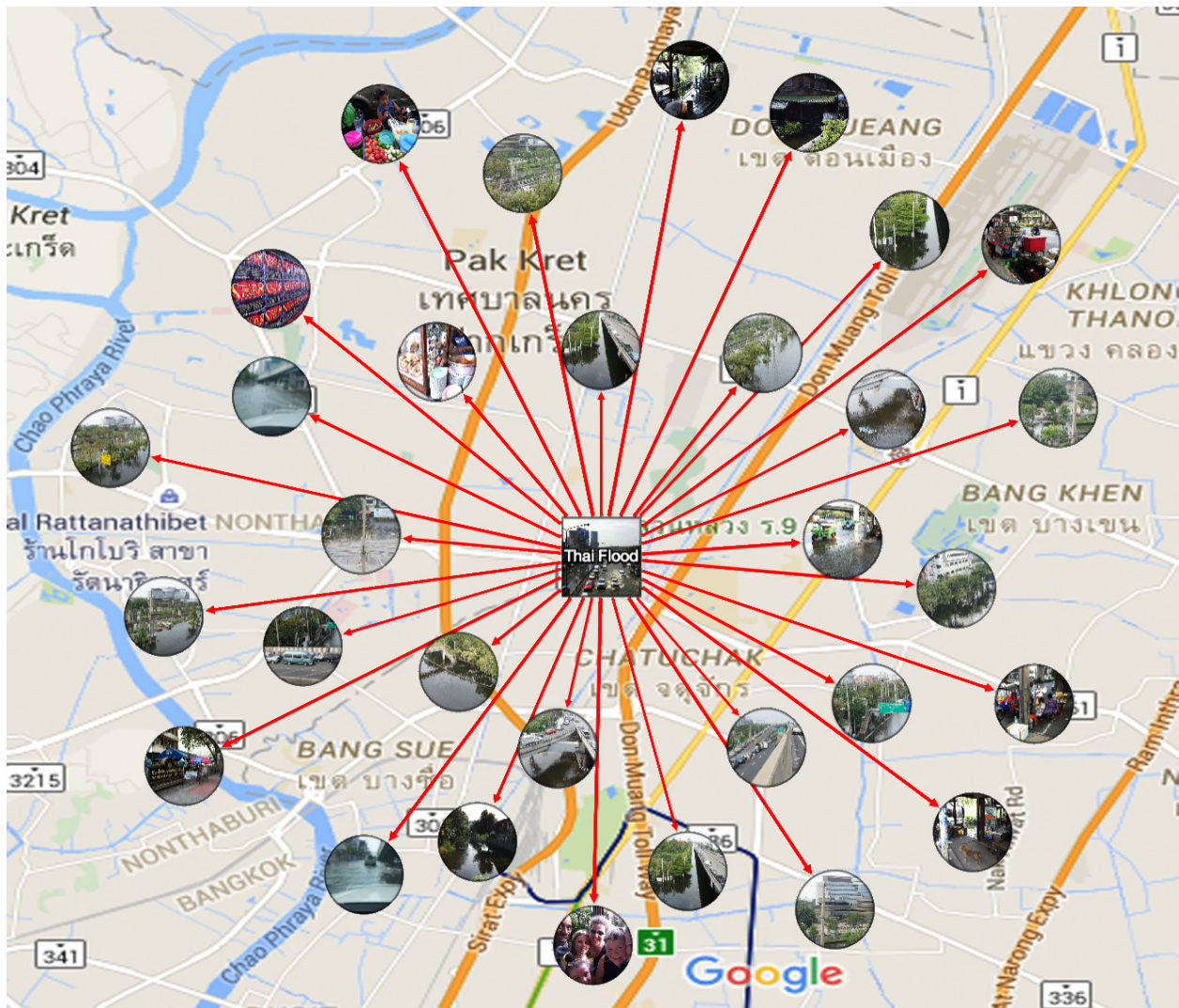


Figure 6.11: Photos from a cluster which classified as a flooded area in Thailand on November 5, 2011.

¹www.thaiflood.com/floodmap

Chapter 7

Micro-Reports

So far, we have discussed how to assimilate STT data streams to detect evolving situations. Now, we focus on the data sensing or data input mechanism. As we mentioned in Chapter 4, the data sources for observing real-world situations are classified into two categories: machine-centric sensing and human-centric sensing. Since the data from the latter category are too broad and extremely noisy, working with these data requires tremendous work in cleaning data, extracting conceptual meaning from a free-text (unstructured data), and estimating missing contextual value such as locations. In this chapter, we introduce a novel mechanism platform called, **Micro-reports** [111], which allows citizens or reporters to observe a surrounding event and report it as truly and fully as possible. This mechanism can create more powerful multimedia participatory sensing using a combination of data from multiple sensors in smartphones and other contextual information.

Participatory sensing utilizes the eyes and ears of citizens to collect information about a situation. In the last decade Twitter emerged as the most popular participatory sensing mechanism to find information about events as well as understand situations and trends. As is well known, Twitter introduced the concept of micro-blogs for sharing short 140 character

message to share one's opinion on events or situations. These micro-blogs were suppose to provide reports on events. And tweets have been used to detect situations and trends. However, getting objective information from tweets has been a problem. Twitter is too broad and subjective for most applications [38]. Crowdsourcing has been widely studied and considered useful for tracking the latest status of situations [69, 74] especially in disaster or emergency situations. For the changing demographics of the society and now significantly more powerful smartphones, it is possible to create more powerful multimedia participatory sensing approaches using multiple sensors and other contextual information.

Most people use their phones for reporting and getting useful information from collected and processed reports by other people. Camera is the main reporting instrument, like pen used to be, for reporting. In fact, smartphones are ideal devices to create spontaneous micro-reports. Just a few years ago people used text for reporting; now they hardly use text. Many sensors, particularly the camera, facilitate spontaneous capture of data and information in context and use them, without any text, as reports. This is a rapidly growing trend. There were one trillion digital photos captured in 2014 and more than 75 percent of those were captured and shared using mobile phones. Photos nowadays play dual role in human experience communication. Photos captured as traditional *Kodak moment* are for remembering and keeping records of special events. They are very valuable to us. We want to preserve details of content and experiences in each photo. However, photos have another, possibly even more dominant role. Photos are increasingly used as a compelling and descriptive universal information creation as well as consumption source. Photos communicate independent of language. Photos capture a total moment so the data contains much more than a narrow perspective or interest at the moment. Photos are captured for detailed descriptions as captured in oft use phrase *a picture is worth a thousand words*. This aspect of photos has remained in the domain of special applications such as medical imaging, remote sensing, and plant disease detection. With tremendous changes in the way we now capture photos,

information element aspect of photos is likely to become at least as important as the memory aspect.

Can we look at dense geo-spatial streams of photo-reports in their context and use the fact that each photo is worth a thousand words to understand subtle as well as higher level, possibly obvious, trends and situations? In this chapter, we explore the use of photo as micro-reports to detect evolving situations in real world. Using YFCC100M data set[138], 100 million photos from Yahoo-Flickr, several experiments have been conducted to validated the importance of photos as micro-reports. Encouraged by that, we are building a framework to import photo-reports from diverse sources and explore their efficacy in applications like situation awareness, trend analysis, and cultural dynamics. For the data analytics on these reports, we use EventShop platform, and consider photo-reports as an information stream from people to detect situations and trends by combining it with other streams. Our initial experiments clearly demonstrate the potential of these micro-reports for important applications. We discuss limitations of micro-blogs and show why micro-blog based applications, such as those using Tweets, could be dramatically improved using micro-reports.

7.1 Limitations of Micro-Blogs

Citizens normally use variety of social media applications to broadcast their opinions and capture their moments as shown in Figure 7.1. Twitter and Facebook Status updates are the most popular examples. At first, people could not see the value in micro-blogging, but soon it became clear that aggregation of information in micro-blogs could be valuable. Though Twitter has proved to provide the first line of information to people during emergency situations, it was not originally designed for detecting real-time situations. Twitter became one of the most successful social and journalistic applications in the last decade. It has become the source of latest alerts, news and popular trends in most areas. In academic

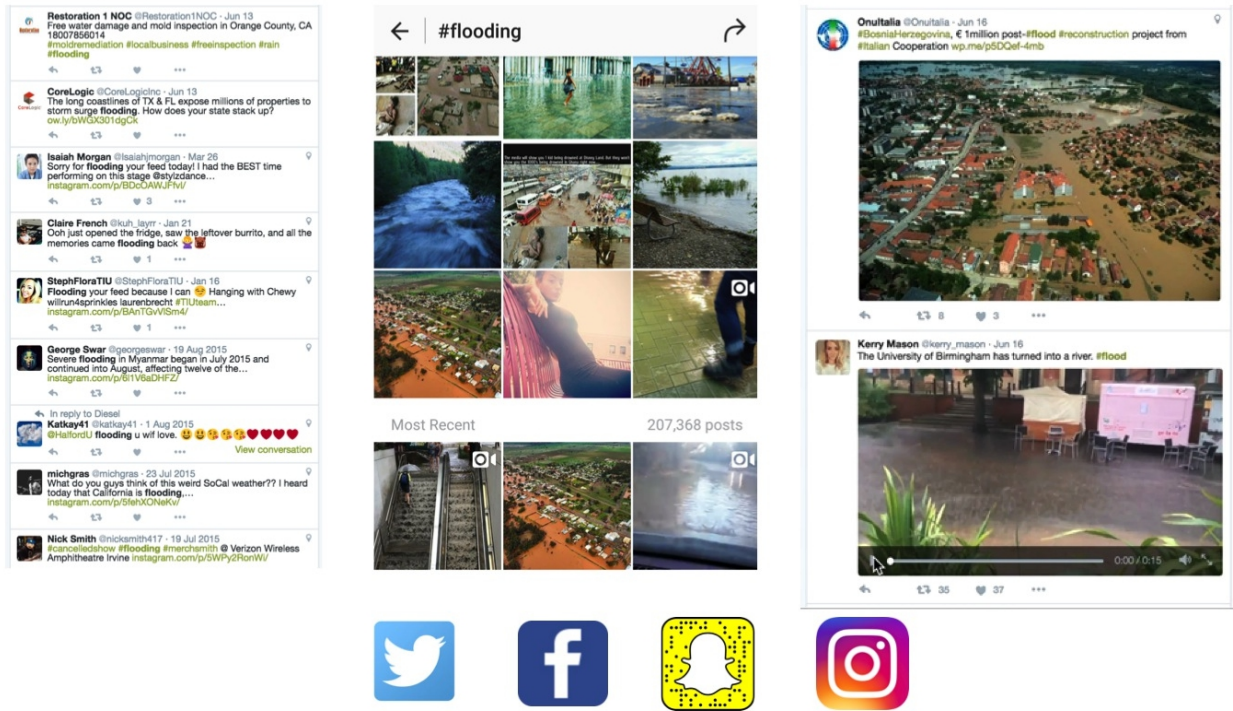


Figure 7.1: Reports of events from citizens through micro-blogs technologies

world, many research papers have shown how tweets could be used for important social analysis. Despite all the success, there are three most critical problems in using this type of micro-blogs to understand evolving situations: *missing/misleading context*, *noise*, and *subjectivity*.

First, the lack of explicit geo-tagging on a large scale of micro-blogs can undermine the accuracy of situation recognition. This can be considered as insufficient data cause by incomplete gathering of information, however misinterpretation in data can also become an issue. An actual location of a posted micro-blog does not always correspond to the location of the published contents. For example, one of the data set from Web Observatory [139] has more than 6 million tweets related to London Olympic Game in 2012. Only about 1% of tweets are geo-tagged, and only 0.1% are tweeted by users within London.

Second, social media applications usually have very low signal to noise ratio [38, 61, 89, 115]. People tweet more than 100 million times a day, yielding a noisy and informal corpus of

140-character. Extracting accurate and useful information requires significant effort in data pre-processing. The performance of traditional NLP tools and named entity classifications is severely degraded on tweets. Tremendous works e.g., [94, 144, 105, 95] have been done in using Latent Dirichlet Allocation (LDA) [45] to detect the topic and sentimental analysis to estimate the feeling of the tweets' content.

Third, the information is usually subjective and limited to textual format. In order to make a right decision, people need to know that the information they receive is a fact or an opinion. In journalism, journalists need to know how reliable statements are before they can report them. Reporting an opinion as a fact is unacceptable. In tweets, the capability to distinguish between facts and opinions is lacking.

For century, in journalism philosophy, the reports must have these characteristics. Truthfulness, accuracy, objectivity, and fairness. All in all, the goal is to “seek the truth and report it as fully as possible” without reporters' own opinion. As a result of these three limitation, micro-blogs totally *violate all journalism philosophy*. In our past study [121], persistent location sensor data, air pollution data and pollen data were combined with citizen contributed Twitter data, to study the situation of asthma risk. The accuracy of situation detection depends on the quality the of data. Micro-blogs from systems like Twitter are very noisy and subjective. Most applications may benefit from objective and factual citizen contributed data.

7.2 Micro-Reports and Beyond

To tackle these problems, we are exploring use of **micro-reports** as compared to **micro-blogs** for radical improvement in the quality of participatory sensing data. Unlike micro-blogs, micro-reports serve like eyewitnesses and field reporters of the events which are more

reliable. Among diverse citizen contributed data, photo and video are the biggest data source. Every minute, millions of photos and videos are uploaded by people to social media. More and more photos and videos have timestamp and geo-tag information. With the right tool, photos and videos are more than visual documents. They become the best resources of micro-reports because of their objective and factual nature.

7.2.1 Basic Characteristics

The following important characteristics are required for micro-reports:

- **Spontaneous:** Micro-reports should be shareable immediately. If creating a report requires thinking and typing a message, fewer people will use it during the event. A report submitted later relies on memory and may not have correct contextual information.
- **Objective:** Micro-reports should emphasize on facts and not subjective opinions of people and their personal attitude toward the facts. Opinions are important, but should appear explicitly as opinions. Objective and subjective components in a report should be clear.
- **Compelling:** Micro-reports should effortlessly capture the moment with rich context along with compelling photo, audio, video and/or other signals. As are well known photos, videos, and audio which make a report of an experience more compelling and more informative.
- **Universal:** Micro-reports need to have a universal language, that is beyond literacy and geographical boundaries. Photos are universal language.

7.2.2 Emerging Challenges and Opportunities

To make use of all benefits of micro-reports, the existing platforms for social media are unsatisfactory. Because of the unique characteristics of micro-reports that we mentioned, new challenges and opportunities in emerging applications are addressed here.

Situation Recognition

For multiple decades, researchers in multimedia community have been developing approaches like entity resolution, object detection and scene recognition, to make sense of a single media. Now, with trillion of micro-reports, real-world phenomena are being observed by multiple media streams. For example, beyond creating a tree detector and testing it over millions of photos, we can use those millions of photos as micro-reports streams and combine them with other available data to detect plant diseases spreading pattern. In [143], visual report from physical surveillance cameras and social tweets are combined to detect evolving real-world events.

Trend Analysis

Trend analysis in social media has been a significant interest for many researchers. Traditionally trends can be defined as the frequently mentioned topics throughout the stream of social media from temporal point of view. The new challenge is to incorporate both temporal and spatial characteristics along with themes or topics from micro-reports for a better understanding of trends. The result of trend analysis can also be used in creating a situation model to recognize and predict situation in real-time.

Cultural Analytics

A culture in any cities is evolving. With the current mobile and social technologies, an opportunity to learn the culture from the eyes of city crowds has arrived. The term “cultural analytics” was first introduced by Lev Manovich. In his work [82], an informative visualization offers social, cultural and political insights about people’s activities from Instagram photos. We need to develop a computational and visualization platform for the analysis of large sets of cultural artifacts beyond text.

7.2.3 Applications from Situation Recognition

Situation recognition combines heterogeneous data streams ranging from static information, stationary sensors, and human generated contents to get insight evolving situations. Quality of data affects the quality of situation detection. EventShop has been addressing different issues due to sparse availability of data as well as use of micro-blogs. By improving quality and density of data from participatory sensing, we can improve the quality of situation detection and understanding as well as sending right information to right people. We described example applications in three different domains to demonstrate the benefits of using micro-reports in situation recognition framework.

Public Health

Social media in public health gains increasingly high attentions in the past few years. It is becoming a common platform for clinicians and public health officials to share information, track diseases, and predict outbreaks. In [88], most of public health mapping tools relying on Tweets, Propeller Health ¹ is a distinguished application that uses micro-reports paradigm

¹<http://propellerhealth.com>

instead of traditional micro-blogs. Propeller health maps asthma triggers and identifies the severity of asthma attacks from the uses of inhalers equipped with special tracking sensors. Combining these focused and high quality data with other environmental data sources can create the medical knowledge on environmental asthma triggers at both individual and society level.

Emergency Management

Emergency reports from social media have proven to be the first line information for people to get the latest status of any emergency situations, for example, hurricane and flood detection, disaster mitigation, flu outbreak, and wildfire detection. In the recent incident, the outbreak of Zika virus disease began in April 2015 in Brazil, and spread to other major countries in South America, Central America and Caribbean. In February 2016, World Health Organization (WHO) declared that cluster of microcephaly cases reported in Brazil was a public health emergency of international concern. They could have contained the outbreak within Brazil if there were a better technology for people to create focused report and for officers to detect the disease outbreak sooner.

Smart Cities

Some example applications include “Smart Cities”, where citizens and local governments alike can report on, and get notifications about air quality, cleanliness, crime, reports and agriculture. One such application is Waze², a community-based traffic and navigation app that displays real-time traffic situation. It uses objective crowd sourcing information from GPS sensor of user’s smartphone. Drivers become alerted before they approach police, accidents, road hazards or traffic jams, all shared by other drivers in real-time. Another

²<https://www.waze.com>

application is *SLN4MOP*, a next generation social networks that have the capability of providing real-time, context-sensitive local information by aggregating data from Sri Lankan farmers through their mobile phones [72].

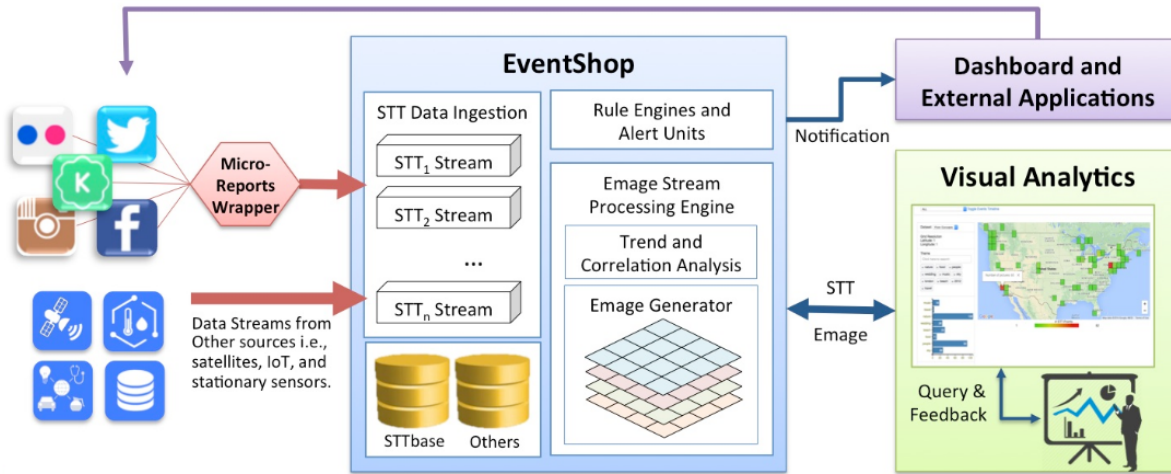


Figure 7.2: From micro-reports to situation recognition

7.3 Situation Recognition from Micro-Reports

To detect situations from massive volumes of micro-reports, we extend the EventShop framework and implement a new visual analytics tool as shown in Figure 7.2.

First, micro-reports from variety of social media are transformed into *events* represented in a JSON format. We believe that events are a much better abstraction of human experience and provide a more appropriate means for managing micro-reports. This JSON is designed to capture six facets in E-Model [145]: temporal, spatial, informational, experiential, structural, and causal. An example of micro-report in JSON form is shown in Figure 7.3.

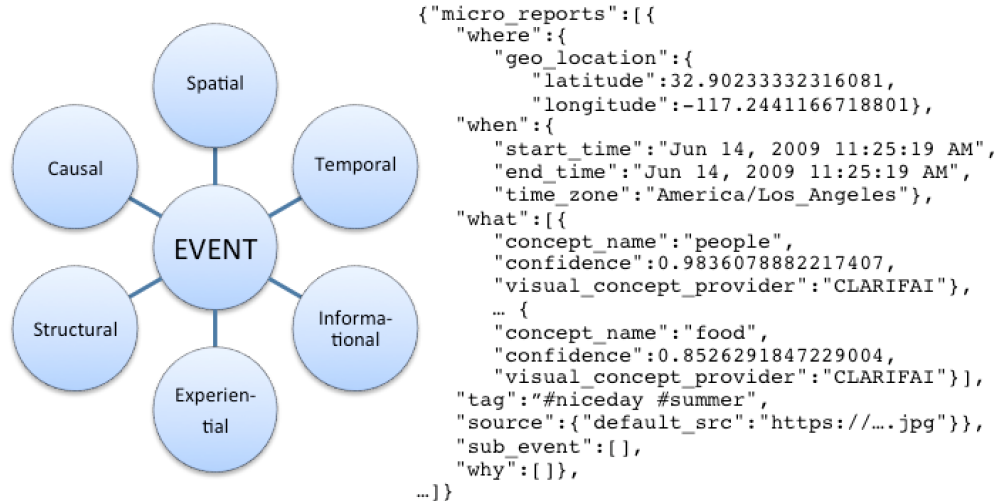


Figure 7.3: Micro-reports in E-model

7.3.1 Micro-Reports Wrapper

To extract temporal and spatial facets from photo and videos media, we rely on timestamp and geo-location from the GPS of the capture devices. For informational facet, We not only use user’s tag, but also automatically classifying the media contents into a number of predefined concepts [50]. The experiential facet is the media sources themselves. Note that some of the facets may not be inferred at this step. For example, the causal facet which is required more understanding in qualitative causal modeling [90]. As of our knowledge, Krumb’s³ micro-report comes closest to capture the intent of the users with the use of Emojis instead of a single shutter button.

7.3.2 Visual Analytics

Our visualization is not only designed for summarizing and retrieving information from large volume of photos, but also facilitate an experimental environment to analyze micro-reports along with other data streams. It is a novel visual analytic tool where the strength of human

³<https://krumbs.net>

and machine are combined to shed the light on unexpected and hidden situations. We ease users to explore micro-events streams with zooming, filtering, and aggregating abilities in three dimensions: space, time and theme. Our user interface contains five main components as shown in Figure 7.4: query control, interactive map, interactive timeline, theme chart, and media gallery.

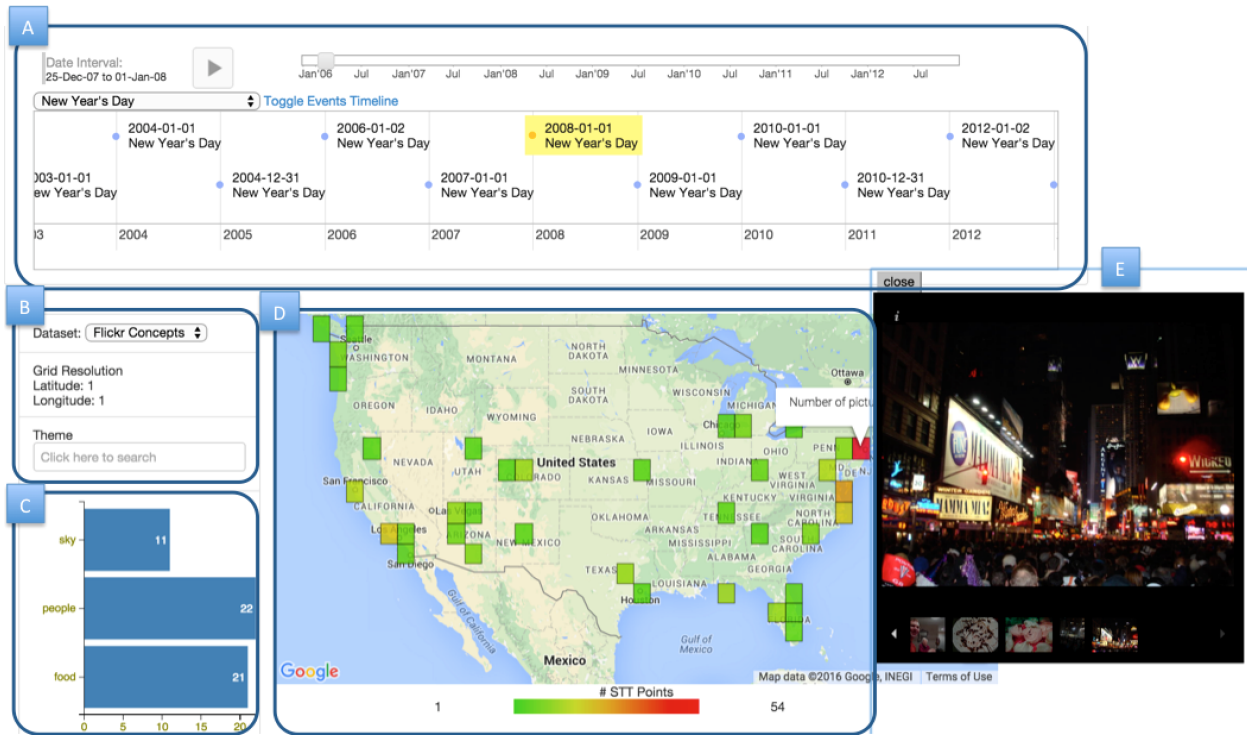


Figure 7.4: Visual analytics tool for Micro-reports: (A) interactive timeline, (B) query control, (C) theme chart, (D) interactive map, and (E) media gallery

We developed this UI to work with EventShop. EventShop aggregates the STT stream according to the frame parameters (bounding box, latitude/longitude units, and time-window) to create an Emage stream. Emage is an event-image where each pixel reflects social interest of that specific theme at particular location and time, for more details please refer to [108].

We visualize the information from micro-reports in the center of our UI, Figure 7.4(D). Users can spatially explore the aggregated micro-report (Emage) over the map. When users detect any interesting incidents, they can zoom-in into particular regions and analyze further. The grid's color represents the total number of photos ranging from green (low) to red (high).

By moving mouse over the grid box, the actual number is displayed. While clicking on the box, the media gallery, Figure 7.4(E), is shown. In this gallery, further detail of each micro-reports can be explored.

On the top, we provide an interactive timeline, Figure 7.4(A), which allows user to select the Emage over time. Users can also click “play” button to see all Emage in animated fashion like a video. On the left side, we provide a query control, Figure 7.4(B), where users can select specific micro-report source such as Flickr or instagram, changing latitude and longitude scale of the Emage grid, and filter for some particular themes. The theme chart, Figure 7.4(C), shows the number of photos with the associated theme. This chart is changing over time, and location. In our example, we can see that number of photo in New York during New Year’s day is significantly higher than other areas. When we hover our mouse over the New York grid, the theme chart shows that most of the photos are about sky, people, and food. Then, we can open up the media gallery to see more details about these micro-reports. The demo of the tool can be found at <http://slnlab.ics.uci.edu/eventshop>.

7.4 Micro-Reports Proof of Concept

An important characteristic of these geo-tagged photo data is its space-time nature. For example, when a public event occurs, people take photos related to the event, which enables detection of event occurrence promptly, simply by observing the photo. In this paper, we investigate the interaction of events, and develop a framework that treats photos as micro-reports to understand evolving situations. Yahoo Flickr Creative Commons 100M (YFCC100M) dataset [137] is used to demonstrate our idea. This dataset contains a list of photos and videos under Creative Commons license. A deep-learning approach was used to detect 1,570 concepts in the photos (such as people, animals, nature, building, sky, and scenery). We use these concepts to create informative and objective micro-reports along

with their location and time. Thus, EventShop can convert photo and video to STT data and use them as powerful data streams for situation recognition.

We are interested in detecting meaningful events reported by users through photos. Since time and space are the crucial components in our study, photos with invalid timestamps and unknown geo-location are filtered out, leaving us with about half of the photos (48.5 millions).

7.4.1 Detecting Evolving Situations from Concepts of Photos

Unlike tweets and user tags which are subjective and noisy, photo visual concepts generated by advanced deep learning technique are objective and standardized. Our experiments provide a proof of concept in detecting real-world situations using these visual concepts.

Detecting Olympic Games

From the year 2010 to 2014, photos in the bounding box of London (-0.489,51.28 0.236,51.686) are selected. Figure 7.5 shows the total number of photos taken changed over time. There are three big peaks which can be caused by some situations. To understand such situations, we zoom into the concepts of photos. For each day, the probabilities for the same concept is aggregated. Figure 7.6 gives an example of how four concepts, people, running, sport and swim evolve during the time of interest. Let's first look at the dynamic change in the 'people' concept, we observe that London is less crowded in winter season and attracts more people in summer and autumn. London usually hosts many kinds of sport events, such as swimming, and running, which also can be detected in this figure. However, single data stream ('people' theme) finds itself hard to detect situations.

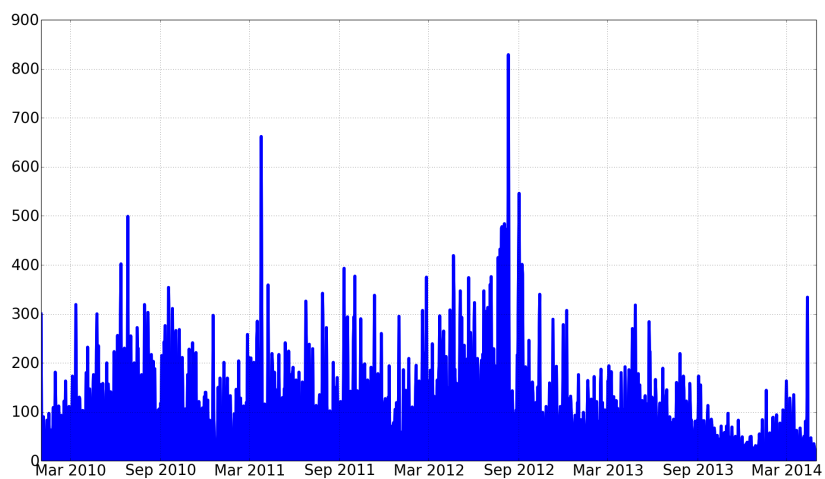
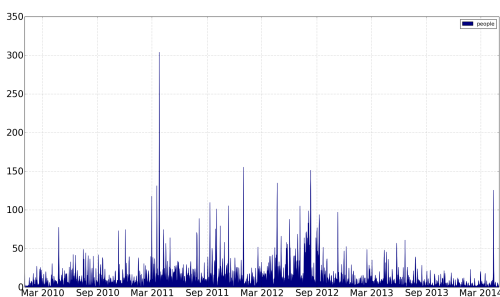
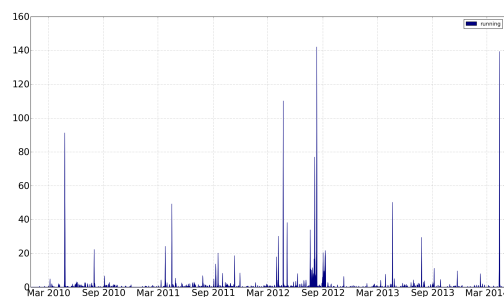


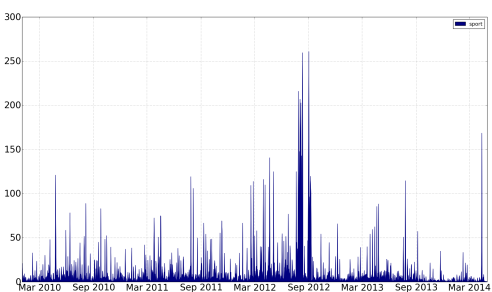
Figure 7.5: Number of photos taken in London



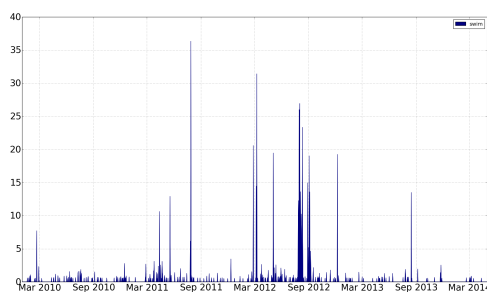
(a) people



(b) running



(c) sport



(d) swim

Figure 7.6: Timeline of concepts

Social events and trends are usually detected by the concurrences of photo's visual concepts. Figure 7.7 shows the evolving concepts in Beijing in the year of 2008. As we

know, Beijing Olympic Games was held in Beijing in the summer of 2008. This event is clearly reflected in Figure 7.7 where the number of these ten concepts ‘basketball’, ‘court game’, ‘gymnastics’, ‘people’, ‘sport’, ‘stadium’, ‘swim’ and ‘tennis’ reaches peak at the same time of Olympic Games. Thus, the event model of “Olympics Games” can be defined as bag of these photo visual concepts. By computing the similarity between the “Olympics Games” event model with evolving *bag of visual concepts*, similar events can be detected. Figure 7.8 shows that during the summer of 2012, two events are detected because of similar visual concept evolving patterns. In fact, these two events are 2012 Summer Olympics in July and August and Paralympic Games in September in London.

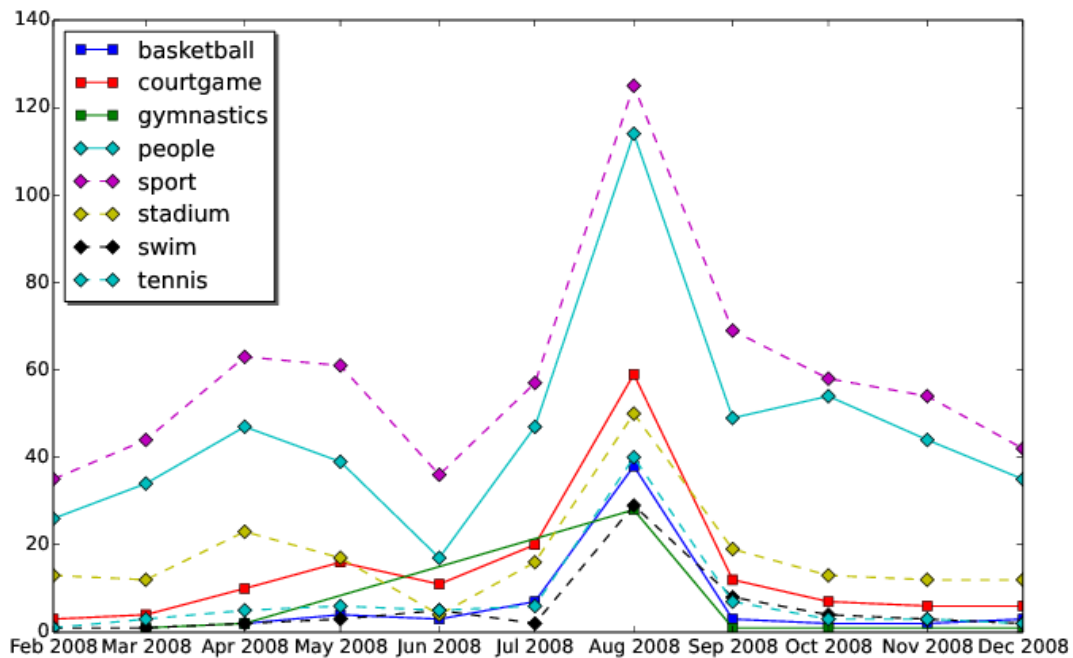


Figure 7.7: Evolving concepts in Beijing in the year of 2008

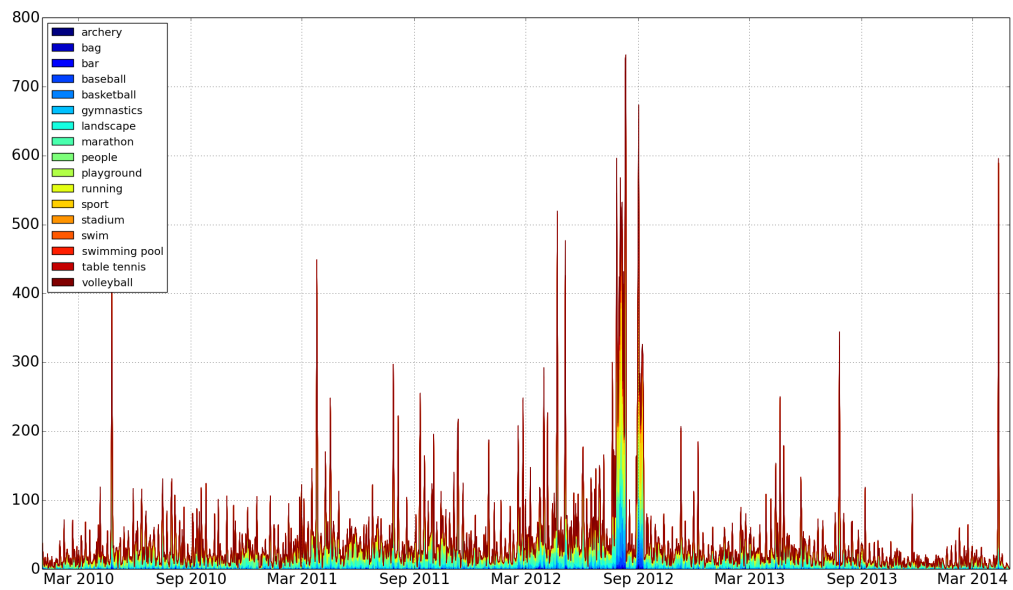


Figure 7.8: Micro-reports of Flickr data

Chapter 8

EventShop Applications

Increasingly, more emerging applications rely on assimilating heterogeneous data streams to solve problems in physical and social situations. In this thesis, EventShop and multimedia micro-reports are developed for building Social Life Networks (SLN) to solve real world problems. The ultimate goal is to make people's lives better by connecting them to the right resources at the right time and at the right place based on a given situation. In the past couple years, several applications have been designed and developed using SLN principle across multiple domains; for example, detecting hurricanes and their migration, identifying demand hot-spots of business products, monitoring flu outbreaks, detecting wildfires, monitoring flood migration, and tracking allergy risks and recommendations. In this chapter, we choose three applications from three different domains to evaluate the applicability of the SLN framework. These applications are a healthcare application for asthma risk management, disaster rescue for flood mitigation, and smart city for trash management in Washington D.C. In some of these applications, EventShop and micro-reports are used for detecting macro situations. For individual users, personal situation information could be obtained from their mobile phones. We have built an interface for EventShop to send personalized actionable information to individual users.

8.1 Healthcare Applications

Traditionally, location and time have not been used as part of clinical information. However, they are actually one of the largest contextual pieces of information. The first effective use of geography in healthcare is believed to have occurred in 1854. When Dr. John Snow plotted the location of cholera deaths on a map in London, he discovered two potential sources of cholera outbreak, namely, two public water pumps [98]. City planners were convinced by his map and removed those pumps and suspected taps. As a result, the outbreak was contained and an epidemic was avoided. In the 2009 TEDMED conference, Bill Davenhall made a strong case that adding environmental data from where the patients lived (such as air quality and types of chemical and particulates found in the areas) to patients' charts could significantly improve patient care. Physicians can create a better diagnosis for each patient since living in some environments may predispose a person to a certain disease. This branch of medicine dealing with the influence of climatic and environmental conditions on health is called "geomedicine".

Beyond discovering the source of disease, the SLN principle is trying to deal with real-time data to recognize evolving environmental situations and personal situations in order to provide personalized recommendations to individuals in real-time. In the healthcare domain, this concept can be used to understand the current epidemic outbreaks or potential risk areas. This can help in preventing serious symptoms in many patients and reduce the number of hospital visits.

8.1.1 Asthma Risk Recommendation

Asthma management is one of our motivating applications for the SLN concept. According to the Centers for Disease Control and Prevention (CDC), an estimated 24 million people,

including more than 6 million children, have asthma [6]. Every year more than 11 million people reported having an asthma attack. This asthma attack can be controlled with an appropriate action plan including medical treatment and control of environmental triggers. A personalized situation-aware recommendation system can aid and improve the lives of many asthma patients. We developed a mobile app for sending out personalized alerts to such users.

The goal of this application is to provide recommendations to an individual based on his/her current situation [109]. Under risky situations, users will receive an alert to ‘stay indoors’ or ‘avoid exerting’, while on favorable days the users will be nudged to ‘go jogging’ and engage in outdoor activities. The macro situation about asthma risk based on several triggers in the environment is derived using EventShop, while the personal situation is identified using a user’s activity level from the FunF framework [25, 8]. FunF is an open-source project for developers to build mobile apps that capture rich personal data and log them securely onto Personal Data Stores [59]. Based on the recent success of rule based systems in research [135], commercial systems (<http://ifttt.com>), as well as situation based systems [99], we decided to use the E-C-A (Event/Situation-Condition-Action) approach for creating alerts and recommendations. Each rule is configured as a standing query so that each time a rule’s conditions are met, the corresponding control action is undertaken. The general rules are set up as follows:

IF (macro condition) and (personal condition) THEN alert.

Macro Asthma Risk Situation

Several factors that cause asthma attacks are related to air quality [129]. One of the significant triggers is air pollution with high levels of pollen, and high concentrations of PM2.5, particles with an aerodynamic diameter < 2.5 microns.

Table 8.1: Data source configuration for asthma risk situation model

Theme	URL/Path	Data Format	Temporal Resolution	Spatial Resolution	Transformation
Pollen Level	http://pollen.com/images/usa_map.gif	Geo Image Stream	daily	0.1 Lat x 0.1 Long	Rasterization
Air Quality	https://files.airnowtech.org/airnow/today/forecast_aqi_yyyymmdd_usa.jpg	Geo Image Stream	daily	0.1 Lat x 0.1 Long	Rasterization
PM2.5 Concentration	https://aqs.epa.gov/api/rawData?user=&pw=&format=DMCSV&m=88501&bdate=yyymmdd&edate=yyymmdd&state=6	CSV Field Stream	hourly	0.1 Lat x 0.1 Long	Spatial Gaussian Process

In our asthma risk model, we assimilate data from three sources: pollen concentration¹, air quality index (AQI) ², and PM2.5 concentration³. Pollen concentration is classified into five levels: low, low-medium, medium, medium-high, and high. EventShop is configured to assign a score to each level from one to five. Both pollen levels and AQI data are available in a map form, while PM2.5 data are available as CSV file of isolated points from measuring stations. To deal with the sparse data of PM2.5, our SLN lab has developed several novel techniques on geospatial interpolations using Spatial Gaussian Process and Graph-based framework which can be found at [130, 131]. For each data source, a higher score means a higher air pollution risk level. The summary of data source configurations used in EventShop is presented in Table 8.1. In this situation model for asthma risk level, values of each data source are normalized into the same scale from 0 to 100 before they are assimilated using EventShop. Finally, the assimilated result is classified into three categories to represent low, medium, and high risk of an asthma attack in the area. The complete model is shown in Figure 8.1.

Personal Situation

Location and activity sensors on a user’s mobile phone are accessible via the FunF framework. When user turns on the application, those sensors are automatically probed every five

¹Pollen Concentrations: <http://pollen.com>

²AirNow: <http://airnow.gov>

³US EPA AirData: <http://www.epa.gov/airquality/airdata>

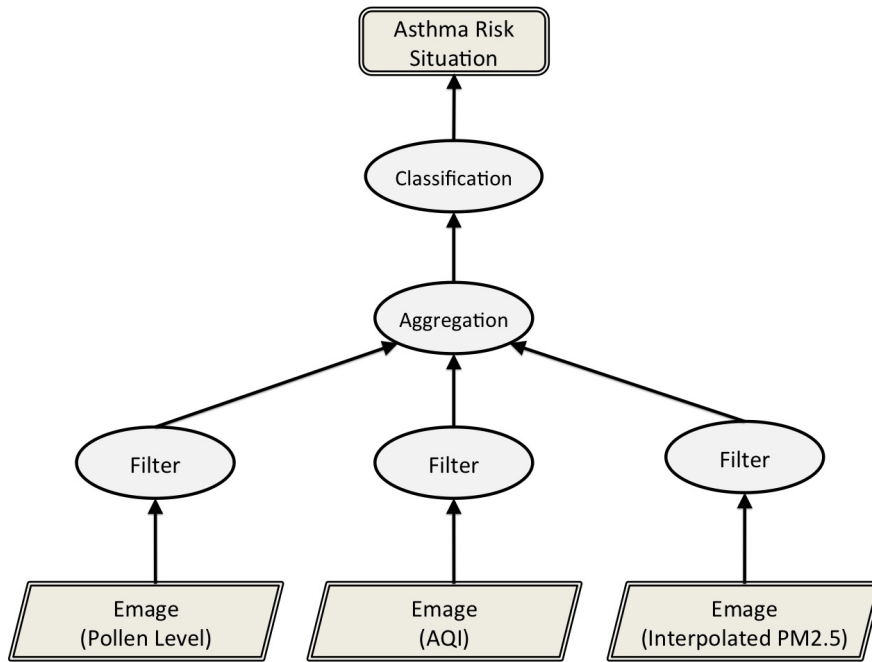


Figure 8.1: Asthma risk situation model

seconds. The user has the option to turn on or turn off the phones as desired. The activity values are classified into two levels: “high” and “low”.

Recommendation Rules

Both macro situations and personal data can be accessed through the EventShop and FunF frameworks. Example rules are presented below.

```

IF asthma_risk == “high” AND user_activity == “high” THEN alert = “stop exerting”
IF asthma_risk == “medium” AND user_activity == “high” THEN alert = “indoor”
IF asthma_risk == “low” AND user_activity == “low” THEN alert = “go jogging”
  
```

Two sample snapshots of the mobile app are shown in Figure 8.2. On the top of the screen, there are control buttons and the recommendation message is shown with highlighted color. On the bottom, the values from the macro risk situations and personal situations are pre-

sented. Multiple users (collaborators) have downloaded the app and tested it out across different geo-locations (different states in the USA) and different user activity levels. After some refinements, it is possible to publicly launch this application soon.

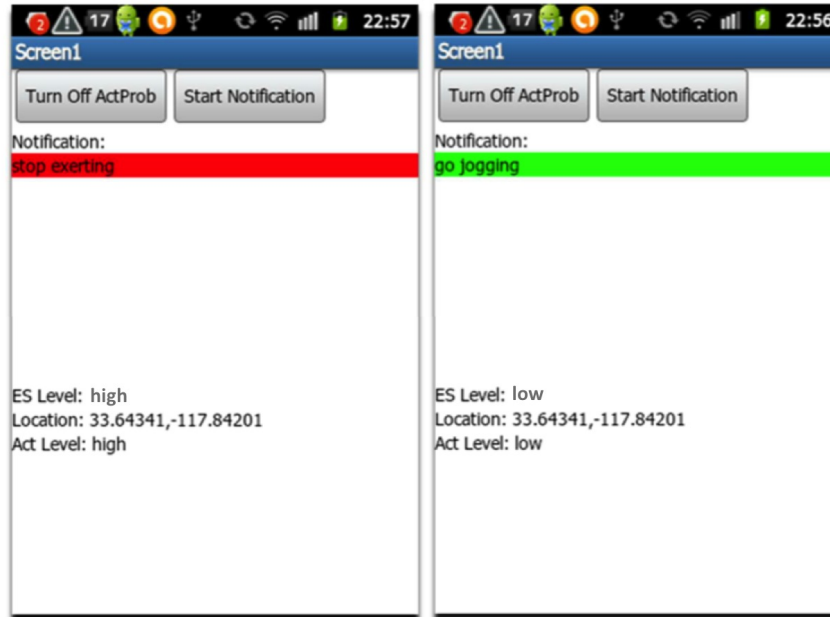


Figure 8.2: Snapshots of the asthma risk recommendation app

8.2 Disaster Management Applications

The term “Disaster Management” is commonly used to represent different phases of a disaster: planning, response, mitigation, and recovery. These phases were identified based on different operations that needed to be considered when dealing with disasters [35, 62, 53]. Given the state of current technology, a disaster management system may comprise three stages as shown in Figure 8.3. In the following, we discuss these stages along with the role of data and information in each of these.

- **Planning:** Planning is in anticipation of a disaster. It includes preparations made to stop disasters from happening, and once to save lives and to help response and rescue



Figure 8.3: Three stages in disaster management. Planning and recovery have different levels of urgency than the response stage.

when it happens. For example, disaster evacuation plans are made and life supply necessities such as food and water are stocked. An important part of the planning is to send proper guidance to people during the disaster response stage. This guidance must be prepared so that it is delivered to people using right media during any situation. During floods, the system should warn people in advance using their phones as well as public broadcast systems. All people still in the area, based on their location from their devices, should be given information about the coming emergency and what they should do.

- **Response:** Response takes place during a disaster and includes actions taken to save lives and prevent further damages in disasters. It puts prepared plans into action. For example, in a flood, one of the responses of citizens is to seek a shelter which has been planned by organizations and governments. Life saving responses usually require timely action. People should have access to the latest information in their vicinity. The best mechanism today is a dashboard on their smart phone that gives flood situation updates, roads, shelters, food availability, and even their friend and family situation. All this information should be prepared using various multimodal data sources ranging from atmospheric and environmental data to people's reports. This may be one of the

most challenging area for multimedia where the semantics of different data streams, multi-granularity, and uncertainty of data must be overcome in order to build current and predictive situation maps.

- **Recovery:** Recovery includes actions for normal or safe situations after disasters. It can go on for months until public facilities such as roads, schools, and hospitals can function well. Recovery of the human health condition may take even longer. During this phase, the system has to assimilate information from both sensors and people about the current state of infrastructure that needs to rebuild or is being rebuilt. More importantly, information about the disaster's material consequences as well as emotional impact on people needs to be managed properly. This will also involve gathering information from various sensors, databases, and people, and build current recovery situation maps that could be used for planning and prioritizing recovery efforts.

In general, sensor networks, camera networks, and IoTs may be used for collecting data, monitoring, and predicting natural disasters such as earthquakes and hurricanes. Data is continuously used by geo-spatial models to predict the times and locations of happenings or movements of disasters. There have been lots of studies focused on developing more accurate models, and lots of efforts in building sensor networks for data collection [84]. The damage of disasters depends on the distance from the disaster and the potential area. In order to estimate the severity of the damage, sensor data from multiple domain-specific sensors are required. To estimate the damage accurately, past data needs to be efficiently stored, mined, and studied.

8.2.1 Emergency Response for Thailand Flood

The response to disasters is meant to help people get resources that will help them get to a safe place or to improve their situation to become safer. As an example, in *Response*,

real-time and prompt decisions like how to rescue a flood victim are crucial. When a flood is imminent, people in the affected area panic about what action they should take. Should they stay home or evacuate to a shelter? In the past, this information was usually broadcast to the whole community, but now we can do better and provide action instructions to an individual or a small group of people using their smartphones. To help people, data from physical sensors (like flood levels monitors), and from governmental or other sources (like available shelters) may be obtained and used. In cases where the above data sources are not available, people in the flood area themselves can still provide more accurate and rich information including texts, photos, and videos for the benefit of individuals and governments. Crowdsourcing on disaster assistance and crisis management has been widely studied and proven to be the first line of information for people to get the latest status of any emergency situation [69]. However, most of these platforms were limited to only visualizing a map of the situations.

Unlike the platform mentioned above, the goal of the SLN platform is not only to recognize the current emergency situation like flood severity, but also to provide actionable information such as the location of the nearest accessible shelter to end users. We evaluate this SLN principle in a real flood situation. In 2011, there was a catastrophic event in Thailand, the author's home country. Several major cities, including the capital city of Thailand, were flooded. During flood situations, it is very important to understand and enhance the ways in which the dynamic aspects of emergencies affect citizens. Organizations should make quick plans for evacuations and providing shelters. We decided to use the EventShop system to manage this emergency situation. Within a day, we began to collect data, recognize flood situations, and provide actionable information to citizens who were in affected areas [108]. We explain more detail in the following section.

Macro Flood Risk Situation

To build this flood management application using EventShop, we first searched for any public

Table 8.2: Data source configuration for Thai flood situation model

Theme	URL/Path	Data Format	Temporal Resolution	Spatial Resolution	Transformation
Flood Affected Area	http://www.thaiflood.com/floodmap	KML Polygon Stream	6 hours	0.01 Lat x 0.01 Long	Rasterization
Shelter Location	http://shelter.thaiflood.com/webservice/request.kml	KML Point Stream	6 hours	0.01 Lat x 0.01 Long	2D Convolution (Gaussian Kernel)
Thai Flood Tweets	Twitter Search API	Text Stream	6 hours	0.01 Lat x 0.01 Long	Spatial Count Aggregation

data sources related to the Thai flood from both professional and social media sources. The former sources included flood affected areas across Thailand⁴, Bangkok risk areas and water barriers, shelters’ locations and availability⁵, parking areas, satellite images, and flood extents. On the other hand, citizens reported their concern via text messaging and uploaded photos or video through a variety of social media sources. Since the multimedia micro-reports component (mentioned in the previous chapter) was not yet available at that time, tweets from central area of Thailand both in Thai and English that were associated with specific hashtags such as “#น้ำท่วม”, “#ThaiFlood”, and “#Flood” across the center part of Thailand were continuously harvested. The summary of data source configurations used in EventShop is shown in Table 8.2.

The macro flood risk situation is derived from the combination of two data sources, including water level measurements in flood affected areas and access to a nearby flood rescue shelter. Both data sources are continuously collected every few hours in the KML format, an international standard format used to display geographic data on Google maps. The flood risk level is higher where a water level is high but the shelter sufficiency is low. Therefore, we first normalized two data sources into the range $[0, 255]$ and then we subtract the shelter Emage from the water level Emage. Finally, we classify the resulting Emage into three categories using the thresholding method with ranges $[0,70)$, $[70, 140)$, and $[140, 255)$. The complete model is shown in Figure 8.4. The flood risk level Emage is shown on the left side of Figure

⁴www.thaiflood.com/floodmap

⁵www.shelters.thaiflood.com

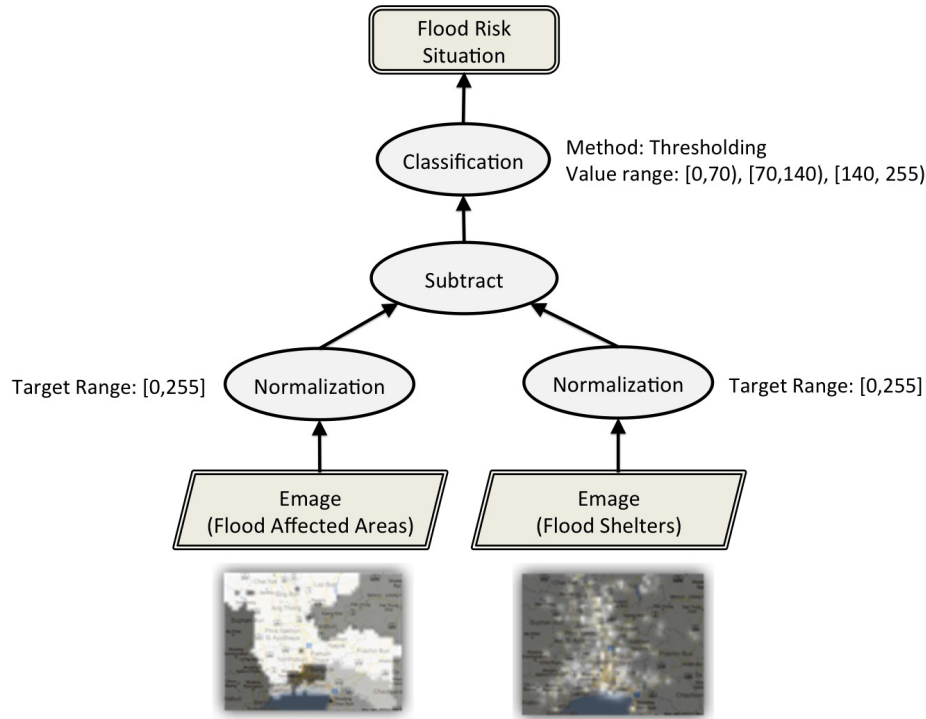


Figure 8.4: Flood risk situation model

8.5. The red, yellow, and green colors represents high, medium, and low flood risk levels respectively.

Personalized Recommendation

This project was conducted five years ago. The multimedia micro-reports did not exist. To reach out to the large number of people in Thailand, we decided to use Twitter since it was the most popular social media at the time. We collected username and user's location from tweets related to Thai flood as mentioned above. Then, for those people who tweeted about flood from the dangerous zones, the EventShop system automatically sent tweets to them and directed them to the nearest shelter in the safe areas. Some of the tweets were being re-tweeted by tweet receivers showing that they received and possibly found them useful. The Twitter account was @SocLifeNetworks. Some examples of tweets are shown on the right side of Figure 8.5.

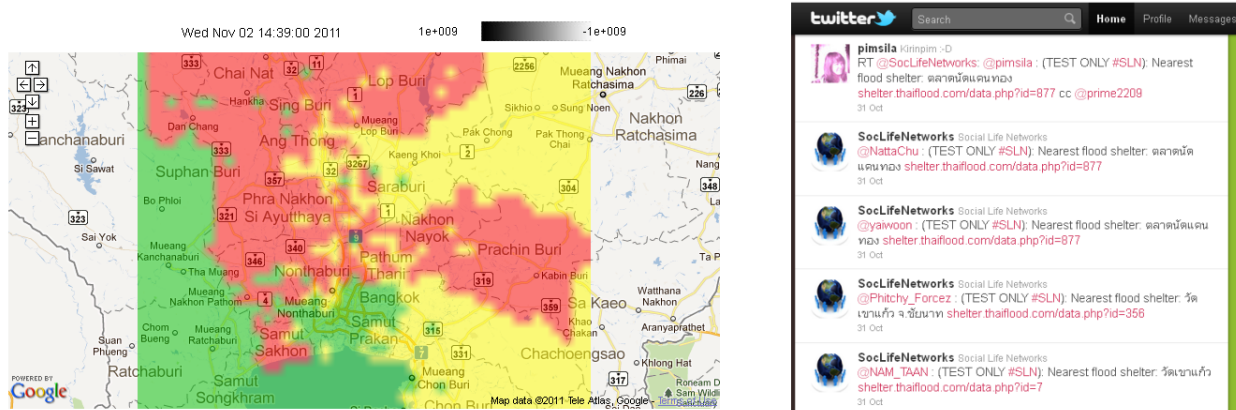


Figure 8.5: Emage result and alert tweets sent from EventShop

Micro-reports and Dashboard

Now, advanced technologies may bring many new opportunities such as visual concepts detection using deep learning technologies in micro-reports for much better situation assessment and communication. Research challenges in developing such a system are discussed in [132]. The following is our vision for building the next generation of the application for managing flood situations. We design and implement a mobile app that allows users to report current flood situations, request help, and provide information about resources' availability as shown in Figure 8.6. Reporters can effortlessly submit their reports using predefined categories including severity of flood situation classification, requesting for help, and providing help to a community.

Figure 8.7(a) shows the mobile dashboard with up-to-date information such as a flood situation forecast, a current situation around a user, and a personalized actionable recommendation. This dashboard acts as the first line of communication between citizens and authorized organizations. To forecast the flood situation, the water level flood model is created based on the historical flood events. We use data from water sensors, precipitation, and satellite images to create the flood model. Then, we use real-time flood levels detected by EventShop to adjust the model and improve the accuracy of the flood prediction model.

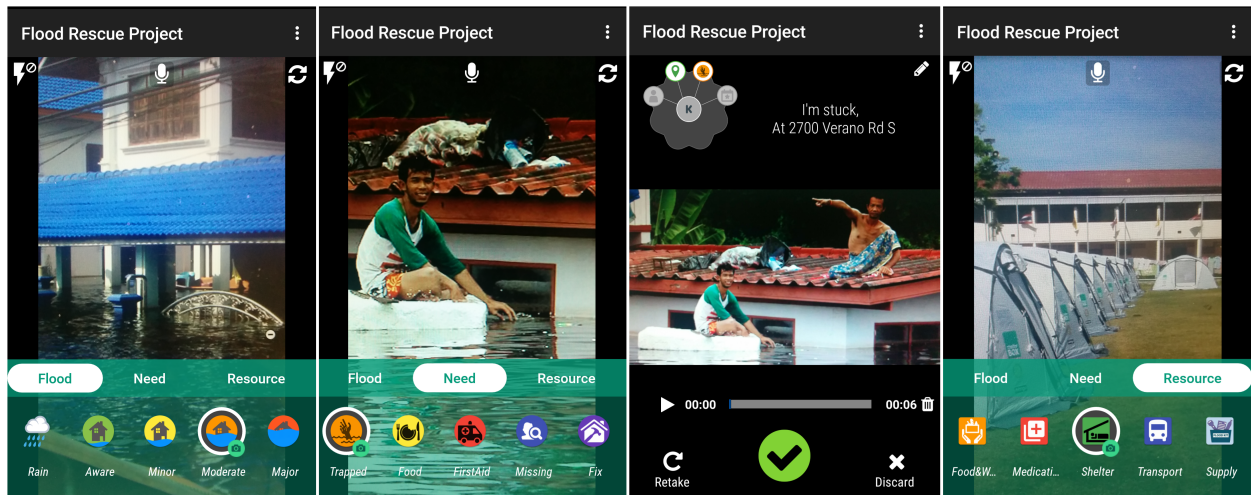
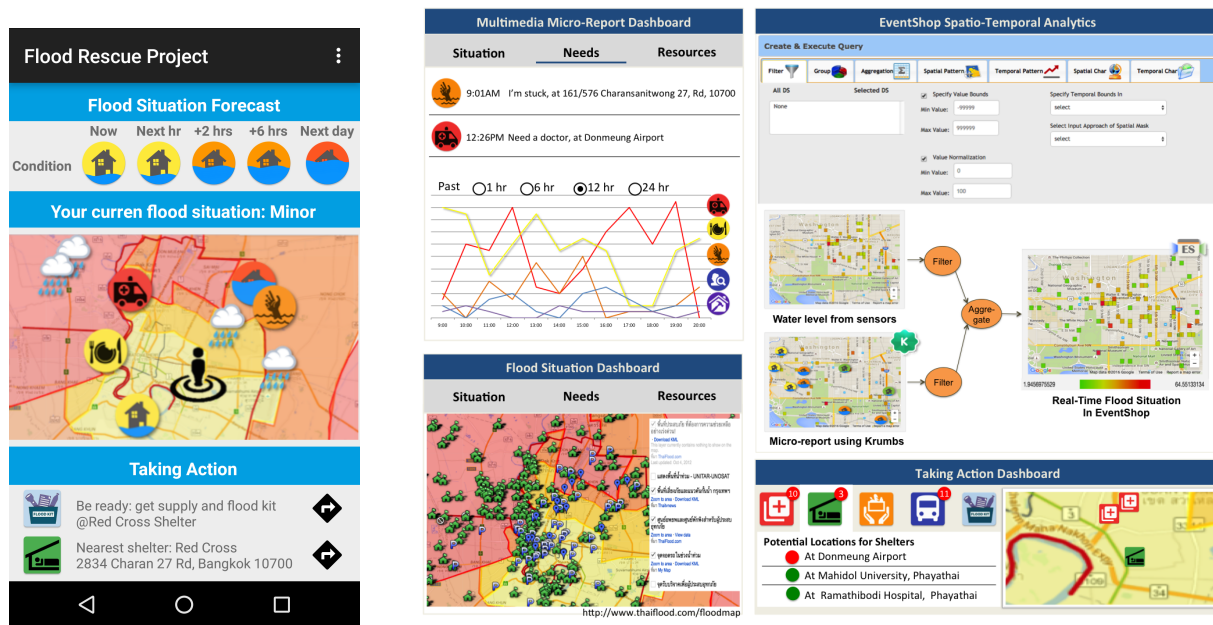


Figure 8.6: Flood multimedia micro-reports using Krumbs intents



(a) Managing flood situation mobile app

(b) Flood Situation Dashboard

Figure 8.7: Dashboard for citizens and city planners

The first goal is to identify affected areas where people need help the most. Furthermore, we can use this information to find optimal shelter locations during evolving flood situations. We propose a city planners' dashboard in Figure 8.7(b). The multimedia micro-reports from our mobile app are located on the top left with three tabs that corresponding to the three categories of reports. All reports are shown in the list and line chart over time. In the

bottom left, data from authorized resources are represented in the map view. On the top right, EventShop provides an experimental environment where analysts can plug in different types of operators onto any data sources to create a complex situation recognition model. For example, water physical sensors and flood situations from multimedia micro-reports are aggregated to create a finer and more complete water situation. The bottom right is where need-resource matching takes place.

8.3 Smart Community and Smart City Applications

The goal of smart Cities is to deploy cyber-physical technologies to improve the quality of life for their citizens. The technical challenge in achieving this goal is to provide **efficient, effective and timely services and resources to address their needs in the context of the situation**. Current technological solutions are quite limited in their situational awareness, or social connectivity. Our initial explorations in creating the concept of Social Life Networks (SLN) [83] was to extend the power of social networks by dynamically monitoring and detecting an individuals situation [75, 83], their surrounding context, and a combination of the two to compute and connect an individual's needs to resources. The smart cities challenge is the 'closing of the loop' on larger scales across services and needs as close to real time as possible. This is an application of Cybernetic design principles [146] to Cyber-Physical-Social systems.

8.3.1 Trash Bins Management

For this application, we ran a pilot study with the Downtown DC Business Improvement District (BID). They have managed a 138 block area in the heart of Washington DC since 1998. As a result of investment and development, downtown DC has become an active

18-hour destination that has seen dramatic changes such as a 400% increase in residential population and the densification of the hospitality industry. These changes require the BID to do an examination of existing procedures in its core operations to determine how to make them more efficient.

This demonstration aims to identify improvements that can be done in public space bin placement, the routing and utilization of personnel and equipment, and in using sensors to determine items of service based upon real-time citizens' needs. The BID Operations team, known as SAMs (Safety/Hospitality and Maintenance), are a highly visible workforce that service over 800 waste bins and nearly 400 recycling bins multiple times daily, while also removing litter from the streets and sidewalks. The SAMS have the ability to perform these services and to assure the public that the area is well maintained on a timely basis.

Macro Trash Fill Level Situation

In today's data rich environment, lots of data is readily available from many open sources, proprietary sources, IoTs, and databases. Increasingly, participatory citizen sensing and crowdsourcing are playing more important roles in understanding current trends and evolving situations. Non-people sources are combined with people sources by weighting them according to the situational context.

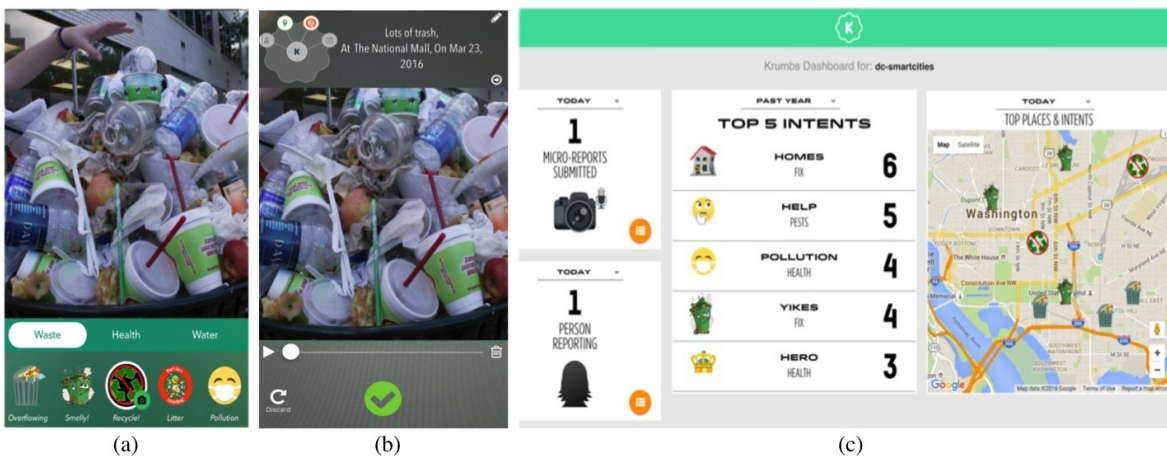


Figure 8.8: (a) Krumbs Intent capture, (b) submitting report, and (c) Krumbs dashboard

Table 8.3: Data source configuration for trash fill level situation model in EventShop

Theme	URL/Path	Data Format	Temporal Resolution	Spatial Resolution	Transformation
Trash Bins Location	http://www.arcgis.com/home/webmap/viewer.html?webmap=14f3eb24f81b4fbb86544e548876a6e6&extent=-77.0368,38.8941,-77.0145,38.9056	Point JSON	30 mins	0.01 Lat x 0.01 Long	Spatial Average Aggregation
Trash Fill Sensors	http://www.arcgis.com/home/webmap/viewer.html?webmap=14f3eb24f81b4fbb86544e548876a6e6&extent=-77.0368,38.8941,-77.0145,38.9056	Point JSON	30 mins	0.01 Lat x 0.01 Long	Spatial Average Aggregation
Trash Micro-Reports	Simulated Data	Media JSON Stream	30 mins	0.01 Lat x 0.01 Long	Spatial Count Aggregation
Events Data	Simulated Data	Point CSV Fields	30 mins	0.01 Lat x 0.01 Long	Spatial Average Aggregation

In this application, we assimilate data sources from both the IoT sensors and user generated content to provide a more complete trash fill level situation. The summary of data source configuration in EventShop is show in Table 8.3. The IoT sensors in trash bins from Enevo provide waste fill levels in 47 commercial sectors across the downtown area. The data include the fill level percentage and the weight of the waste inside the bins. On the other hand, we explored use of **micro-reports** as compared to **micro-blogs** for radical improvement in the quality of participatory sensing data. A mobile app is built from Krumbs SDK to help citizens report trash situations. Citizens are reporters. A report is represented in media-JSON format. Figure 8.8 shows a scenario in capturing and reporting trash levels from a reporter. Figure 8.8(a) shows the Intent capture screen of Krumbs, Figure 8.8(b) shows the report captured and uploaded to the server, and Figure 8.8(c) shows the dashboard that displays all citizens’ reports in various detail as they are received. The prototype of the Krumbs app is already launched. The pilot group from the SAMS teams has been using the application since June 2016.

To recognize the state of trash bins at important locations, sensor signals will be received from those as shown in Figure 8.9(a). Citizen reporting via Krumbs can provide the status of trash bins at these and other locations, as shown in Figure 8.9(b). Humans can capture information that the trash sensor cannot detect, such as bad smell, toxins, and trash overflow

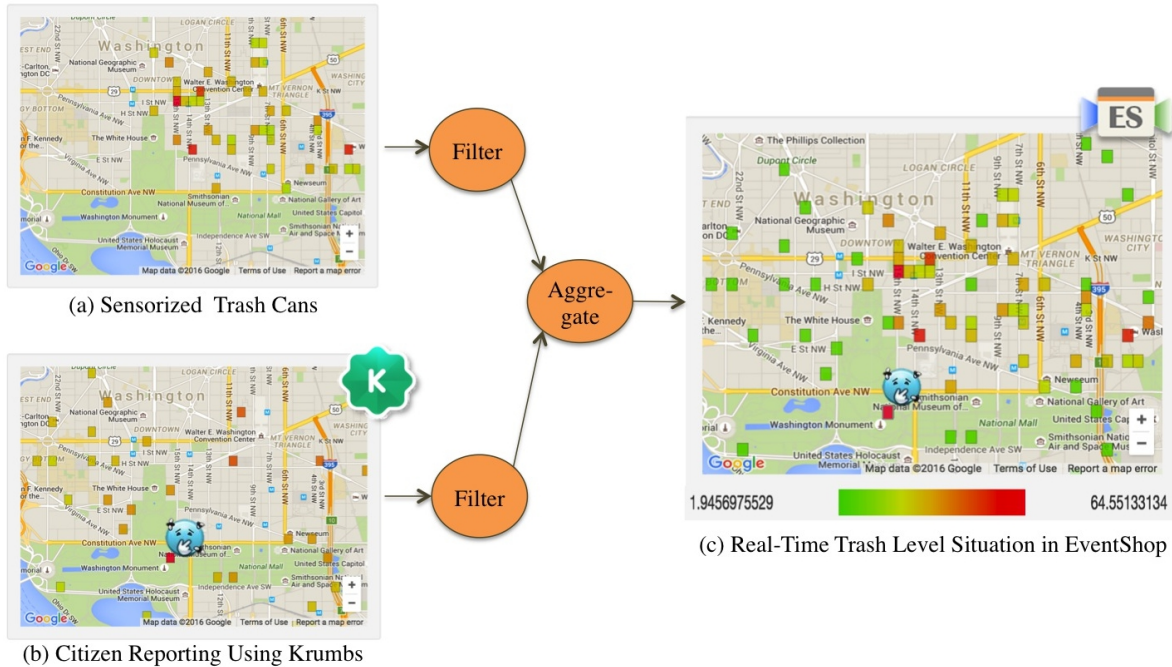


Figure 8.9: Real-time trash situation from sensorized trash cans and citizen reporting using Krumbs

outside the bins. These two data sources could be assimilated to obtain the status of most or all of the trash bins in the city as shown in Figure 8.9(c).

Situation Prediction based on Events History

Situation recognition based only on the latest observations may not be sufficient for timely action in response to critical events. Accurate prediction of the trash-cans' states can lead to better planned actions by the city, especially to avoid health hazards. Typically, predictions are based on limited statistical models and data. Use of large scale data integration, theoretically along with machine learning models can provide more accurate predictions. For smart city applications, we are exploring spatio-temporal prediction analytics, and combining them with real-time object states.

In Figure 8.10, for example, for each upcoming event in the city, we can model the potential volume in a trash bin that will be generated for that event based on its past history and its location. We will create a spatio-temporal model of the trash bin state using a spatio-

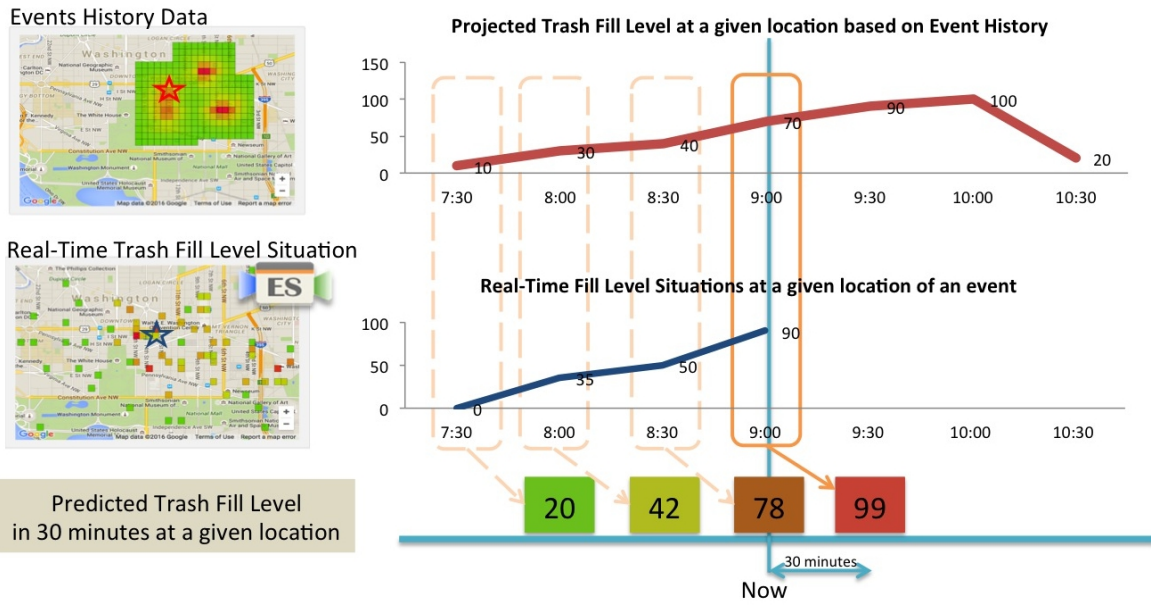


Figure 8.10: Predicting trash situations in 30 minutes at a given event's location by combining a trash prediction model based on events history and a current real-world trash situation.

temporal kernel density estimation. By combining this trash prediction model with the actual current trash situation, we can weight the spatial kernel of the historical data for the current predictive task to accomplish more accurate trash prediction in the near future. This information will significantly help city planners make better decisions, from increasing the number of trash bins to rerouting trash pickup routes.

Chapter 9

Conclusions and Future Research

9.1 Conclusions

We are living in the big data era. Data can be generated by anything and anyone - anytime and anywhere. We believe that most of these data have space, time, and theme value which can be used for observing real-world phenomena. In this dissertation, we introduce a next generation of EventShop, a situation recognition platform. We present a new model and architecture of the system. The EventShop system ingests heterogeneous data streams into the data store, aggregates them to create snapshots of the world, and assimilates those snapshots to detect real-world situations. Once situations are detected, appropriate actions can be taken.

We reviewed the works from various domains related to situation recognition from heterogeneous spatio-temporal data streams. Different types of data sources and data sensors were explored. We examined many existing data stream processing systems. Most of them had four main components in common:- data integration, data store, real-time analytics, and

data visualization. Some systems might have only a few components. We showed that the existing works were not able to efficiently detect real-world situations.

We presented the fundamentals of EventShop including its data model - STT and Eimage, situation modeling operators, and user interface. We discussed its unique features relative to other systems. On the other hand, some limitations were discussed. To deal with those limitations, we proposed an archival system and multi-resolution Eimage stream processing engine.

The archival system was designed and implemented to handle heterogeneity in data streams from disparate data sources. Data were ingested by the data stream ingestor and stored inside the data store. The data stream ingestor accessed the raw data, extracted space-time-theme values, transformed them into a unified format, STT, and finally aggregated STTs to create Eimages. Both raw data and STT data were archived in the data store for future historical data analysis. A big data management system called AsterixDB was chosen for in this work.

Then multi-resolution stream processing engine was required in order to deal with different spatio-temporal granularities of data streams. We discussed some error estimation methods when converting Eimage from its original resolution to a desired resolution. We also proposed a new density-grid based clustering algorithm on STT data streams. We tested it with 100 millions photos from Flickr and showed that it could detect city flood situations and sport events like the Olympic Games.

In many situations like disasters, human are the best sensors in observing and reporting current situations. However, report mechanisms were limited to micro-blogs such as Twitter. Micro-reports were introduced to replace noisy and subjective micro-blogs. Micro-reports were compelling, spontaneous, universal, and objective. They allowed reporters to report facts about events separated from their opinions.

The EventShop system was first developed in 2011 in an academic lab at the University of California, Irvine. Two years later, with support from tech companies, EventShop became an open-source platform. Since then, we have provided many tutorial sessions on Situation Recognition from multimodal data using EventShop in multimedia conferences [123, 124, 125]. EventShop has been used to develop situation-aware applications in various domains to help people across the globe. Some significant applications include asthma risk management in USA, flood migration in Thailand, and trash bins in Downtown DC, USA.

9.2 Future Research

We have made a significant improvement in EventShop. It has already been used by many organizations from both academia and industry. Some future research challenges still remain. These include achieving scalability, enriching advanced operators, and improving usability and user experience.

9.2.1 Scalability

Currently the EventShop system performs on a single node machine with the possibility of distribution on multiple nodes. The following aspects should be explored to solve some scalability issues.

- **Load Shedding:** In high rate data streams, data input rates can exceed the system processing capacity. The system will become overloaded and latency will deteriorate. In that case, proper load shedding techniques are required in order to drop the data points that have the least impact on the final results. Some work has been done on tuple-based data streams [134, 47, 36], but not on spatio-temporal data streams.

- **Data Indexing:** To efficiently query historical data in the data store, indexes are used to quickly locate data instances without having to search the entire dataset. AsterixDB provides three types of index: B+ trees, R trees, and text indexes. These indexes are implemented with log-structured merge (LSM) tree data structure [29]. However, not all indexes are suitable for spatio-temporal data types. Further research is needed to find the best index for STT and Emage data type.

9.2.2 Advanced Operators

In this work, we introduce two new operators for interpolating and clustering STT data at different grid resolutions. Since EventShop is an open-source platform, the set of operators can be extendable by anyone. Application developers with a computer science background can easily add new operators to support their domain specific applications. As the number of users increases, the operators in EventShop will be enriched and advanced. The following examples are some more sophisticated operators that could be implemented by researchers.

- **Geospatial Interpolation:** An interpolation process is a crucial task in spatial data analysis. Plenty of spatial interpolation methods have been presented in many studies, such as inverse distance weighting, nearest neighbor, spline, and Kriging. To improve the interpolation accuracy on unsampled locations, Tang [130, 133] proposed a novel method using spectral analysis to generate features at higher spatial resolution. The Spatial-Gaussian-Process based statistical operator is implemented and can be integrated into the EventShop framework.
- **Situation Estimation and Prediction:** Recognition of situations in real-time is good, but it is much better if the situations are predictable. One example is the use of multimodal data and multimedia modeling tools to predict an area that might face future flooding [12]. An accurate prediction of the severity of disastrous flooding will

significantly reduce damage and improve rescue plans. The major problem making prediction difficult is the complexity of situation behaviors. One interesting approach is to use high resolution 3D technologies to simulate and visualize realistic behavior. Spatio-temporal dimensions also have to be considered in the predictive analytics. Rishabh [113] improved the situation prediction results by using different models according to the surrounding contexts, which were defined by users.

- **Error propagation:** In this work, we assume that the data from any data sources are perfect, which is not always true. Many data streams face uncertainty issues from missing values, network failure, and trustworthiness of the sources or sensors. Numerous approaches try to handle this data uncertainty by using probability theory. In addition, some of the original data or attributes can be lost during window-aggregation over unbounded data streams. Lastly, errors during data conversion from the original structure to a desirable structure are unavoidable. The challenge is to measure the error propagation caused by each data processing step, and find the most optimum situation recognition model. We need a new cost evaluation method that will consider both data accuracy and computational cost.

9.2.3 Usability and User Experience

User experience is an important issue. The EventShop toolkit was not only designed for computer science users, but also for any expert who wants to create situation recognition applications. These experts usually have various cultural backgrounds and expertise levels. We received several suggestions from users of EventShop in the past. Here are some areas that could be improved:

- **Data Visualization:** With the current implementation, loading a high resolution Emage takes too much time and resources. Data visualization tools such as d3¹, leaflet², carto³, mapbox⁴, and Google maps⁵ have been rapidly improving over the past five years. They provide highly efficient and interactive maps for large scale data sets. In addition, exploring data and Emages in virtual reality may create multiple opportunities for situation detection.
- **User Interaction:** The work-flow of setting up a situation recognition application is not intuitive with the current UI. Many steps are required to create one situation model but new users may have a difficult time finding where to start. The stages of data source registration, situation model creation, alert configuration, etc., need to be better presented to the users. In addition, a robust debugging environment can help users inspect the intermediate results of each step in the situation model.
- **Portability:** Similar to PSD files in PhotoShop, the ability to export and save the current situation configuration can be very useful for future refinement. This portability also allows users to share the saved model with others or import it on different machines. For example, users can export the model from the development machine and deploy it again on the production machine.

¹<https://d3js.org/>

²<http://leafletjs.com/>

³<https://carto.com/>

⁴<https://www.mapbox.com/>

⁵<https://developers.google.com/maps/>

Bibliography

- [1] 50 billion connected IoT devices by 2020. <http://www.smartgridnews.com/story/50-billion-connected-iot-devices-2020/2015-04-21>. Accessed: 2015-04-21.
- [2] Akka: Build powerful concurrent and distributed applications more easily. <http://akka.io/>. Accessed: 2016-08-07.
- [3] Apache Kafka: a high-throughput distributed messaging system. <http://kafka.apache.org/>. Accessed: 2016-08-07.
- [4] ArcGIS Software. <http://www.arcgis.com/features/>. Accessed: 2016-05-28.
- [5] AsterixDB: more engine less truck. <http://asterixdb.ics.uci.edu/>. Accessed: 2016-08-07.
- [6] Asthma Facts. https://www.epa.gov/sites/production/files/2016-05/documents/asthma_fact_sheet_english_05_2016.pdf. Accessed: 2016-08-07.
- [7] Crowdmap: a tool that allows you to crowdsource information and see it on a map and timeline. <https://crowdmap.com/>. Accessed: 2016-08-07.
- [8] FunF: open sensing framework. <http://funf.media.mit.edu>. Accessed: 2016-08-07.
- [9] GeoChat: an open source, group communications technology. <http://instedd.org/technologies/geochat/>. Accessed: 2016-08-07.
- [10] GRASS GIS. <https://grass.osgeo.org/>. Accessed: 2016-05-28.
- [11] Krumbs: A micro reporting and analytics toolkit for your applications. <https://www.krumbs.net/>. Accessed: 2016-08-07.
- [12] Modelling better flood responses in port phillip bay. <http://www.csiro.au/en/Research/D61/Areas/Data-for-decisions/Disaster-management/Flood-modelling>. Accessed: 2016-07-23.
- [13] MongoDB unleashes the power of software and data for innovators everywhere. <https://www.mongodb.com/>. Accessed: 2016-08-07.

- [14] MySQL The world's most popular open source database. <https://www.mysql.com/>. Accessed: 2016-08-07.
- [15] QGIS: A Free and Open Source Geographic Information System. <http://www.qgis.org/en/site/>. Accessed: 2016-05-28.
- [16] Sahana Foundation: Open Source Disaster Management Software. <https://sahanafoundation.org/>. Accessed: 2016-08-07.
- [17] StreamBase. StreamBase homepage. <http://www.streambase.com/>. Accessed: 2016-08-26.
- [18] The Asterix Query Language, Version 1.0. <https://asterixdb.ics.uci.edu/documentation/aql/manual.html>. Accessed: 2016-08-07.
- [19] This is what happens on the Internet in 60 seconds. <http://www.marketwatch.com/story/one-chart-shows-everything-that-happens-on-the-internet-in-just-one-minute>. Accessed: 2016-04-26.
- [20] TIBCO RENDEZVOUS MESSAGING MIDDLEWARE. <http://www.tibco.com/products/automation/enterprise-messaging/rendezvous>. Accessed: 2016-08-26.
- [21] Tomnod: a service that uses the power of crowdsourcing to scan satellite images. <http://www.tomnod.com/>. Accessed: 2016-08-07.
- [22] Ushahidi: an open source tool for information collection, visualization and interactive mapping. <https://www.ushahidi.com/>. Accessed: 2016-08-07.
- [23] Yahoo Pipes. <http://pipes.yahoo.com/pipes/>. Accessed: 2013-10-01.
- [24] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal The International Journal on Very Large Data Bases*, 12(2):120–139, 2003.
- [25] N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659, 2011.
- [26] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz. Hadoop gis: a high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, 6(11):1009–1020, 2013.
- [27] E. Albek, E. Bax, G. Billock, K. M. Chandy, and I. Swett. An event processing language (epl) for building sense and respond applications. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 136b–136b. IEEE, 2005.

- [28] S. Alsubaiee, Y. Altowim, H. Altwaijry, A. Behm, V. Borkar, Y. Bu, M. Carey, I. Cetindil, M. Cheelangi, K. Faraaz, et al. Asterixdb: A scalable, open source bdms. *Proceedings of the VLDB Endowment*, 7(14):1905–1916, 2014.
- [29] S. Alsubaiee, A. Behm, V. Borkar, Z. Heilbron, Y.-S. Kim, M. J. Carey, M. Dreseler, and C. Li. Storage management in asterixdb. *Proceedings of the VLDB Endowment*, 7(10):841–852, 2014.
- [30] A. Amini and W. Ying. Dengris-stream: A density-grid based clustering algorithm for evolving data streams over sliding window. In *Proc. International Conference on Data Mining and Computer Engineering*, pages 206–210, 2012.
- [31] H. Appelrath, D. Geesen, M. Grawunder, T. Michelsen, D. Nicklas, et al. Odysseus: a highly customizable framework for creating efficient event stream management systems. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 367–368. ACM, 2012.
- [32] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom. Stream: The stanford data stream management system. Technical Report 2004-20, Stanford InfoLab, 2004.
- [33] A. Arasu, S. Babu, and J. Widom. The cql continuous query language: semantic foundations and query execution. *The VLDB JournalThe International Journal on Very Large Data Bases*, 15(2):121–142, 2006.
- [34] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, et al. Web service choreography interface (wsci) 1.0. *Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems*, 2002.
- [35] N. G. Association et al. *Comprehensive emergency management: A governor’s guide*. [Department of Defense], Defense Civil Preparedness Agency, 1979.
- [36] B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 350–361. IEEE, 2004.
- [37] J. Bacon, K. Moody, J. Bates, C. Ma, and A. McNeil. Generic support for distributed applications. *Computer*, 33(3):68–76, 2000.
- [38] L. Barbosa and J. Feng. Robust sentiment detection on twitter from biased and noisy data. In *ICCL*, pages 36–44. Association for Computational Linguistics, 2010.
- [39] A. Belussi, C. Combi, and G. Pozzani. Formal and conceptual modeling of spatio-temporal granularities. In *Proceedings of the 2009 International Database Engineering & Applications Symposium*, pages 275–283. ACM, 2009.
- [40] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

- [41] M. K. Bergman. White paper: the deep web: surfacing hidden value. *Journal of electronic publishing*, 7(1), 2001.
- [42] C. Bettini and R. De Sibi. Symbolic representation of user-defined time granularities. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):53–92, 2000.
- [43] C. Bettini, C. E. Dyreson, W. S. Evans, R. T. Snodgrass, and X. S. Wang. A glossary of time granularity concepts. In *Temporal databases: Research and practice*, pages 406–413. Springer, 1998.
- [44] V. Bhatnagar, S. Kaur, and S. Chakravarthy. Clustering data streams using grid-based synopsis. *Knowledge and information systems*, 41(1):127–152, 2014.
- [45] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
- [46] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams: a new class of data management applications. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 215–226. VLDB Endowment, 2002.
- [47] D. Carney, U. Çetintemel, A. Rasin, S. Zdonik, M. Cherniack, and M. Stonebraker. Operator scheduling in a data stream manager. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 838–849. VLDB Endowment, 2003.
- [48] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Content-based addressing and routing: A general model and its application. Technical report, Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado, 2000.
- [49] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, and M. A. Shah. Telegraphcq: continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 668–668. ACM, 2003.
- [50] S.-F. Chang, D. Ellis, W. Jiang, K. Lee, A. Yanagawa, A. C. Loui, and J. Luo. Large-scale multimodal semantic concept detection for consumer video. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 255–264. ACM, 2007.
- [51] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. Niagaracq: A scalable continuous query system for internet databases. In *ACM SIGMOD Record*, volume 29, pages 379–390. ACM, 2000.
- [52] Y. Chen and L. Tu. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2007.

- [53] T. J. Cova. Gis in emergency management. *Geographical information systems*, 2:845–858, 1999.
- [54] A. A. Cruz, J. Bousquet, and N. Khaltayev. *Global surveillance, prevention and control of chronic respiratory diseases: a comprehensive approach*. World Health Organization, 2007.
- [55] F. Daniel, F. Casati, S. Soi, J. Fox, D. Zancarli, and M.-C. Shan. Hosted universal integration on the web: The mashart platform. In *Service-Oriented Computing*, pages 647–648. Springer, 2009.
- [56] M. Dao, S. Pongpaichet, L. Jalali, K. Kim, R. Jain, and K. Zettsu. A real-time complex event discovery platform for cyber-physical-social systems. In *International Conference on Multimedia Retrieval, ICMR '14, Glasgow, United Kingdom - April 01 - 04, 2014*, page 201, 2014.
- [57] M. Dao, K. Zettsu, S. Pongpaichet, L. Jalali, and R. Jain. Exploring spatio-temporal-theme correlation between physical and social streaming data for event detection and pattern interpretation from heterogeneous sensors. In *2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29 - November 1, 2015*, pages 2690–2699, 2015.
- [58] G. De Francisci Morales and A. Bifet. Samoa: Scalable advanced massive online analysis. *The Journal of Machine Learning Research*, 16(1):149–153, 2015.
- [59] Y.-A. de Montjoye, E. Shmueli, S. S. Wang, and A. S. Pentland. openpds: Protecting the privacy of metadata through safeanswers. *PloS one*, 9(7):e98790, 2014.
- [60] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.
- [61] L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP*, pages 198–206, 2013.
- [62] T. E. Drabek and G. J. Hoetmer. Emergency management: Principles and practice for local government. *INTERNATIONAL CITY MANAGEMENT ASSOCIATION, WASHINGTON, DC(USA). 1990.*, 1990.
- [63] A. Eldawy and M. F. Mokbel. A demonstration of spatialhadoop: an efficient mapreduce framework for spatial data. *Proceedings of the VLDB Endowment*, 6(12):1230–1233, 2013.
- [64] A. Eldawy, M. F. Mokbel, S. Alharthi, A. Alzaidy, K. Tarek, and S. Ghani. Shahed: A mapreduce-based system for querying and visualizing spatio-temporal satellite data. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 1585–1596. IEEE, 2015.
- [65] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

- [66] P. T. Eugster, R. Guerraoui, and C. H. Damm. On objects and events. In *ACM SIGPLAN Notices*, volume 36, pages 254–269. ACM, 2001.
- [67] F. Fabret, H. A. Jacobsen, F. Llibat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *ACM SIGMOD Record*, volume 30, pages 115–126. ACM, 2001.
- [68] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.
- [69] H. Gao, G. Barbier, R. Goolsby, and D. Zeng. Harnessing the crowdsourcing power of social media for disaster relief. Technical report, DTIC Document, 2011.
- [70] J. Gao, J. Li, Z. Zhang, and P.-N. Tan. An incremental data stream clustering algorithm based on dense units detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 420–425. Springer, 2005.
- [71] M. Gao. *Eventshop: an information engine for processing massive heterogeneous spatio-temporal data streams*. PhD thesis, University of California, Irvine, 2012.
- [72] A. Ginige, L. De Silva, T. Ginige, P. Giovanni, A. I. Walisadeera, M. Mathai, J. Goonetillake, G. Wikramanayake, G. Vitiello, M. Sebillio, et al. Towards an agriculture knowledge ecosystem: a social life network for farmers in sri lanka. In *9th Conference of the Asian Federation for Information Technology in Agriculture (AFITA 2014): ICTs for future Economic and Sustainable Agricultural Systems, Perth, Australia*, pages 170–179, 2014.
- [73] M. F. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, 2007.
- [74] M. F. Goodchild and J. A. Glennon. Crowdsourcing geographic information for disaster response: a research frontier. *International Journal of Digital Earth*, 3(3):231–241, 2010.
- [75] A. Gupta and R. Jain. Social Life Networks: A Multimedia Problem? In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 203–212, New York, NY, USA, 2013. ACM.
- [76] A. Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.
- [77] M. Haklay. How good is volunteered geographical information? a comparative study of openstreetmap and ordnance survey datasets. *Environment and planning B: Planning and design*, 37(4):682–703, 2010.
- [78] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

- [79] T. Hengl. Finding the right pixel size. *Computers & Geosciences*, 32(9):1283–1298, 2006.
- [80] G. Hesse and M. Lorenz. Conceptual survey on data stream processing systems. In *Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on*, pages 797–802. IEEE, 2015.
- [81] G. R. Hjaltason and H. Samet. Incremental distance join algorithms for spatial databases. In *ACM SIGMOD Record*, volume 27, pages 237–248. ACM, 1998.
- [82] N. Hochman and L. Manovich. Zooming into an instagram city: Reading the local through social media. *First Monday*, 18(7), 2013.
- [83] R. Jain and D. Sonnen. Social life networks. *IT Professional Magazine*, 13(5):8, 2011.
- [84] S. Jain, R. C. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile Networks and Applications*, 11(3):327–339, 2006.
- [85] L. Jalali, M.-S. Dao, R. Jain, and K. Zettsu. Complex asthma risk factor recognition from heterogeneous data streams. In *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–6, June 2015.
- [86] L. Jalali, D. Huo, H. Oh, M. Tang, S. Pongpaichet, and R. Jain. Personicle: Personal chronicle of life events. In *Workshop on Personal Data Analytics in the Internet of Things (PDA@ IOT) at the 40th International Conference on Very Large Databases (VLDB), Hangzhou, China, 2014*.
- [87] D. Jayanthi and G. Sumathi. A framework for real-time streaming analytics using machine learning approach.
- [88] T. A. Kass-Hout and H. Alhinnawi. Social media in public health. *British Medical Bulletin*, 108(1):5–24, 2013.
- [89] E. Kouloumpis, T. Wilson, and J. D. Moore. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11:538–541, 2011.
- [90] R. J. L. Jalali. Bringing deep causality to multimedia data streams. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 221–230. ACM, 2015.
- [91] P. Lagacherie and A. McBratney. Spatial soil information systems and spatial soil inference systems: perspectives for digital soil mapping. dsm 2004 montpellier 13-17 september 2004, 2005.
- [92] B. Leban, D. McDonald, and D. Forster. A representation for collections of temporal intervals. In *AAAI*, pages 367–371, 1986.
- [93] S. Liao and Y. Bai. A new grid-cell-based method for error evaluation of vector-to-raster conversion. *Computational Geosciences*, 14(4):539–549, 2010.

- [94] C. Lin and Y. He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM, 2009.
- [95] B. Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.
- [96] D. C. Luckham. *Event processing for business: organizing the real-time enterprise*. John Wiley & Sons, 2011.
- [97] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE intelligent systems*, (2):46–53, 2001.
- [98] K. S. McLeod. Our sense of snow: the myth of john snow in medical geography. *Social science & medicine*, 50(7):923–935, 2000.
- [99] M. Montanari, S. Mehrotra, and N. Venkatasubramanian. Architecture for an automatic customized warning system. In *Intelligence and Security Informatics, 2007 IEEE*, pages 32–39. IEEE, 2007.
- [100] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh. Citysense: An urban-scale wireless sensor network and testbed. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 583–588. IEEE, 2008.
- [101] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 170–177. IEEE, 2010.
- [102] P. Ning, X. S. Wang, and S. Jajodia. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):5–38, 2002.
- [103] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen. The information bus: an architecture for extensible distributed systems. In *ACM SIGOPS Operating Systems Review*, volume 27, pages 58–68. ACM, 1994.
- [104] C. Parent, S. Spaccapietra, and E. Zimányi. Spatio-temporal conceptual models: data structures+ space+ time. In *Proceedings of the 7th ACM international symposium on Advances in geographic information systems*, pages 26–33. ACM, 1999.
- [105] M. J. Paul and M. Dredze. You are what you tweet: Analyzing twitter for public health. *ICWSM*, 20:265–272, 2011.
- [106] M. S. Perry. *A framework to support spatial, temporal and thematic analytics over semantic web data*. PhD thesis, Wright State University, 2008.
- [107] D. J. Peuquet. Making space for time: Issues in space-time data representation. *GeoInformatica*, 5(1):11–32, 2001.

- [108] S. Pongpaichet, V. K. Singh, M. Gao, and R. Jain. EventShop: Recognizing Situations in Web Data Streams. In *WWW*, pages 1359–1368, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences.
- [109] S. Pongpaichet, V. K. Singh, R. Jain, and A. P. Pentland. Situation fencing: making geo-fencing personal and dynamic. 2013.
- [110] S. Pongpaichet, M. Tang, L. Jalali, and R. Jain. Observing real-world phenomena through eventweb over space,time and theme. 2nd International Workshop on Building Web Observatories, ACM Web Science 2014, 2014.
- [111] S. Pongpaichet, M. Tang, L. Jalali, and R. Jain. Using photos as micro-reports of events. In *ICMR*. ACM, 2016.
- [112] G. Pozzani and E. Zimányi. Defining spatio-temporal granularities for raster data. In *Data Security and Security Data*, pages 96–107. Springer, 2010.
- [113] I. Rishabh. *Situation estimation and prediction in spatio-temporal data streams*. PhD thesis, University of California, Irvine, 2013.
- [114] C. A. Rueda-Velásquez. *Geospatial Image Stream Processing: Models, techniques, and applications in remote sensing change detection*. ProQuest, 2007.
- [115] H. Saif, Y. He, and H. Alani. Alleviating data sparsity for twitter sentiment analysis. CEUR Workshop Proceedings (CEUR-WS. org), 2012.
- [116] A. Santanche, S. Nath, J. Liu, B. Priyantha, and F. Zhao. Senseweb: Browsing the physical world in real time. *Demo Abstract, ACM/IEEE IPSN06, Nashville, TN*, 2006.
- [117] A. Sheth. Citizen sensing, social signals, and enriching human experience. *IEEE Internet Computing*, 13(4):87, 2009.
- [118] A. Sheth and M. Perry. Traveling the semantic web through space, time, and theme. *Internet Computing, IEEE*, 12(2):81–86, 2008.
- [119] V. K. Singh, M. Gao, and R. Jain. Social Pixels: Genesis and Evaluation. In *ACM MM*, pages 481–490.
- [120] V. K. Singh, M. Gao, and R. Jain. From microblogs to social images: event analytics for situation assessment. In *Proceedings of the international conference on Multimedia information retrieval*, pages 433–436. ACM, 2010.
- [121] V. K. Singh, M. Gao, and R. Jain. Situation Recognition: An Evolving Problem for Heterogeneous Dynamic Big Multimedia Data. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 1209–1218. ACM, 2012.
- [122] V. K. Singh and R. Jain. *Situation Recognition using EventShop*. Springer; Auflage: 1st ed. 2016, 2016.

- [123] V. K. Singh, S. Pongpaichet, and R. Jain. Situation recognition from multimodal data. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 1–2. ACM, 2016.
- [124] V. K. Singh, S. Pongpaichet, and R. Jain. Situation recognition from multimodal data. In *Proceedings of the 2016 IEEE International Conference on Multimedia and Expo*, pages 1–2. ACM, 2016.
- [125] V. K. Singh, S. Pongpaichet, and R. Jain. Situation recognition from multimodal data. In *Proceedings of the 2016 ACM Multimedia*, pages 1–2. ACM, 2016.
- [126] M. B. Srivastava, J. A. Burke, M. Hansen, A. Parker, S. Reddy, T. Schmid, K. Chang, S. Ganerwal, M. Allman, V. Paxson, et al. Network system challenges in selective sharing and verification for personal, social, and urban-scale sensing applications. *Center for Embedded Network Sensing*, 2006.
- [127] M. Stonebraker, U. Çetintemel, and S. Zdonik. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, 34(4):42–47, 2005.
- [128] R. Strom, G. Banavar, T. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward. Gryphon: An information flow based approach to message brokering. *arXiv preprint cs/9810019*, 1998.
- [129] M. Tang, P. Agrawal, and R. Jain. Habits vs environment: What really causes asthma? In *Proceedings of the ACM Web Science Conference*, page 30. ACM, 2015.
- [130] M. Tang, P. Agrawal, S. Pongpaichet, and R. Jain. Geospatial interpolation analytics for data streams in eventshop. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2015.
- [131] M. Tang, P. Agrawal, S. Pongpaichet, and R. Jain. A graph based multimodal geospatial interpolation framework. In *ICME*. IEEE, 2016.
- [132] M. Tang, S. Pongpaichet, and R. Jain. Research challenges in developing multimedia systems for managing emergency situations. In *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 2016.
- [133] M. Tang, X. Wu, P. Agrawal, S. Pongpaichet, and R. Jain. Integration of diverse data sources for spatial pm2.5 data interpolation,. IEEE, 2016.
- [134] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *Proceedings of the 29th international conference on Very large data bases- Volume 29*, pages 309–320. VLDB Endowment, 2003.
- [135] S. Teleke, M. E. Baran, S. Bhattacharya, and A. Q. Huang. Rule-based control of battery energy storage for dispatching intermittent renewable sources. *IEEE Transactions on Sustainable Energy*, 1(3):117–124, 2010.

- [136] D. Terry, D. Goldberg, D. Nichols, and B. Oki. *Continuous queries over append-only databases*, volume 21. ACM, 1992.
- [137] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.
- [138] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *Commun. ACM*, 59(2):64–73, Jan. 2016.
- [139] R. Tinati, X. Wang, T. Tiropanis, and W. Hall. Building a real-time web observatory. *Internet Computing, IEEE*, 19(6):36–45, 2015.
- [140] L. Tu and Y. Chen. Stream data clustering based on grid density and attraction. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(3):12, 2009.
- [141] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.
- [142] X. Wang, X. Zhou, and S. Lu. Spatiotemporal data modelling and management: a survey. In *Technology of Object-Oriented Languages and Systems, 2000. TOOLS-Asia 2000. Proceedings. 36th International Conference on*, pages 202–211. IEEE, 2000.
- [143] Y. Wang. Socializing multimodal sensors for information fusion. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 653–656. ACM, 2015.
- [144] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twiterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.
- [145] U. Westermann and R. Jain. Toward a common event model for multimedia applications. *IEEE MultiMedia*, (1):19–29, 2007.
- [146] N. Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*, volume 25. MIT press, 1961.
- [147] C. F. Yoder. *Astrometric and geodetic properties of Earth and the Solar System*. Wiley Online Library, 1995.
- [148] M. Yuan. Temporal gis and spatio-temporal modeling. In *Proceedings of Third International Conference Workshop on Integrating GIS and Environment Modeling, Santa Fe, NM*, 1996.
- [149] W. Zhang, L. Zhang, Y. Ding, T. Miyaki, D. Gordon, and M. Beigl. Mobile sensing in metropolitan area: Case study in beijing. In *Mobile Sensing Challenges Opportunities and Future Directions, Ubicomp2011 workshop*, 2011.

Appendix A

EventShop Web Application

In addition to the back end stream processing engine, EventShop provides a front end user-friendly GUI (Graphical User Interface) including all spatio-temporal analysis operators mentioned in the previous chapter. This web application has been designed to provide a simple and effective way to interact with a vast amount of data sources collection. While many of the operators and situation models included in the application are self-explanatory or can be learned by trial and error, this chapter serves as a comprehensive companion resource for working with the application.

A.1 Getting Started

The EventShop web application prototype is a web browser based program that delivers the facility to apply data analysis operators on the real-time and historical spatio-temporal data streams. The majority of the process executed by the application is done on the UCI EventShop's server, therefore making it possible to interact with complex processing on nearly any computer with a web browser.

A.1.1 System Requirements

The web application is designed for Mozilla Firefox 5.0 or higher. Additional web browsers may work, but have not been tested and are not supported. The computer should also meet the minimum system requirements provided by the manufacturer of the web browser. For the best performance, it is recommend to have a high-speed internet connection similar to digital subscriber line (DSL), cable, or faster.

A.1.2 Launching EventShop Web Application

The EventShop web application prototype is available online. To get started, please type the following address into your web browser or click on the link here.

<http://eventshop.ics.uci.edu/eventshop>

The “Login” page will be loaded as shown in Figure A.1.

A.1.3 Log In and Sign Up

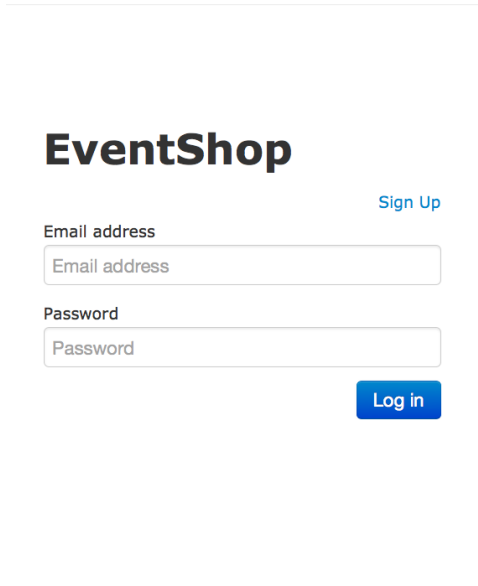
At the Login page (Figure A.1), users who have previously registered for the EventShop web application must log in by:

Entering their **Email Address**

Entering their **Password**

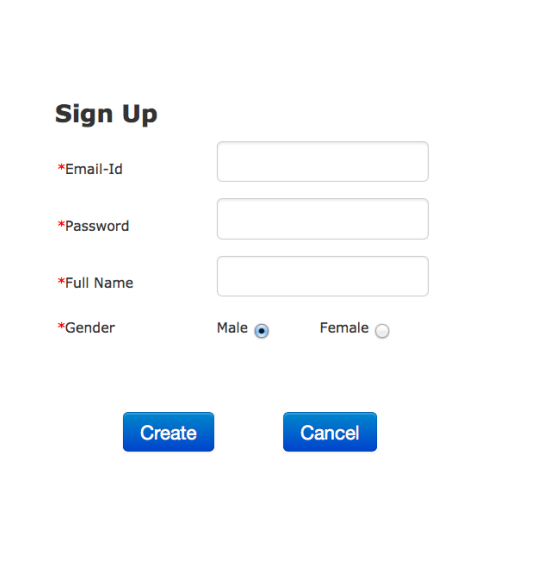
Click **Log In** to advance to the “Home” page and begin using the application.

Users who have not previously registered must click “Sign Up” link to access the “Signup” page as show in Figure A.2. User will be asked to enter or select the following information (all items with a red asterisk (*) are required).



The login page features the 'EventShop' logo at the top left. To the right of the logo is a blue 'Sign Up' link. Below the logo are two input fields: 'Email address' and 'Password'. A blue 'Log in' button is positioned at the bottom right of the form area.

Figure A.1: Login Page



The sign up page is titled 'Sign Up'. It contains four input fields: '*Email-Id', '*Password', '*Full Name', and '*Gender'. The '*Gender' field has two radio buttons, 'Male' (which is selected) and 'Female'. At the bottom of the form are two blue buttons: 'Create' and 'Cancel'.

Figure A.2: Signup Page

Email Address: enter user’s email address.

Password: enter user’s password. Passwords are case sensitive.

Full Name: enter user’s first name and last name.

Gender: select one of the bullet point.

Create: click this button to submit the information. If success, the page will be redirected to the Login page where users can continue to log in.

Cancel: click this button to cancel, and go back to the Login page.

A.1.4 Application Layout

After logging into the EventShop application, the users will advance to the EventShop “Home” page. The layout of this Home page is divided into a series of panels as show in Figure A.3. The basic components are:

a) **Register Data Sources:** to display a current list of data sources registered with the system, to register new data sources into the system, and to control and visualize data sources

DSID	DS Name	Description	Source
350	Pollen	Pollen level across USA	www.pollen.com
357	AQI	Air quality index across USA	www.airnow.gov
361	Tweets_Asthma	Tweets related to asthma and allergy	www.twitter.com

Table A.1: Default Data Source Set

- b) **Create and Execute Query:** to create and register query by applying different operators to the registered data sources
- c) **Query Graph:** to graphically visualize the intermediate query currently being composed by the user
- d) **Registered Queries:** to display a current list of queries registered with the system and to control the query processes
- e) **E-mage:** to see an E-mage of a data source

Each components and its functionality are described in more detail in the sections that follow.

A.2 Register Data Sources Panel

The register data sources panel is in the lower left of the web interface. This panel displays the list of data sources registered with the system in the table, and includes the following information: *data source's ID (DSID)*, *data source's name (DS Name)*, and *status* (“connecting” or “available”) as shown in Figure A.4.

We provide these following three data sources in Table A.1 as a sample set for all the users. Users are not allowed to change these data sources, but they can “view” the available data source E-mage on the screen.

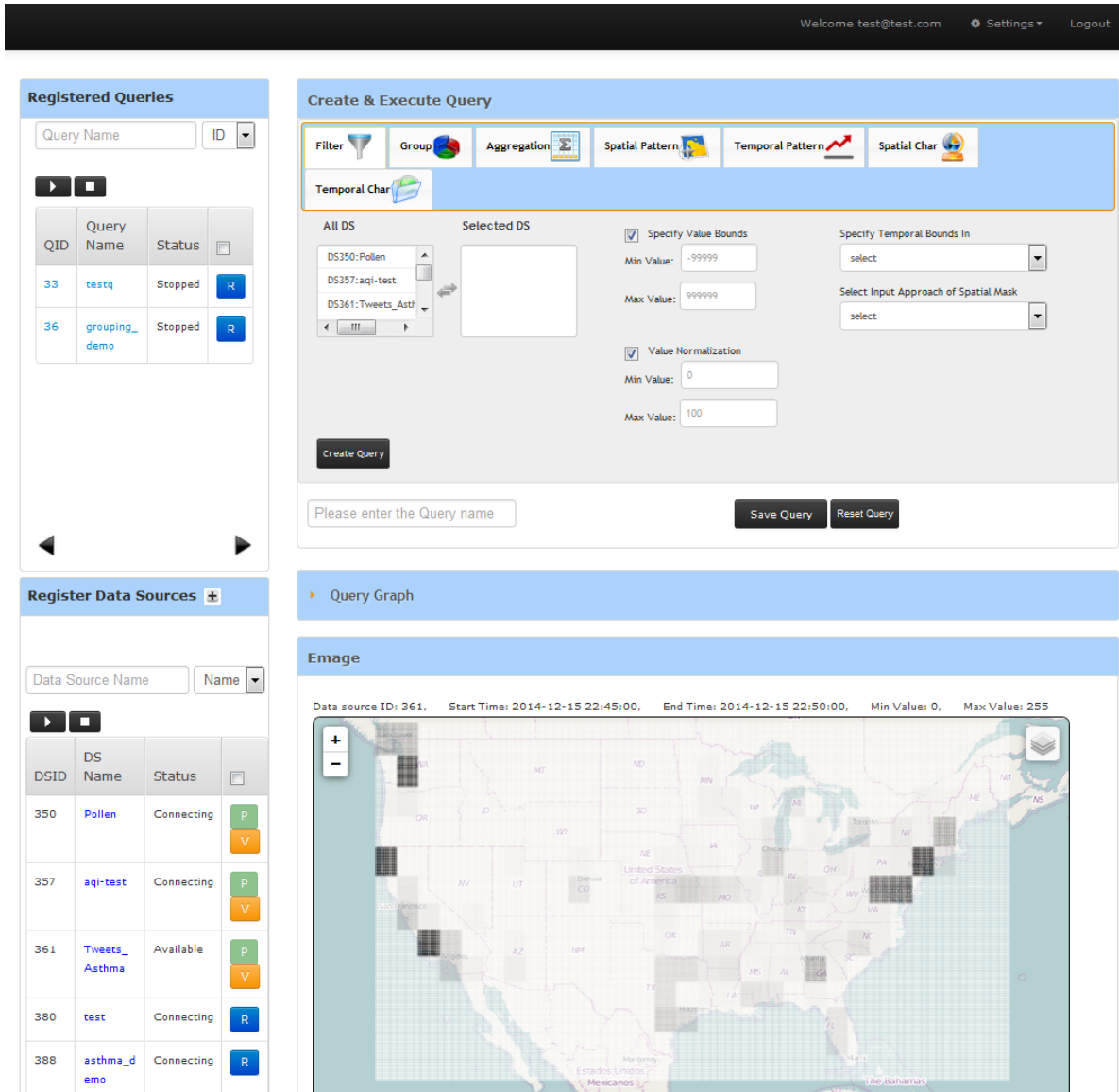


Figure A.3: Snapshot of EventShop Home Page

A.2.1 Controlling and Searching Data Sources

Based on the data source's status, different controller buttons are shown.

- If the status is “connecting” which means the data source has been registered but has not been collecting any data, only the run [R] button will be shown. User can click this button to start collecting data. At this point, based on the time window, it may

DSID	DS Name	Status	
350	Pollen	Connecting	P V
357	aqi-test	Connecting	P V
361	Tweets_ Asthma	Available	P V
380	test	Connecting	R
388	asthma_d emo	Connecting	R

Figure A.4: Registered Data Source Panel

take longer period of time for an E-mage to be ready.

Important! Please click “R” button only once, otherwise multiple data collection process will be created which can cause the system to malfunction.

- If the status is “available” which mean the data source has been registered and collecting data up to date, both pause [P] button and view [V] button will be shown. Users can either pause the collecting process by clicking on [P] button or view the latest E-mage of the selected data source by clicking on [V] button. The result E-mage will be presented in the E-mage Panel on the bottom-right of this application. When the data source collecting process is paused, its status is changed to “connecting” and its E-mage is not available to view.

To search for a particular data source, users can just type the data source name into the text box. Only the data sources that match the user’s input will be shown in the list.

A.2.2 Adding New Data Source

The registered data source panel can be used to add different types of data sources to the system. The users can click on the [+] button to start this process. This leads to a configuration panel (Figure A.5) for the data source including Theme, Name, URL/File Path, and Data Format. We currently support three types of data sources including Data Streams (e.g. Twitter, Flickr, or a simulated data stream), Visual Images (e.g., Pollen maps, weather maps from any URL), and CSV Data Files (comma separated data uploaded by the user). The data in this file can be either a 2D array format with each row corresponding to a row of data in the E-mage or a table-like data format with each row corresponding to one data record/observation which contains multiple fields including latitude, longitude, value and etc.

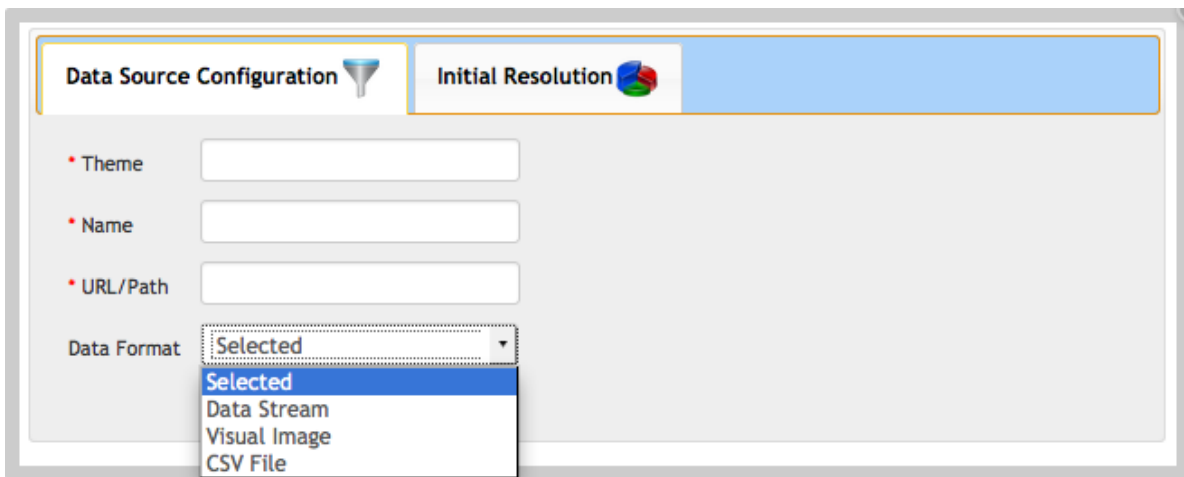


Figure A.5: Data Source Configuration (Step 1)

Selecting any of the three options leads to more configuration options for each of the categories (Figure A.6). The options to be configured for each of the category are as follows:

1. *Data Stream*

Supported: select the supported wrappers from the list.

Data Source Configuration Initial Resolution

- Theme: Asthma
- Name: Twitter-Asthma14
- URL/Path: www.twitter.com
- Data Format: Data Stream
- Bag of Words:
 - Supported: Twitter
 - flue, asthma, allergy

(a) Data Stream

Data Source Configuration Initial Resolution

- Theme: Pollen
- Name: Visual-Pollen
- URL/Path: http://www.pollen.com/imag
- Data Format: Visual Image
- Visual Upload Parameters:
 - Translataion Matrix Path: Browse... translationMatrix.txt
 - Color Matrix Path: Browse... colorMatrix.txt
 - Mask Path: Browse... population.txt
 - Ignore Colors after: 5

(b) Visual Image

Data Source Configuration Initial Resolution

- Theme: asthma
- Name: csv-asthma_demo
- URL/Path: atasource/tweets_asthma.txt
- Data Format: CSV File
- Type of Dataformat: CSVArray
- `{"datasource_type": "grid", "spatial_wrapper": "sum"}`

(c) CSV File - Array

Data Source Configuration Initial Resolution

- Theme: pm25
- Name: csv-pm25
- URL/Path: 004/sln/datasource/pm25.txt
- Data Format: CSV File
- Type of Dataformat: CSVField
- `{"datasource_type": "point", "spatial_wrapper": "sum", "lat_index": 1, "lon_index": 2, "val_index": 5}`

(d) CSV File - Table Field

Figure A.6: Data Source Configuration (Step 2) for Different Data Formats

Bag of Words: enter specific terms to be searched for in the data stream (only for Twitter and Flickr wrapper), or leave it blank for a simulated data stream source.

2. Visual Image

Translation Matrix Path: enter a path to a user created file which contains the

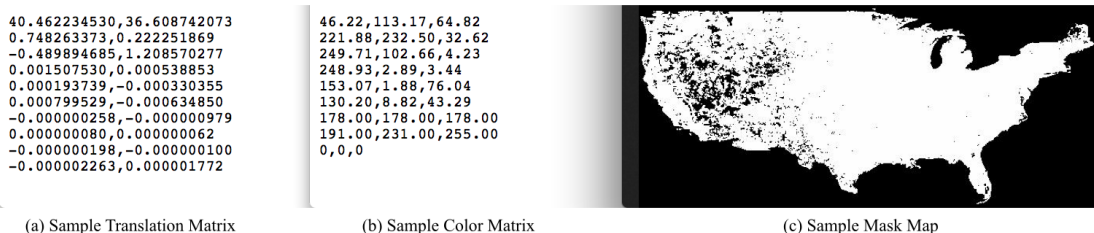


Figure A.7: Example of Visual Image Data Source Parameters

translation matrix. (Translation matrix is a 12X2 matrix which translates the incoming map/ image from its base projection system to Plate Caree or equirectangular projection used by the system. Please refer to http://en.wikipedia.org/wiki/Map_projection, and http://en.wikipedia.org/wiki/Camera_matrix for more details. We provide an executable file to help users generate this matrix). For a sample see Figure A.7

Color Matrix Path: enter a path to a matrix which contains RGB values of samples of different classes which need to be interpreted from the source data. For a sample see Figure A.7

Mask Path: enter an optional mask image to filter input data only to certain regions such as populated mainland USA. For a sample see Figure A.7

Ignore Colors After: enter an optional parameter (index value) to ignore RGB values in the color matrix after a certain index (to get rid of background color, or unnecessary data classes)

3. CSV File

Type of Data Format: select between 2D array or table-like data format.

File Path: upload the data on users local computer for analysis. Data needs to be in comma-separated-values (CSV) format.

Key-Value Pair: enter a key-value data in JSON format to specify the `datasource_type`, `spatial_wrapper`, `lat_index`, `lon_index`, and `value_index`. For CSVArray, the `datasource_type`

is “grid”, and the spatial_wrapper can be one of the “sum”, “min”, “max”, and “average”. The first key means the original structure of the STT Observation, while the later identifies the resolution wrapper function to map the original data resolution to the data source initial resolution (explain in the next step). For CSVField, users have to provide three additional values to identifies the column index of latitude, longitude and value in the file. By default, the datasource_type is “point”, and spatial_wrapper is “sum”.

Parameter	Value
* Time Window(in sec)	300
Synchronize at nth Sec	0
* Unit of Latitude	0.1
* Unit of Longitude	0.1
* Southwest of Latitude	24
* Southwest of Longitude	-125
* Northeast of Latitude	50
* Northeast of Longitude	-66

Figure A.8: Data Source Configuration (Step 3) Defining Initial Resolution Parameters

The users also need to configure the initial and final resolution parameters for the data source. Each configuration (refer Figure A.8) includes the spatio-temporal bounding boxes, and the minimum resolution. The initial resolution defines the incoming rate (and the bounding box) of data to be extracted from the data-source. This data can be translated into a final resolution which needs to be common across different sources being used in any situation query. Note that the default values have been configured to match that of mainland US at

minimum 0.1 lat X 0.1 long resolution, with data being refreshed every 10 seconds. The process of translating the data from the initial resolution to the final resolution is handled by an internal resolution-mapper.

Click “Add” button to register data source or click “Cancel” button to cancel adding data source process

A.2.3 Sample of Data Sources

This section provides an example configuration of the available data sources. User can try to add this following data sources into the system.

Air Quality Index	
Theme	aqi
Name	aqi_demo
URI	http://eventshop.ics.uci.edu/eventshopdata/sampled/aqi_map.jpg
Data Format	Visual Image
Trans Matrix	http://eventshop.ics.uci.edu/eventshopdata/sampled/aqi_trans_mat
Color Matrix	http://eventshop.ics.uci.edu/eventshopdata/sampled/aqi_col_mat
Mask Image	http://eventshop.ics.uci.edu/eventshopdata/sampled/aqi_mask.png
Ignore after	6 (colors)

Tweets about Asthma and Allergy	
Theme	asthma
Name	asthma_demo
URI	http://eventshop.ics.uci.edu/eventshopdata/sampled/tweets_asthma.txt
Data Format	CSV File
Type	CSVArray
Key-Value	{“datasource_type”:“grid”, “spatial_wrapper”:“sum”}

PM2.5 (Particulate Matter 2.5)	
Theme	pm25
Name	pm25_demo
URI	http://eventshop.ics.uci.edu/eventshopdata/sampled/pm25.txt
Data Format	CSV File
Type	CSV Field
Key-Value	{“datasource_type”:“point”, “spatial_wrapper”:“sum”, “lat_index” :1, “lon_index” :2, “val_index” :5}

Interpolated PM2.5 (Particulate Matter 2.5)	
Theme	interpolation_pm25
Name	interpolation_pm25_demo
URI	http://eventshop.ics.uci.edu/eventshopdata/sampled/interpolation.txt
Data Format	CSV File
Type	CSV Field
Key-Value	{“datasource_type”:“point”, “spatial_wrapper”:“sum”, “lat_index” :1, “lon_index” :2, “val_index” :3}

A.3 E-mage Panel

A.3.1 Visualizing Data Source E-mage

The E-mage panel on the bottom right of the EventShop interface displays the data source E-mage over the map as shown in Figure A.9. On the top of this panel, it shows this following information: Data Source ID, Start Time, End Time, Minimum Value, and Maximum Value. User can interactively navigate within the map for example, zoom in, zoom out and pan. The underneath map can be changed by click on the “layer” icon at the top right. User can see the value of each cell in the E-mage by mouse over the cell. The value of that particular cell is shown in the box at the lower left. The legend represents the color and value range is on the bottom right.

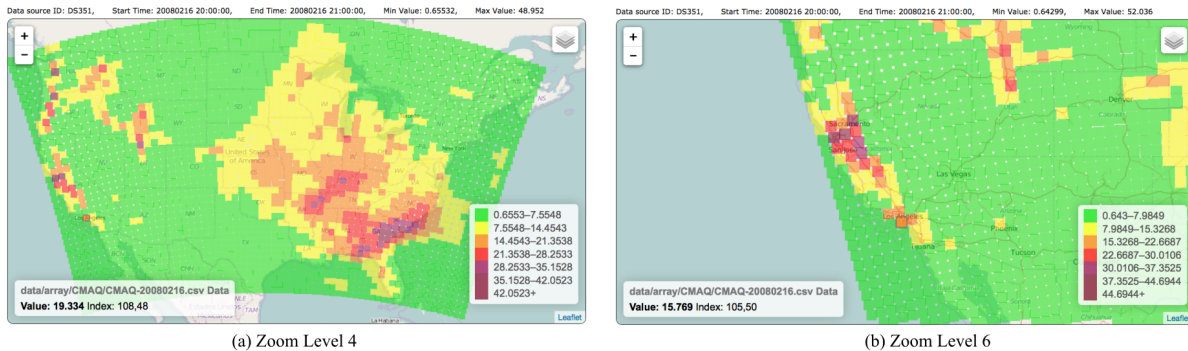


Figure A.9: Data Source E-mage Visualization

A.4 Create and Execute Query Panel

A.4.1 Forming Queries using Operators

The users can form queries by applying any of the 7 categories of operations made available. The set of operators supported includes:

Filter

To filter each E-mage in the incoming data source based on one or combination of the following choices.

All DS and Selected DS: select ONE data source by click on the data source in the All DS box, the selected data source will be moved to the Selected DS box. To remove the selected data source, click on the data source in the Selected DS box. It will be moved back to the All DS box.

Value Bounds: provide a minimum and maximum value bound

Temporal Bounds: select from these two options: relative time interval before now or absolute time interval (Figure A.11)

Spatial Mask: select the input approach of the spatial mask from these three options: select on the map, select by place name or upload a matrix file (Figure A.12)

Value Normalization: choose to normalize values of a result E-mage into a new value range by checking on the Value Normalization check box and providing minimum and maximum value range (Figure A.10).

Create Query: click this button to create an intermediate query

Grouping

Using approaches like Kmeans and Thresholding to segment each E-mage in an E-mage stream into different groups e.g., high, mid, or low swine flu activity. (Figure A.13)

Selected DS: select ONE data source by click on the data source in the All DS box, the selected data source will be moved to the Selected DS box.

Group by: select grouping method from dropdown list. For Kmeans approach, users have to specify the number of groups. For Thesholding approach, users provide the thresholds values for each group.

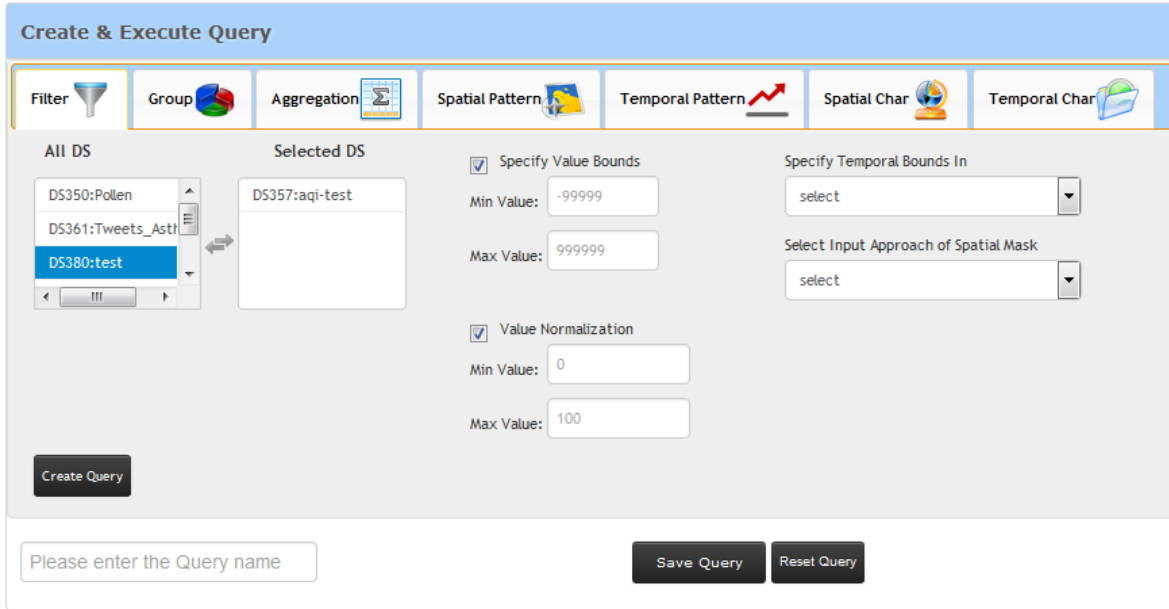


Figure A.10: Filter Operation Configuration

Group Colors: provide a color code for visualizing each group.

Create Query: click this button to create an intermediate query.

Aggregation

Aggregation operator combines E-mages from one or more E-mage streams using options like: Max, Min, Sum, Average, Subtraction, Multiplication, Division, And, Or, Not, Xor, and Convolution (refer Figure A.14). Note that some aggregates, such as *Subtraction* and *Division*, can only be applied between two E-mages. *Not* is only allowed on one E-mage.

Selected DS: select one or more data sources by click on the data source in the All DS box, the selected data source will be moved to the Selected DS box. All selected data sources will be shown in the right box. The value of resulting E-mage can also be normalized.

Aggregation Operator: select aggregation operator from the dropdown list.

Value Normalization: choose to normalize values of a result E-mage into a new value range by checking on the Value Normalization check box and providing minimum and maximum value range.

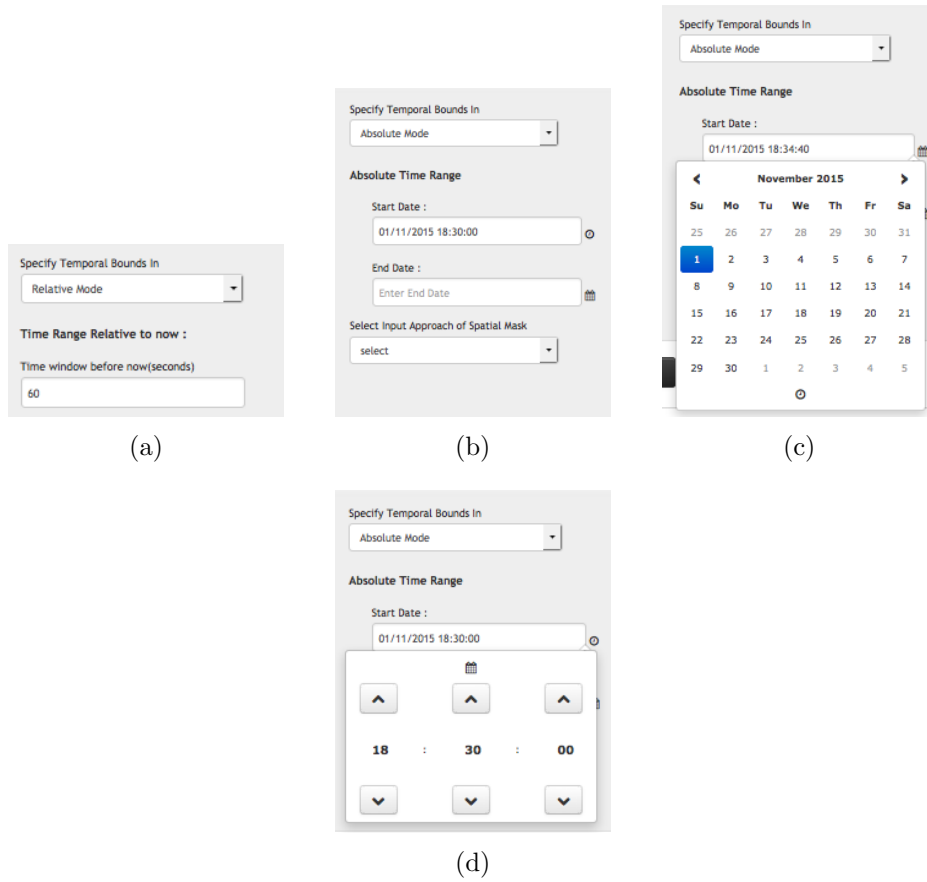


Figure A.11: Filter Temporal Bounds in (a) Relative Mode and (b-d) Absolute Mode

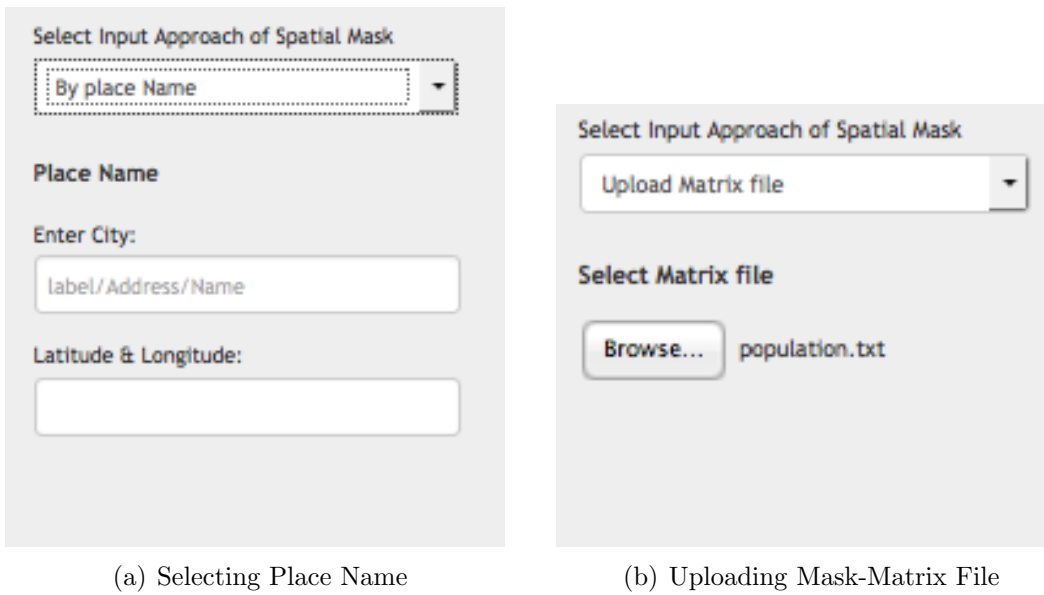


Figure A.12: Filter Spatial Bounds

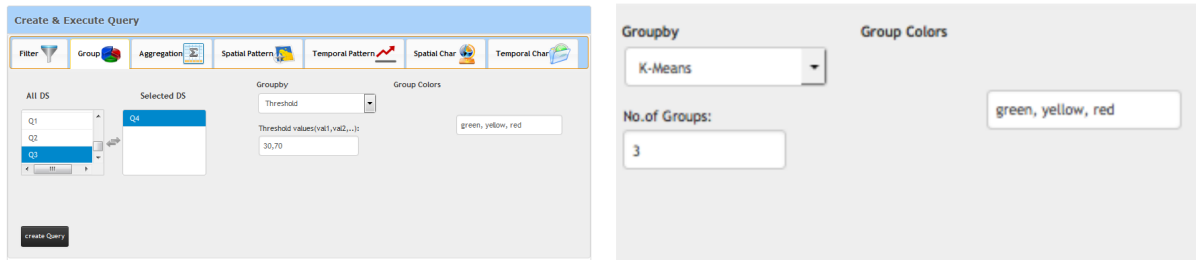


Figure A.13: Grouping Operation Configuration

Create Query: click this button to create an intermediate query

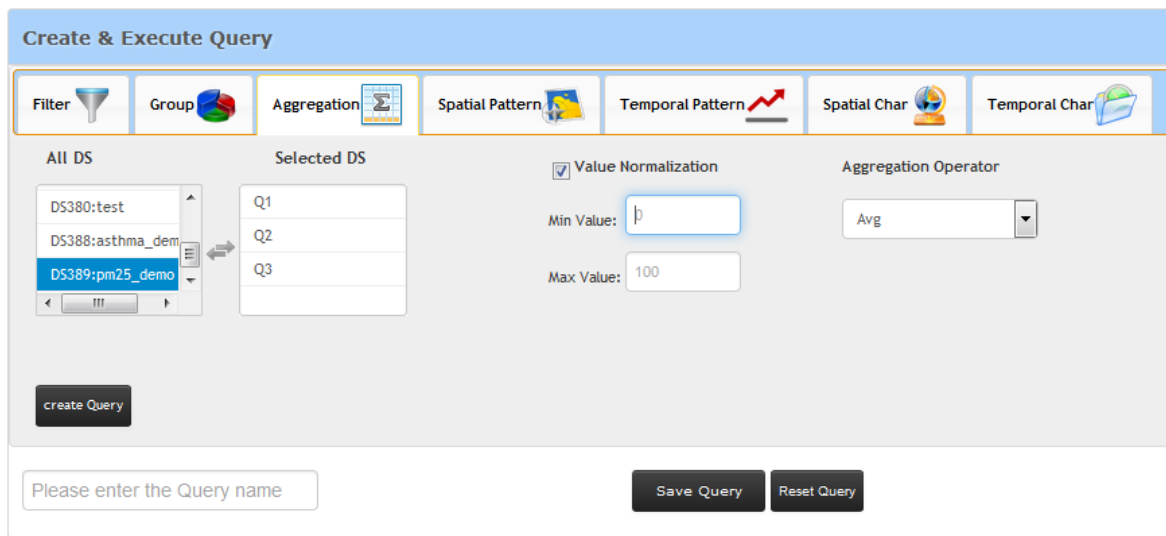


Figure A.14: Aggregation Operation Configuration (for Averaging Values on Three E-mages)

Spatial Characterization

The input of the operator is an E-mage stream. This operator takes a single E-mage stream and computes a spatially relevant property of each E-mage in the E-mage stream. Then generate a *stel stream* as an output. Each stel in the stream stores the spatial coordinate where the measure is taken and the associated value.

Selected DS: select one or more data sources by click on the data source in the All DS box, the selected data source will be moved to the Selected DS box.

Spatial Characterization Operator: select the spatial characterization operator from the dropdown list. Following characterization measures are allowed, epicenter, coverage, min, max, sum and avg (refer Figure A.15).

Create Query: click this button to create an intermediate query

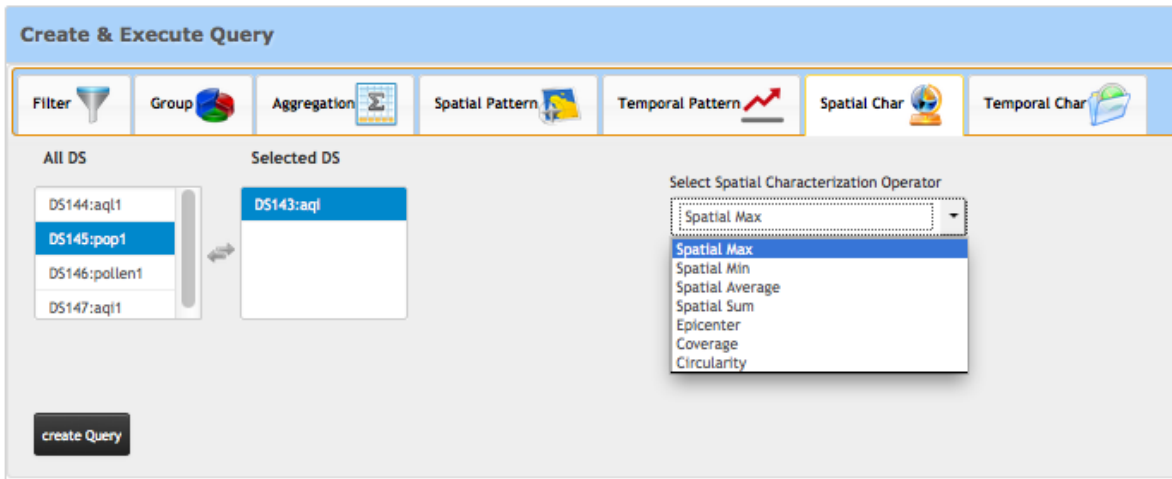


Figure A.15: Spatial Characterization Operator Configuration

Spatial Pattern Matching

Spatial pattern matching operator takes E-mage from an E-mage stream and a two-dimensional pattern as inputs. Users can choose to normalize the pattern resolution as well as values for matching purpose (refer Figure A.16). There are two options to get the 2D pattern which are uploading from an image file (e.g., bmp and png format) or generating from the system.

Select Pattern Source: select between two options to get the 2D pattern which are “Input from File” or “Create New”. For the first option, users have to upload an image file (e.g., bmp or png format). For the later, users need to specify the resolution of the pattern (i.e., number of rows and number of columns), and a spatial distribution including Gaussian and Linear distribution. For Gaussian distribution, we require a center point coordinate, x variance, y variance, and the amplitude. For Linear distribution, we require a stating point,

starting value, directional gradient and value gradient (refer Figure A.17).

Create Query: click this button to create an intermediate query

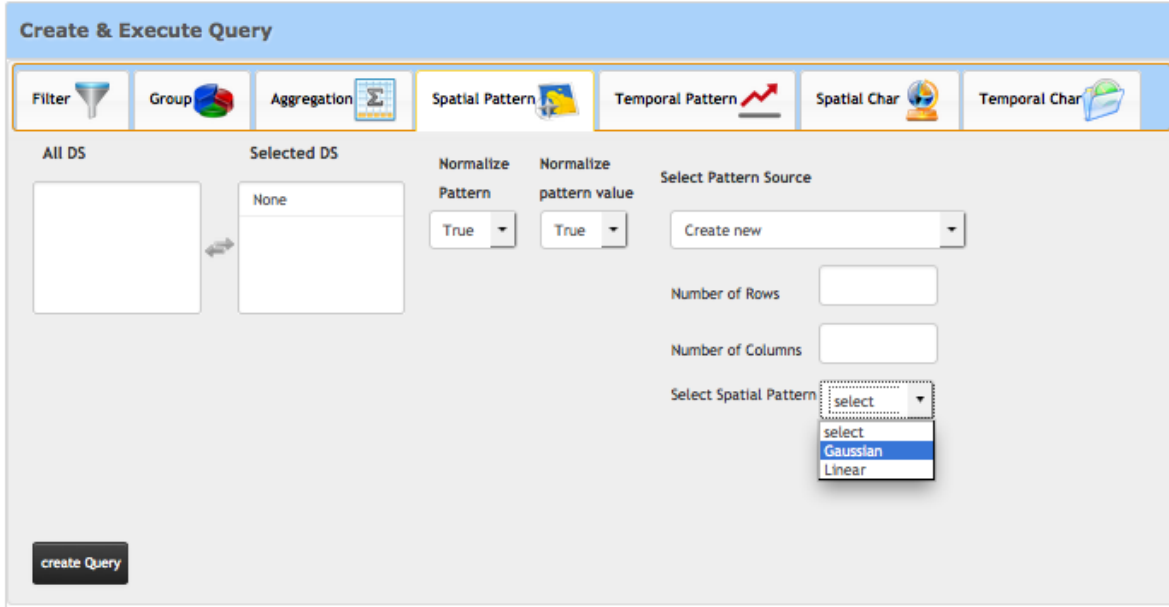
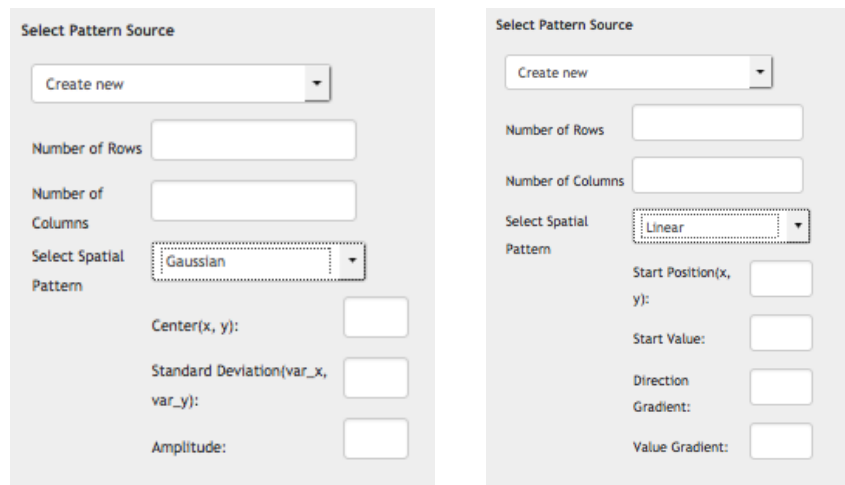


Figure A.16: Spatial Pattern Matching Operator Configuration



(a) Gaussian Distribution

(b) Linear Distribution

Figure A.17: Create Spatial Pattern using Gaussian and Linear Spatial Distribution

Temporal Characterization

The input of this operator is a *stel stream*. This operator takes a window of stels from a stream, and computes its characteristics related to temporal property like displacement, velocity, acceleration, growth-rate, and periodicity. The output is also a stel stream. User need to provide the time window in seconds over which the measure is taken.

Create Query: click this button to create an intermediate query

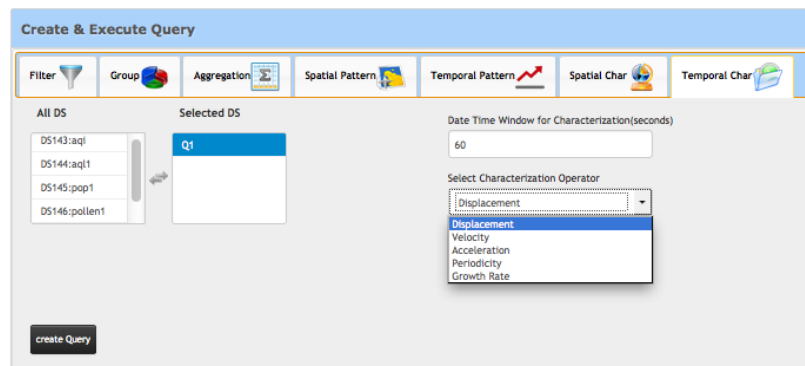


Figure A.18: Temporal Characterization Operator Configuration

Temporal Pattern Matching

Temporal pattern matching operator takes 1D pattern and a window of stels stream as input, and tries to find the similarity between these two inputs. Output is still a stel stream. Each stel stores not only the similarity value, but also the center position of the sub window of stels where there is the highest similarity value. Users have to state the window size in seconds over the input stream, and choose whether they want to normalize pattern size or pattern value. Users can select the pattern from file (CVS format) or generate the temporal pattern using our system tool. There are three patterns supported in our current version which are Linear, Periodic, and Exponential growth. Users need to specify the sampling

rate, and the duration in seconds. For linear pattern, users need to provide slope and Y-intercept. Exponential pattern requires the base value and the scale factor. Finally, the periodic pattern parameters include frequency, amplitude, and phase delay.

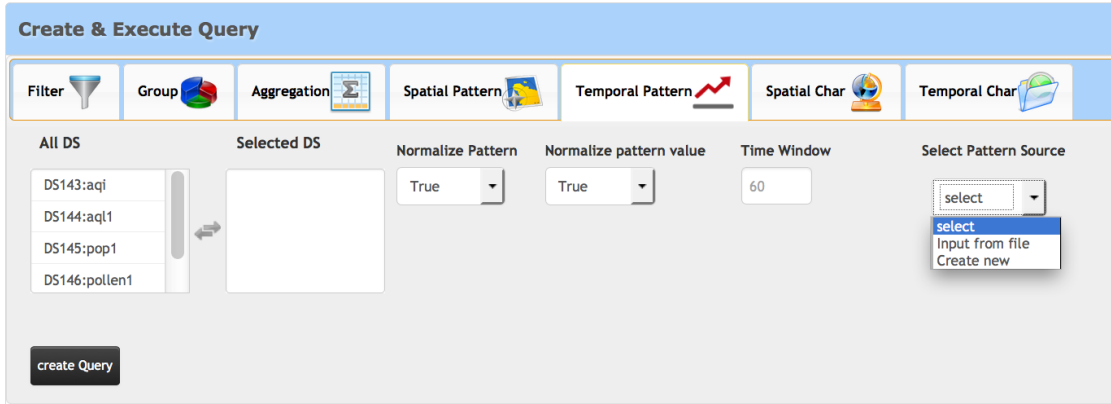


Figure A.19: Temporal Pattern Matching Operator Configuration

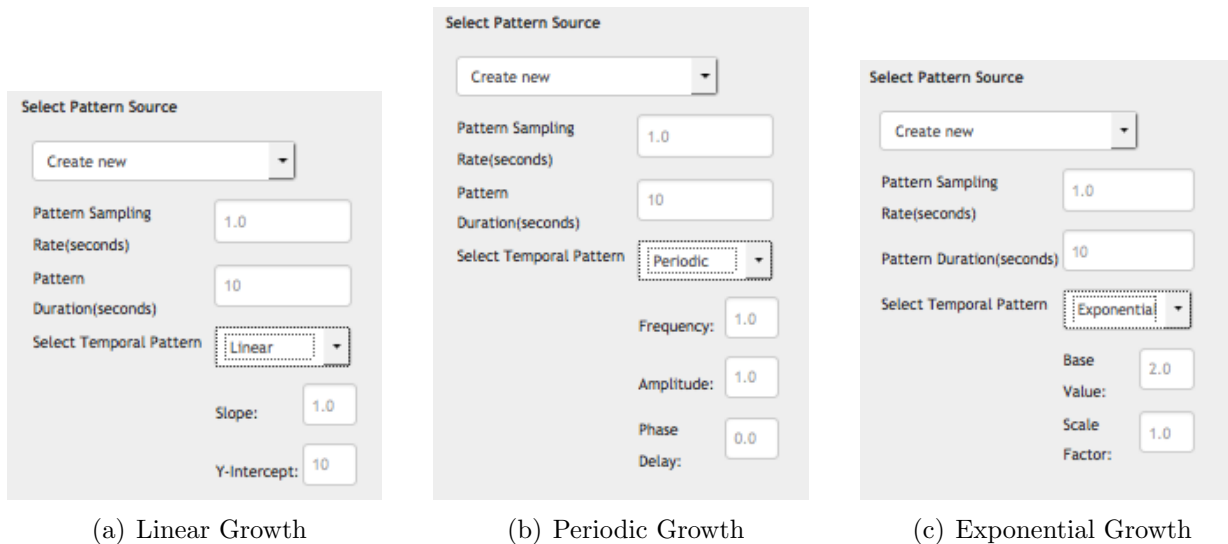


Figure A.20: Create Temporal Pattern using Linear, Periodic, and Exponential Growth

A.4.2 Registering Query

Each operator works on an input data source and a set of relevant parameters. An input data source can be any of the data sources registered in the data source panel, or any of the

intermediate queries. We differentiate between registered queries and intermediate queries as follows.

- Registered queries are finalized queries ready to be executed as standing queries on incoming data streams here onwards.
- Intermediate queries are parts of a final (composite) query still under the construction.

For example if a query involves Grouping after Filtering on data source 7. The source would first create an intermediate query for Filtering on data source 7. This intermediate query would then be used as an input for the Grouping operator. Once satisfied with the operators, the user can save the query for further execution.

Query Name: put a name of the query in the textbox on the left as shown in Figure A.21.

Reset Query: select this button to reset the query. All intermediate queries will be removed from the system.

Save Query: select this button to save/register the query to the system. Before saving, users have to config the final resolution of the query E-mage result using the popup window (Figure A.22).

Final Resolution: config the final frame parameters in this window. The final resolution E-mage of the registered query has the same parameters as the initial resolution of the data source. Click “Save Changes” to register the query and it would be shown as available in the Registered Queries panel. Click “Close” to cancel query registering process and go back to the Home page.

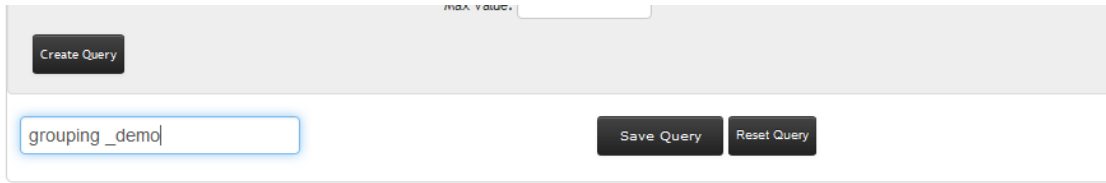


Figure A.21: Query Registration

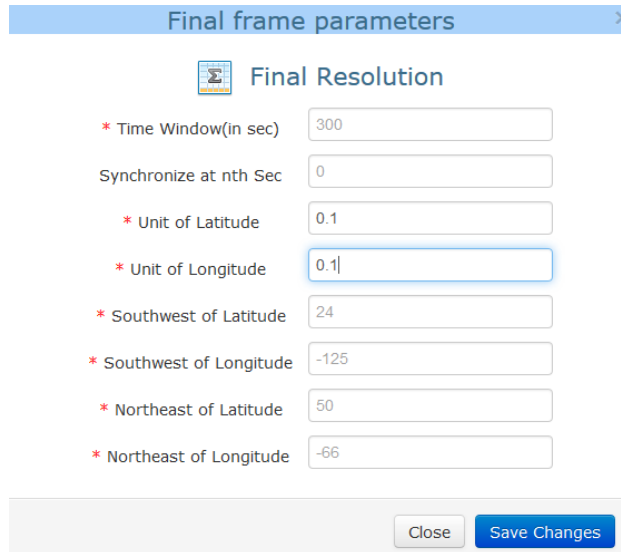


Figure A.22: Query Final Frame Parameters

A.5 Query Graph Panel

This panel displays a query graph to aid users in forming query. Nodes in the tree represent data source, operator, or intermediate/finalized query. Its root node is the finalized query, leaf nodes are the data sources, and other nodes are either operators or intermediate queries. Each query node is represented in a rounded rectangle shape, while each data source and operator node are represented by an ellipse shape. A direction in the tree show how the data flow from end to end.

A.5.1 Visualizing Query Graph

In the following example, three data sources and three operators are used to create one finalized query. First, we apply Filter operator on the data source “ds357”. This step creates an intermediate query called “Q1” as shown in Figure A.23(a). We repeat the same step on the data source “ds350” and “ds361”. The intermediate queries “Q2” and “Q3” are generated respectively. Second, we apply Aggregation operator on those intermediate query and generate a new intermediate query called “Q4” as shown in Figure A.23(b). Finally, we apply Grouping operator on the query “Q4” to form a complex situation model A.23(c).

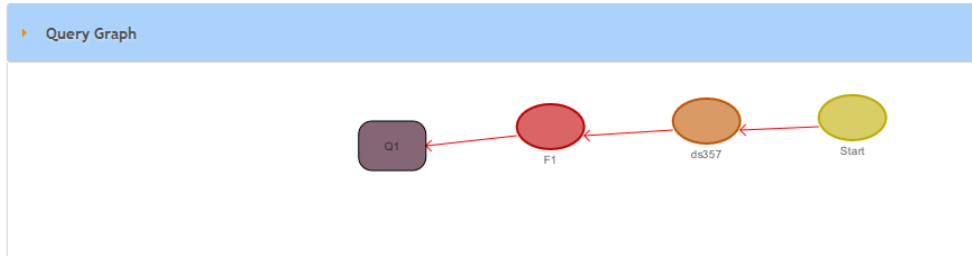
A.6 Registered Queries Panel

The registered queries panel is in the top left of the web interface. This panel in Figure A.24 displays the list of queries registered with the system in the table, and includes the following information: *query’s ID (QID)*, *query’s name (Query Name)*, and *status* (“Stopped” or “Running”).

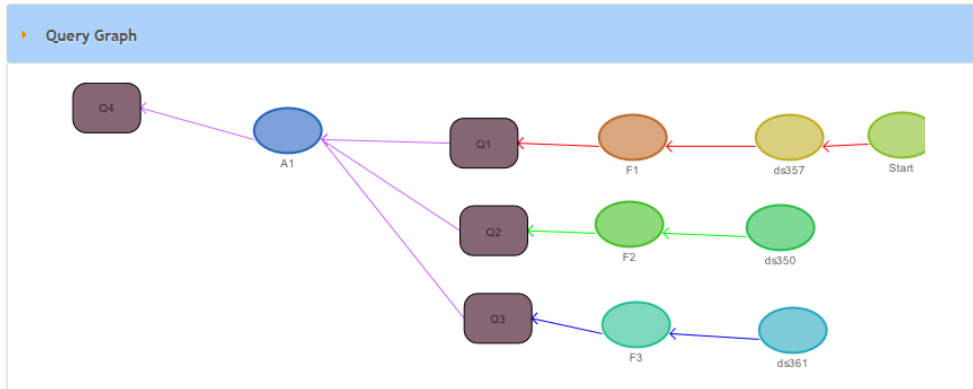
A.6.1 Executing Registered Query

Based on the query’s status, different controller buttons are shown.

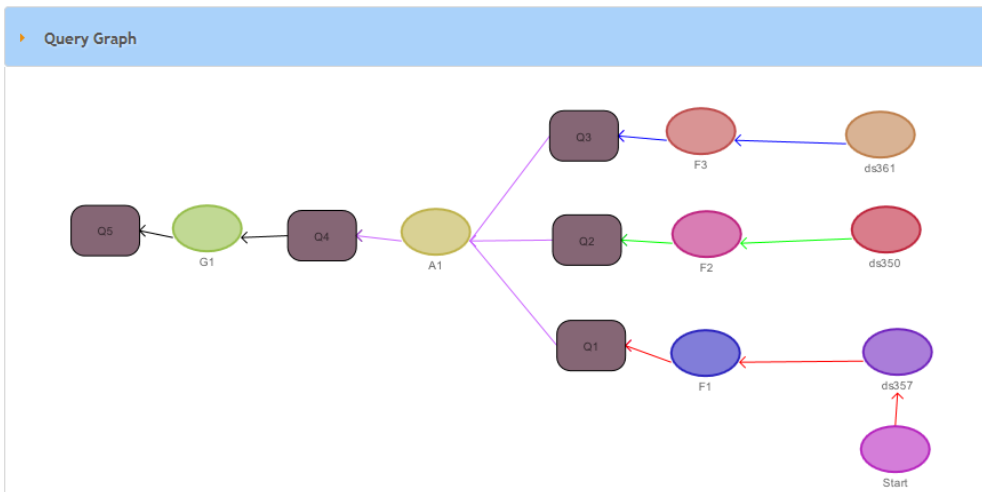
- If the status is “Stopped” which means the query has been registered but has not been running, only the run [R] button will be shown. User can click this button to instructs the backend to compute the query results. It would be applied as a standing queries on incoming data streams here onwards. At this point, based on the time window, it



(a) Intermediate Query: Filter



(b) Intermediate Query: Aggregation Follows Filter



(c) Finalized Query: Grouping on the Previous Intermediate Query

Figure A.23: Evolving Query Graph

may take longer period of time for the first E-mage result to be ready. Then its status is changed to “Running”.

Important! Please click [R] button only once, otherwise multiple execution

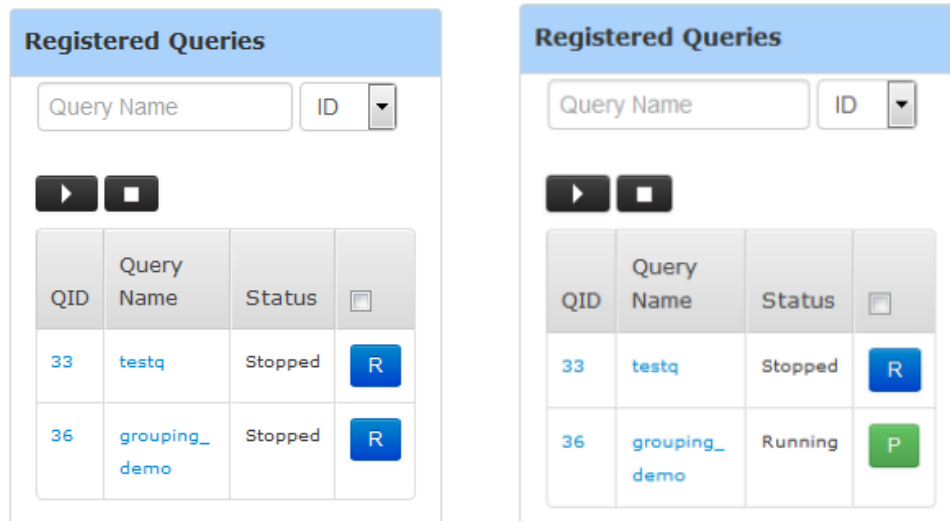


Figure A.24: Registered Query Panel

processes will be created and cause the system to malfunction.

- If the status is “Running” which mean the query has been registered and running to get the current result E-mage, only pause [P] button will be shown. Users can pause the query process by clicking on [P] button. When the query execution process is paused, its status is changed to “Stopped” as shown in the right of Figure A.24

A.6.2 Searching Registered Query

To search for a particular query, users can just type the query name into the text box. Only the query that match the user’s input will be shown in the list.

A.6.3 Visualizing Output E-mage

To view the query result, click on the query's name link. The new page will be opened as show in Figure A.25. Users are allow to open multiple result pages for convenient in comparing the results from different queries. On the top of this query page, information about the registered query are presented including *Query ID*, *Start Time*, *End Time*, *Min Value*, and *Max Value*. We present an EventShop output E-mage on the map using Leaflet and D3 packages.

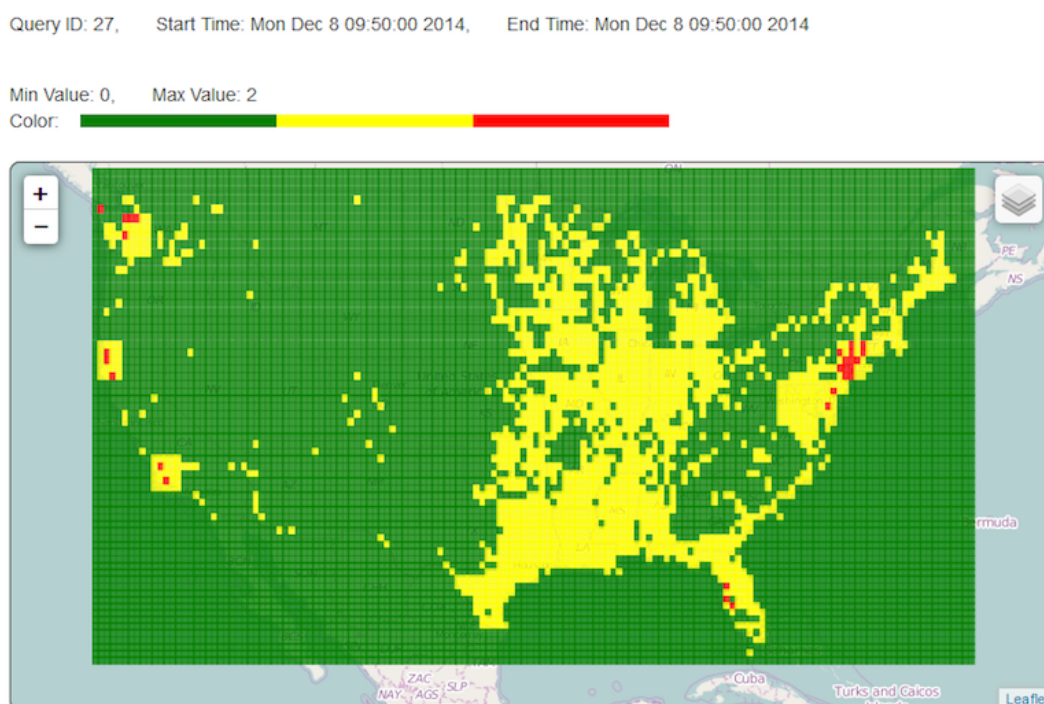


Figure A.25: Query Result Page

Different types of queries result in different types of outputs, which may include one (or more) of the following: *numeric value*, *temporal value*, *location*, *E-mage*. The output type is decided by the last operator used in the query. The output type for each of the possible last operators is summarized in Table A.2.

While any of the operators can be used to retrieve situational information, applications focusing on situation classification will tend to use “Classification” as the final query operator.

S.No	Type (of the last operator)	Numeric value	Temporal value	Spatial location	E-mage
1	Filter				X
2	Aggregate				X
3	Classification				X
4	Characterization: Spatial	X	X	X	
5	Characterization: Temporal	X	X		
6	Pattern matching: Spatial	X	X		
7	Pattern matching: Temporal	X	X		

Table A.2: Output formats for different types of queries

A.7 Taking Actions

The users can choose to send out mass-customized messages to people based on the situations detected. For example, the system can be configured to send out tweets to people in ‘unsafe hurricane zone’ to move to the nearest ‘safe hurricane zone’ location if they have talked about this topic in their tweets. This process can be configured as shown in Figure A.26

The users need to configure data source for obtaining the usernames of intended audience, the condition for sending out alerts (IF clause), conditions for redirecting to another location (the THEN clause) and any other (optional) accompanying messages.

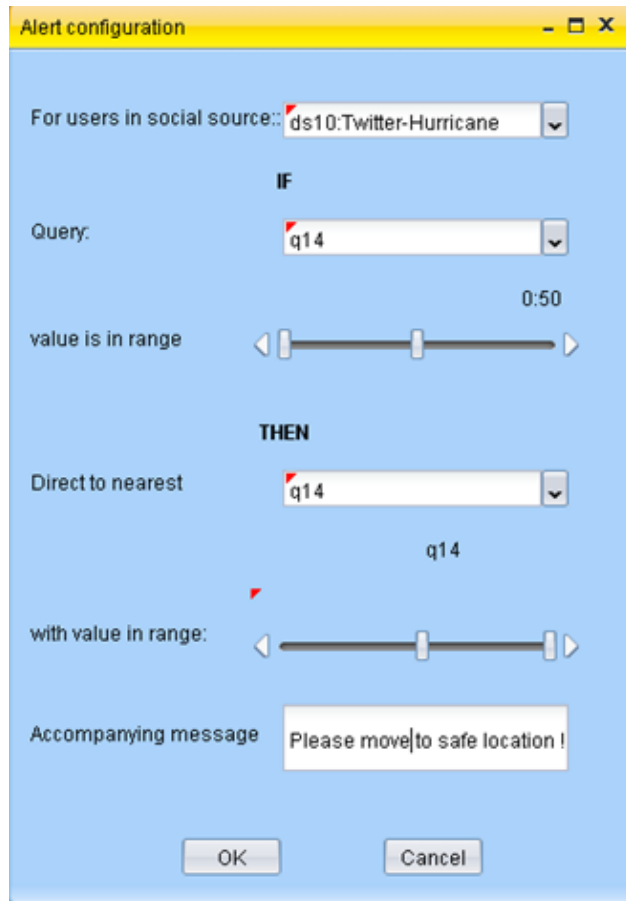


Figure A.26: Configuration Options for Sending Alerts Based on Different Situations

Appendix B

EventShop Rest API

This section describe about Rest API providing in our EventShop system.

B.1 Data Source

We provide four basic functions including register data source, enable data source, disable data source, and delete data source. For the EventShop server, the open-source version is hosted at "http://sln.ics.uci.edu:8085/eventshoplinux/". Users can replace [host] keyword with that url throughout examples in this section. In addition, each rest API requires a valid user's ID. Users should provide their own user's ID, or use the demo account on the open source server which is "81".

B.1.1 Register Data Source

This API is used for registering or creating a new data source in the EventShop system.

Type	POST
Content-Type	application/json
Path	[host]*/rest/dataSourceService/createDataSource

We provide some example requests here.

(i) Example request for the **rest endpoint** data source providing data in **csv filed** format:

```

1 {
2     "Name": "AQI_CSV",
3     "Theme": "AQI",
4     "Url": "http://sln.ics.uci.edu:8085/eventshoplinux/aqi.csv",
5     "Format": "rest",
6     "User_Id": 81,
7     "Syntax" : "\"timestamp\": \"DATETIME\", \"theme\": \"STRING\", \"value\":
      \"NUMBER\", \"loc\": {\"lon\": \"NUMBER\", \"lat\": \"NUMBER\"}",
8     "Time_Window": 300000,
9     "Latitude_Unit": 2.0,
10    "Longitude_Unit": 2.0,
11    "boundingbox": "24,-125,50,-66",
12    "Sync_Time": 300000,
13    "Wrapper_Name": "csvField",
14    "Wrapper_Key_Value": "{ \"datasource_type\": \"point\", \"spatial_wrapper
      \": \"sum\", \"loc.lat_index\" : 0, \"loc.lon_index\" : 1, \"value_index
      \": 2}",
15    "Bag_Of_Words": ""
16 }

```

(ii) Example request for the **rest endpoint** data source providing data in **json** format:

```

1 {
2     "Name": "Weather",
3     "Theme": "OpenWeather",
4     "Url": "http://api.openweathermap.org/data/2.1/find/station",
5     "Format": "rest",
6     "User_Id": 81,
7     "Syntax" : "\"timestamp\": \"DATETIME\", \"theme\": \"STRING\", \"value\":
      \"NUMBER\", \"loc\": {\"lon\": \"NUMBER\", \"lat\": \"NUMBER\"}",
8
9     "Time_Window": 300000,
10    "Latitude_Unit": 0.2,
11    "Longitude_Unit": 0.2,
12    "boundingbox": "32.249974,-5.969727,42.000325,5.257813",
13    "Sync_Time": 300000,
14    "Bag_Of_Words": "",

```

```

15     "Wrapper_Name": "json",
16     "Wrapper_Key_Value": "{\"datasource_type\": \"point\", \"spatial_wrapper
    \": \"sum\", \"isList\": true, \"rootElement\": \"\", \"tokenizeElement
    \": \"list\", \"loc.lat_path\": \"coord.lat\", \"loc.lon_path\": \"
    coord.lon\", \"value_path\": \"wind.speed\"}"
17 }

```

(iii) Example request for the **rest endpoint** data source providing data in **XML** format:

```

1 {
2     "Name": "XMLtest",
3     "Theme": "XMLtest",
4     "Url": "http://uk-air.defra.gov.uk/assets/rss/current_site_levels.xml"
5
6     "Format": "rest",
7     "User_Id": 81,
8     "Syntax" : "\"timestamp\": \"DATETIME\", \"theme\": \"STRING\", \"value\":
9         \"NUMBER\", \"loc\": {\"lon\": \"NUMBER\", \"lat\": \"NUMBER\"}",
10
11     "Time_Window": 300000,
12     "Latitude_Unit": 0.2,
13     "Longitude_Unit": 0.2,
14     "boundingbox": "32.249974,-123.969727,42.000325,-114.257813",
15     "Sync_Time": 300000,
16     "Wrapper_Name": "xml",
17     "Wrapper_Key_Value": "{\"datasource_type\": \"point\", \"
    spatial_wrapper\": \"sum\", \"isList\": true, \"rootElement\": \"rss
    /channel\", \"tokenizeElement\": \"item\", \"loc_lat_path\" : \"/
    item/description/text()\", \"loc_lat_grok\": \"\": {%NUMBER:loc_lat}&
    deg;{%GREEDYDATA:waste1}{%NUMBER:loc_lon}&quot;N{%GREEDYDATA:waste
    2}{%WORD:status} at index {%NUMBER:index}\", \"loc_lon_path\": \"/
    item/description/text()\", \"loc_lon_grok\": \"\": {%NUMBER:loc_lat}&
    deg;{%GREEDYDATA:waste1}{%NUMBER:loc_lon}&quot;N{%GREEDYDATA:waste
    2}{%WORD:status} at index {%NUMBER:index}\", \"value_path\": \"/item
    /description/text()\", \"value_grok\": \"\": {%NUMBER:lat}&deg;{%
    GREEDYDATA:waste1}{%NUMBER:lon}&quot;N{%GREEDYDATA:waste2}{%WORD:
    status} at index {%NUMBER:value}\", \"timestamp_path\": \"/item/
    pubDate/text()\", \"timestamp_format\": \"dd MMM yyyy hh:mm:ss\",
    \"timestamp_grok\": \"%{%WORD:garbage}, {%GREEDYDATA:timestamp} +{%
    NUMBER:garbage1}\"}"
17 }

```

The response of this API is the data source's ID as shown in the example below.

```

1 {
2     "ID": 1234

```

```
3 }
```

B.1.2 Enable Data Source

This API allows user to start the data ingestor process on a particular data source.

Type	POST
Content-Type	application/json
Path	[host]/eventshoplinux/rest/dataSourceService/enableDataSource

Example request is

```
1 {  
2   "ID": 1234  
3 }
```

Example response is

```
1 Enabled Data source with ID 1234
```

B.1.3 Disable Data Source

This API allows users to stop data ingestor process on a particular data source.

Type	POST
Content-Type	application/json
Path	[host]/eventshoplinux/rest/dataSourceService/disableDataSource

Example request is

```
1 {  
2   "ID": 1234  
3 }
```

Example response is

```
1 Disabled Data source with ID 1234
```

B.1.4 Delete Data Source

Allow one to delete a particular data source. This process cannot be undone. If the data source is currently used in any registered query, it cannot be deleted.

Type	POST
Response format	JSON
Path	http://{host}[:port]/eventshoplinux/rest/dataSourceService/deleteDataSource

Example request is

```
1 {  
2   "ID": 1234  
3 }
```

Example response is

```
1 Data source is deleted successfully.
```

B.1.5 View Data Source Metadata

This API allows users to data source metadata.

Type	GET
Content-Type	application/json
Path	http://{host}:{port}/eventshoplinux/webresources/datasourceservice/datasources/{Data source ID}

Example URL request is

```
1 http://eventshop.ics.uci.edu:8085/eventshoplinux/webresources/  
   datasourceservice/datasources/9
```

Example response is

```
1 {
2   "srcVarName":null,
3   "srcName":"windspeed_json",
4     "control":0,
5     "type":null,
6     "status":"Connecting",
7     "srcTheme":"windspeed_json",
8     "initParam":{"dsQuery":null,
9     "genEmage":true,"start":0,"timeWindow":30000000,"syncAtMilSec":30000,"end
    ":0,"numOfColumns":7,"numOfRows":7,"longUnit":1.5,"spatial_wrapper":
    null,"timeType":0,"neLat":60.0,"neLong":40.0,"swLat":50.0,"swLong":
    30.0,"latUnit":1.5},"url":"http://eventshop.ics.uci.edu/dabuntu/
    weatherList.json","archive":0,"wrapper":{"wrprId":"10","wrprName":
    "json","wrprType":"PULL","wrprBagOfWords":"","wrprVisualTransMat":"","
    wrprVisualColorMat":"","wrprVisualMaskMat":"","wrprVisualIgnore":0,
    "wrprCSVFileURL":null,"wrprArchStartTime":null,"wrprArchEndTime":null
    ,"wrprArchGenRate":null,"wrprKeyValue":{"\"datasource_type\": \"point
    \",\"spatial_wrapper\": \"sum\", \"isList\": true, \"rootElement\":
    \"\", \"tokenizeElement\": \"list\", \"loc.lat_path\": \"coord.lat\", \"
    loc.lon_path\": \"coord.lon\", \"value_path\": \"wind.speed\", \"
    date_time_path\": \"dt\", \"date_time_format\": \"Long\", \"name_path
    \": \"name\"}}}, \"syntax\":\"timestamp\": \"DATETIME\", \"theme\": \"
    STRING\", \"value\": \"NUMBER\", \"loc\": {\"lon\": \"NUMBER\", \"lat\":
    \"NUMBER\"}, \"date_time\": \"DATETIME\", \"name\": \"STRING\"\", \"
    srcID\":\"9\", \"srcFormat\":\"file\", \"supportedWrapper\":\"json\", \"bagOfWords\":
    [\"\"], \"userId\":null, \"access\":null, \"boundingbox\":null, \"unit\":0, \"
    emailOfCreator\":null, \"visualParam\":{\"colorMatPath\":null, \"
    translationMatrix\":null, \"maskPath\":null, \"ignoreSinceNumber\":0, \"
    tranMatPath\":null, \"colorMatrix\":null}, \"finalParam\":{\"dsQuery\":null, \"
    genEmage\":false, \"start\":0, \"timeWindow\":0, \"syncAtMilSec\":0, \"end\":0, \"
    numOfColumns\":0, \"numOfRows\":0, \"longUnit\":0.0, \"spatial_wrapper\":null, \"
    timeType\":null, \"neLat\":0.0, \"neLong\":0.0, \"swLat\":0.0, \"swLong\":0.0, \"
    latUnit\":0.0}}
```

B.2 STT Rule Engine API

B.2.1 Create Rlue

To create a custom rule for the data source.

There are three components required for each rule.

“source”: to specified which data source you want to apply this rule

“rules”: is the list of conditional rules you want to apply in this particular custom rule. (similar to WHERE clause in SQL). Each conditional rule consists of three components: dataField, ruleOperator, and ruleParamters as described in previous section.

“extractFields”: to limit the returned fields in your result, you can specify which fields you want to extract. (Note: the stt_id, stt_when, stt_where are required in the result and are returned by default, so you do not have to specified them here) Result: is rule id. Please take note of this rule id since you will have to use it in other rest services such as for search to your STT data

Type: POST

Content-Type: application/json

Path: /eventshoplinux/rest/rulewebservice/rule

Note: change source value to your existing data source

Following are some examples of rule file configuration:-

(i) Rule with “>” **greater than** operator.

```
1 {
2     "source": "dsXX",
3     "rules": [
4         {
5             "dataField": "value",
6             "ruleOperator": ">",
7             "ruleParameters": "50"
8         }
9     ],
10    "extractFields": "location.lat,location.lon,value,"
11 }
```

(ii) Rule with **Geo Coordinates**.

```
1 {
2     "source": "dsXX",
```



```

3     "rules": [
4         {
5             "dataField": "location",
6             "ruleOperator": "coordinates",
7             "ruleParameters": "7.45,69.25,32.6,92.0"
8         }
9     ],
10    "extractFields": "location.lat,location.lon,value,"
11 }

```

(iii) Rule with **Equals** operator. Which is used to compare strings.

```

1 {
2     "source": "dsXX",
3     "rules": [
4         {
5             "dataField": "theme",
6             "ruleOperator": "equals",
7             "ruleParameters": "Krumbs_SB_Test"
8         }
9     ],
10    "extractFields": "location.lat,location.lon,value,"
11 }

```

(iv) Composite rule. With **AND** logic. Example Data Input:

```

1 {
2     "source": "ds9061",
3     "rules": [
4         {
5             "dataField": "stt_what.intent_used_synonym_index.value",
6             "ruleOperator": "equals",
7             "ruleParameters": "0"
8         },
9         {
10            "dataField": "stt_what.intent_index_in_category.value",
11            "ruleOperator": "equals",
12            "ruleParameters": "1"
13        }
14    ],
15    "extractFields": "stt_what.intent_used_synonym_index,stt_what.
16    caption,"

```

Example Result: 76

B.2.2 Get Rule

to show detail of registered rule

Type: GET

Content-Type: application/json

Path: /eventshoplinux/rest/rulewebservice/rule/ruleid

Example URL: <http://sln.ics.uci.edu:8085/eventshoplinux/rest/rulewebservice/rule/76>

Example result:

```
1 {"rules":[{"ruleID":null,"ruleParameters":"0","ruleName":null,"dataField":"stt_what.intent_used_synonym_index.value","ruleOperator":"equals"},{"ruleID":null,"ruleParameters":"1","ruleName":null,"dataField":"stt_what.intent_index_in_category.value","ruleOperator":"equals"}],"source":"ds9061","ruleID":76,"ruleName":null,"extractFields":"stt_what.intent_used_synonym_index,stt_what.caption","userId":0}
```

B.2.3 Get List of Rules

Method: Get Path: /rest/rulewebservice/getrules/dsid Parameters: dsid: id of the rule Ex-

ample: <http://sln.ics.uci.edu:8085/eventshoplinux/rest/rulewebservice/getrules/ds9061> Ex-

ample result:

```
1 [{"rules":[],"source":"ds9061","ruleID":68,"ruleName":"","extractFields":"stt_what,stt_value,raw_data","userId":0},{"rules":[{"ruleID":null,"ruleParameters":"0","ruleName":null,"dataField":"stt_what.intent_used_synonym_index.value","ruleOperator":"equals"},{"ruleID":null,"ruleParameters":"1","ruleName":null,"dataField":"stt_what.intent_index_in_category.value","ruleOperator":"equals"}],"source":"ds9061","ruleID":76,"ruleName":"","extractFields":"stt_what.intent_used_synonym_index,stt_what.caption","userId":0}]
```

B.3 Search STT and Eimage

(i) to search for STT data in EventShop within bounding box and time interval.

Method: GET

Path: /rest/sttwebservice/search/ruleid/box/minlat,minlon/maxlat,maxlon/starttime/endtime

Parameters: ruleid, minlat,minlon,maxlat,maxlon, starttime, endtime

ruleid: For the default rule, please refer to the table above. In this default rule, all data can be returned. You can also create a custom rule, and use that instead of default rule. We will explain about custom rule further in the next section.

minlat,minlon: to specify minimum point (bottom left) of the bounding box area of your interest. If “null” value is used, the within operator will be ignored.

maxlat,maxlon: to specify maximum point (top right) of the bounding box area of your interest. If “null” value is used, the within operator will be ignored.

starttime: to specify start time of your interest. The result will show all STT after this start time. Its format is timestamp in millisecond. If ”null” value is used, the parameter will be ignored.

endtime: to specify end time of your interest. The result will show all STT before this end time. Its format is timestamp in millisecond. If ”null” value is used, the parameter will be ignored.

Note: If one of the two points of the bounding box is missing, the within operator will be ignored. Unlike bounding box, for time interval you may specify either start or end time, or both of them.

Example:

http://sln.ics.uci.edu:8085/eventshoplinux/rest/sttwebservice/search/
68/box/null/null/null/null - to show all STT

http://sln.ics.uci.edu:8085/eventshoplinux/rest/sttwebservice/search/
68/box/10.0,-130.0/40.0,-110.0/1456597468829/1456597468830

Example Result:

```
1 [ { "stt_id" : "cFWzMbnzoJ26" ,
2   "stt_where" : { "point" : [ 37.300049 , -121.9734274] } ,
3   "stt_when" : { "datetime" : 1456597468829 } ,
4   "stt_what" : { "media_source_photo" : { "value" : "" } , "
      intent_used_synonym" : { "value" : "Cars" } , "
      intent_used_synonym_index" : { "value" : "0" } , "
      intent_index_in_category" : { "value" : "1" } , "intent_name" : { "
      value" : "Cars" } , "intent_category_name" : { "value" : "Transport" } ,
      "intent_category_id" : { "value" : "SoGnQ0gml6" } , "intent_emoji_id"
      : { "value" : "diNCCZa1NX" } , "intent_emoji_unicode" : { "value" : "1f
      60d" } , "caption" : { "value" : "" } , "media_source_audio" : { "value"
      : "" } } ,
5   "stt_value" : 1 ,
6   "raw_data" : "{\\"media\\":[{\\"when\\":{\\"start_time\\":\\"2016-02-27T10:24:28
      .829Z\\",\\"end_time\\":\\"2016-02-27T10:24:28.829Z\\"},\\"where\\":{\\"
      geo_location\\":{\\"latitude\\":38.300049,\\"longitude\\":-121.9734274},\\"
      revgeo_places\\":[{\\"city\\":\\"San Jose\\",\\"country\\":\\"United States\\"
      ,\\"latitude\\":37.300058,\\"longitude\\":-121.973325,\\"name\\":\\"1310
      Vernal Dr\\",\\"unformatted_address\\":\\"1310 Vernal Dr, San Jose, CA 951
      30\\",\\"street\\":\\"1310 Vernal Dr\\"}]}],\\"why\\":[{\\"intent_used_synonym
      \\":\\"Cars\\",\\"intent_used_synonym_index\\":0,\\"
      intent_index_in_category\\":1,\\"intent_name\\":\\"Cars\\",\\"
      intent_category_name\\":\\"Transport\\",\\"intent_category_id\\":\\"SoGnQ0
      gml6\\",\\"intent_emoji_id\\":\\"diNCCZa1NX\\",\\"intent_emoji_unicode\\":\\"
      1f60d\\"}],\\"media_type\\":\\"PHOTO\\",\\"media_source\\":{\\"default_src\\":
      \\",\\"mime_type\\":\\"\\",\\"caption\\":\\"SUV, \\n At 1310 Vernal Dr\\"},
      {\\"when\\":{\\"start_time\\":\\"2016-02-27T10:24:28.829Z\\",\\"end_time\\":
      \\"2016-02-27T10:24:28.829Z\\"},\\"where\\":{\\"geo_location\\":{\\"latitude
      \\":37.300049,\\"longitude\\":-121.9734274},\\"revgeo_places\\":[{\\"city\\"
      :\\"San Jose\\",\\"country\\":\\"United States\\",\\"latitude\\":37.300058,\\"
      longitude\\":-121.973325,\\"name\\":\\"1310 Vernal Dr\\",\\"
      unformatted_address\\":\\"1310 Vernal Dr, San Jose, CA 95130\\",\\"street
      \\":\\"1310 Vernal Dr\\"}]}],\\"media_type\\":\\"AUDIO\\",\\"media_source\\":{\\"
      default_src\\":\\"\\",\\"mime_type\\":\\"\\"}]}],\\"id\\":\\"cFWzMbnzoJ26\\"}"
      , {...}, ... ]
```

(ii) to search for STT data in EventShop within circle area and time interval.

Method: GET

Parameters: {ruleid}, {centerlat,lon,radius}, {starttime}, {endtime}

ruleid: For the default rule, please refer to the table above. In this default rule, all data can be returned. You can also create a custom rule, and use that instead of default rule. We will explain about custom rule further in the next section.

centerlat,lon,radius: to specify the center point and radius of your circle area of interest. If "null" value is used, the within operator will be ignored.

starttime: to specify start time of your interest. The result will show all STT after this start time. Its format is timestamp in millisecond. If "null" value is used, the parameter will be ignored.

endtime: to specify end time of your interest. The result will show all STT before this end time. Its format is timestamp in millisecond. If "null" value is used, the parameter will be ignored.

Note: For time interval you may specify either start or end time, or both of them.

Example URLs:

`http://sln.ics.uci.edu:8085/eventshoplinux/rest/sttwebservice/search/68/circle/null/null/null - to show all STT`

`http://sln.ics.uci.edu:8085/eventshoplinux/rest/sttwebservice/search/68/circle/37.0,-120.0,10.0/1456597468829/1456597468830`

Example Result:

1 [{ "stt_id" : "cFWzMbnzoJ26" ,

```

2  "stt_where" : { "point" : [ 37.300049 , -121.9734274] } ,
3  "stt_when" : { "datetime" : 1456597468829 } ,
4  "stt_what" : { "media_source_photo" : { "value" : "" } , "
    intent_used_synonym" : { "value" : "Cars" } , "
    intent_used_synonym_index" : { "value" : "0" } , "
    intent_index_in_category" : { "value" : "1" } , "intent_name" : { "
    value" : "Cars" } , "intent_category_name" : { "value" : "Transport" } ,
    "intent_category_id" : { "value" : "SoGnQ0gml6" } , "intent_emoji_id"
    : { "value" : "diNCCZa1NX" } , "intent_emoji_unicode" : { "value" : "1f
    60d" } , "caption" : { "value" : "" } , "media_source_audio" : { "value"
    : "" } } ,
5  "stt_value" : 1 ,
6  "raw_data" : "{ \"media\": [{ \"when\": { \"start_time\": \"2016-02-27T10:24:28
    .829Z\", \"end_time\": \"2016-02-27T10:24:28.829Z\" }, \"where\": { \"
    geo_location\": { \"latitude\": 38.300049, \"longitude\": -121.9734274 }, \"
    revgeo_places\": [ { \"city\": \"San Jose\", \"country\": \"United States\"
    , \"latitude\": 37.300058, \"longitude\": -121.973325, \"name\": \"1310
    Vernal Dr\", \"unformatted_address\": \"1310 Vernal Dr, San Jose, CA 951
    30\", \"street\": \"1310 Vernal Dr\" } ] }, \"why\": [ { \"intent_used_synonym
    \": \"Cars\", \"intent_used_synonym_index\": 0, \"
    intent_index_in_category\": 1, \"intent_name\": \"Cars\", \"
    intent_category_name\": \"Transport\", \"intent_category_id\": \"SoGnQ0
    gml6\", \"intent_emoji_id\": \"diNCCZa1NX\", \"intent_emoji_unicode\": \"
    1f60d\" } ] }, \"media_type\": \"PHOTO\", \"media_source\": { \"default_src\":
    \"\", \"mime_type\": \"\" }, \"caption\": \"SUV, \n At 1310 Vernal Dr\" },
    { \"when\": { \"start_time\": \"2016-02-27T10:24:28.829Z\", \"end_time\":
    \"2016-02-27T10:24:28.829Z\" }, \"where\": { \"geo_location\": { \"latitude
    \": 37.300049, \"longitude\": -121.9734274 }, \"revgeo_places\": [ { \"city\"
    : \"San Jose\", \"country\": \"United States\", \"latitude\": 37.300058, \"
    longitude\": -121.973325, \"name\": \"1310 Vernal Dr\", \"
    unformatted_address\": \"1310 Vernal Dr, San Jose, CA 95130\", \"street
    \": \"1310 Vernal Dr\" } ] }, \"media_type\": \"AUDIO\", \"media_source\": {
    \"default_src\": \"\", \"mime_type\": \"\" } ] }, \"id\": \"cFWzMbnzoJ26\" } }
    , { ... }, ... ]

```

(iii) get Emage within the area of interest and time interval

Method: GET

Path: /eventshoplinux/rest/sttwebservice/search/{ruleid}/box/{minlat,
minlon}/{maxlat,maxlon}/{starttime}/{endtime}/{latunit,lonunit}/count/
null/emage

Parameters: ruleid, minlat, minlon, maxlat, maxlon, starttime, endtime, latunit, lonunit

Most of the parameters are similar to the STT search API above, except latunit, lonunit

latunit,lonunit: specify the resolution of grid in the Emage.

Example URL:

```
http://sln.ics.uci.edu:8085/eventshoplinux/rest/sttwebservice/search/
183/box/33.485539857568966,-117.97681121826173/33.71430842400522,-117.
62318878173829/1456531200000/1457481600000/0.0125,%200.0125/count/
null/emage
```

Example Result:

```
1 [{"cell":
2 {"rectangle":[{"point":[33.65,-117.85001]},{"point":[33.6625,-117.83751]}]},
3 "count":9.0,
4 "values":{"orderedlist":[
5 {"stt_id":"9bkLXpOiUQ","stt_where":{"point":[33.6527,-117.83975]},"stt_when":{"
  "datetime":1456743644157},"stt_what":{"media_source":{"value":"https://d3j
  4aoik7k8ki7.cloudfront.net/b478cfd8-52ac-4664-ab1b-e434c19dcc48.jpeg"},"
  caption":{"value":"Yawn, \n At 930 Bridge Rd"}},"stt_value":1},
6 {"stt_id":"asnXg9Fexm","stt_where":{"point":[33.652596,-117.83984]},"stt_when"
  :{"datetime":1456743735151},"stt_what":{"media_source":{"value":"https://d
  3j4aoik7k8ki7.cloudfront.net/69c9df78-2601-4e22-b0e8-ee3832f4e6bb.jpeg"},"
  caption":{"value":"Yawn, \n At 436 Bridge Rd"}},"stt_value":1},
7 {"stt_id":"TWhS2iKYDT","stt_where":{"point":[33.652596,-117.83984]},"stt_when"
  :{"datetime":1456744631563},"stt_what":{"media_source":{"value":"https://d
  3j4aoik7k8ki7.cloudfront.net/5044e076-eab1-4851-86cb-d98460fee342.jpeg"},"
  caption":{"value":"Yawn, \n At 436 Bridge Rd"}},"stt_value":1},
8 {"stt_id":"MfzqHWTpZu","stt_where":{"point":[33.652596,-117.83984]},"stt_when"
  :{"datetime":1456743483792},"stt_what":{"media_source":{"value":"https://d
  3j4aoik7k8ki7.cloudfront.net/e55f312b-fbe8-41d6-9c95-7be8a2f3f528.jpeg"},"
  caption":{"value":"Yawn, \n At 436 Bridge Rd"}},"stt_value":1},
9 {"stt_id":"c6VNs49Z1i","stt_where":{"point":[33.652596,-117.83984]},"stt_when"
  :{"datetime":1456726167729},"stt_what":{"media_source":{"value":"https://d
  3j4aoik7k8ki7.cloudfront.net/84b0f539-83c2-4577-ad2d-1c51fe59e27d.3gp"},"
  caption":{"value":""}},"stt_value":1},
10 {"stt_id":"5NidbzBzsU","stt_where":{"point":[33.652596,-117.83984]},"stt_when"
  :{"datetime":1456743195001},"stt_what":{"media_source":{"value":"https://d
  3j4aoik7k8ki7.cloudfront.net/4aa20084-8abf-4c4f-a191-18ecde8bca9d.jpeg"},"
  caption":{"value":"Yawn, \n At 436 Bridge Rd"}},"stt_value":1},
11 {"stt_id":"AxP7yTupVY","stt_where":{"point":[33.65266,-117.83986]},"stt_when":
  {"datetime":1456744196663},"stt_what":{"media_source":{"value":"https://d3
  j4aoik7k8ki7.cloudfront.net/046ba3ce-03ab-4459-9798-7ee047058027.jpeg"},"
  caption":{"value":"Yawn, \n At 420 Stanford"}},"stt_value":1},
12 {"stt_id":"H6Y7zHwgdJ","stt_where":{"point":[33.652596,-117.83984]},"stt_when"
  :{"datetime":1456745667609},"stt_what":{"media_source":{"value":"https://d
  3j4aoik7k8ki7.cloudfront.net/e8a0ef5b-95c8-4fb5-851a-c68c7471a55f.jpeg"},"
  caption":{"value":"Yawn, \n At 436 Bridge Rd"}},"stt_value":1},
```

```

13 {"stt_id":"G9AIboDZi0","stt_where":{"point":[33.6527,-117.83975]},"stt_when":{"
    "datetime":1456900106257},"stt_what":{"media_source":{"value":"https://d3j
    4aoik7k8ki7.cloudfront.net/747d700c-3a01-4881-b0b1-e758871af6e6.jpeg"},"
    caption":{"value":"Yawn, \n At 930 Bridge Rd"}},"stt_value":1}}},
14
15 {"cell":
16 {"rectangle":[{"point":[33.6375,-117.85001]},{"point":[33.6500000000,-117.8375
    1]}]}},
17 "count":4.0,
18 "values":{"orderedlist":[
19 {"stt_id":"3TslWDtD4f","stt_where":{"point":[33.644398,-117.8419]},"stt_when":
    {"datetime":1456865899607},"stt_what":{"media_source":{"value":"https://d3
    j4aoik7k8ki7.cloudfront.net/2e481dbb-c4d0-46ce-ba22-f7e52121eb2f.jpeg"},"
    caption":{"value":"Yawn, \n At Inner Ring Rd"}},"stt_value":1},
20 {"stt_id":"aEl2Tzao77","stt_where":{"point":[33.644398,-117.8419]},"stt_when":
    {"datetime":1456865899608},"stt_what":{"media_source":{"value":"https://d3
    j4aoik7k8ki7.cloudfront.net/98478efc-6a7e-4ac5-be45-47b6d8e0a422.jpeg"},"
    caption":{"value":"Yawn, \n At Inner Ring Rd"}},"stt_value":1},
21 {"stt_id":"Z2qS21EboS","stt_where":{"point":[33.648544,-117.83829]},"stt_when"
    :{"datetime":1457307672810},"stt_what":{"media_source_photo":{"value":"
    https://d3j4aoik7k8ki7.cloudfront.net/671fe2cc-8e33-4e37-910d-e414838e4fff
    .jpeg"},"caption":{"value":"Yawn, \n At 4065-4069 Campus Dr,"}}},"stt_value
    ":1},
22 {"stt_id":"ngx8T50P8U","stt_where":{"point":[33.64437,-117.84186]},"stt_when":
    {"datetime":1457473120434},"stt_what":{"media_source_photo":{"value":"
    https://d3j4aoik7k8ki7.cloudfront.net/e71ac5b5-b4a6-44c3-9657-ede1a7bc2d3c
    .jpeg"},"caption":{"value":"Yawn, \n At Inner Ring Rd,"}}},"stt_value":1}}]}

```