UNIVERSITY OF CALIFORNIA,
IRVINE

Personalized Situation Recognition

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Vivek Kumar Singh

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

2012

DEDICATION

To my great-grandfather Shri Parshadi Lal.

TABLE OF CONTENTS

Page

LIST OF FIGURES **vi**

LIST OF TABLES **ix**

ACKNOWLEDGMENTS **x**

CURRICULUM VITAE **xi**

ABSTRACT OF THE DISSERTATION **xv**

1 Introduction **1**
- 1.1 The emerging eco-system and a motivating application 2
  - 1.1.1 Motivating application: Flu risk based recommendations . . . 4
- 1.2 Difficulties in handling situations 5
- 1.3 Contributions 8
- 1.4 Outline of the thesis 10

2 Understanding and using Situations **12**
- 2.1 Defining *situations* 12
  - 2.1.1 Previous definitions 12
  - 2.1.2 Proposed definition 17
- 2.2 Problem of situation recognition 18
- 2.3 Situation aware applications 20
- 2.4 Design goals for framework to build situation-aware applications . . . 26
  - 2.4.1 Expressive power 27
  - 2.4.2 Lower the floor 27
  - 2.4.3 Raise the ceiling 27
- 2.5 Components required for the framework 28
  - 2.5.1 The building blocks 28
  - 2.5.2 Modeling approach 29
  - 2.5.3 Rapid prototyping toolkit 29

3 Related Work **30**
- 3.1 Situation awareness across research areas 30
- 3.2 Progress in the field of 'concept recognition' from multimedia data . . 37

3.3 Toolkit support for situations, context, and data analytic applications 39

4 Overall Framework 42
4.1 Design features of the framework . . . 43
4.2 Situation Modeling . . . 45
4.3 Situation Recognition . . . 46
4.3.1 Data stream selection . . . 47
4.3.2 Data ingestion . . . 47
4.3.3 Data unification . . . 47
4.3.4 Spatiotemporal Aggregation . . . 48
4.3.5 Situation evaluation . . . 50
4.4 Situation visualization, personalization, and alerts . . . 50
4.4.1 Situation visualization . . . 50
4.4.2 Personalization and Alerts . . . 52

5 Situation Modeling 53
5.1 Operators and Operands . . . 54
5.1.1 Operands . . . 55
5.1.2 Operators . . . 56
5.2 The Wizard for modeling situations . . . 58
5.3 Enhancing and Instantiating the model . . . 60
5.3.1 Refining the model . . . 60
5.3.2 Instantiating the model . . . 60
5.4 Example: Modeling Epidemic Outbreaks . . . 61

6 Data Representation and Situation Recognition Algebra 65
6.1 Data representation . . . 66
6.1.1 Data unification . . . 66
6.1.2 Spatiotemporal aggregation . . . 68
6.2 Analysis operators (Situation recognition algebra) . . . 69
6.2.1 Filter (Π) . . . 69
6.2.2 Aggregation ($\oplus$) . . . 74
6.2.3 Classification (γ) . . . 76
6.2.4 Characterization (@) . . . 77
6.2.5 Pattern Matching (ψ) . . . 80
6.2.6 Combining operators to create composite queries . . . 85

7 EventShop: Toward Interactive Situation Recognition 87
7.1 System design: overview . . . 90
7.2 Data Ingestor . . . 92
7.2.1 Data Sources . . . 92
7.2.2 Iterators . . . 95
7.2.3 Handling different types of data . . . 97
7.3 Stream Query Processor . . . 100
7.3.1 Query . . . 101

7.3.2 Operators . . . 103
7.4 Presentation of results . . . 110
7.5 Discussion . . . 111

8 Personalization and Alerts 113
8.1 Personalized Situations . . . 114
8.1.1 Data types and representation . . . 115
8.1.2 Operators . . . 116
8.1.3 Modeling Personalized Situations . . . 118
8.2 Alerts . . . 119
8.2.1 Data types . . . 120
8.2.2 Situation-Action Rules . . . 120
8.3 EventShop support for personalized alerts . . . 122
8.4 Example: Asthma/ allergy recommendation system . . . 123
8.4.1 Defining macro situation . . . 124
8.4.2 Defining personalized situation . . . 125
8.4.3 Situation-action rules . . . 125

9 Evaluations 127
9.1 Validating the design principles . . . 128
9.1.1 Using humans as sensors . . . 128
9.1.2 Space and time semantics of social media data . . . 129
9.2 Validating the data representation and analysis operations . . . 131
9.2.1 Application: Business analysis . . . 132
9.2.2 Application: Political event analytics . . . 135
9.2.3 Application: Seasonal characteristics analysis . . . 136
9.3 Building multiple applications using the framework . . . 140
9.3.1 Flood evacuation in Thailand . . . 140
9.3.2 Wildfire recognition in California . . . 145
9.3.3 Hurricane monitoring . . . 148
9.3.4 Flu epidemic monitoring and recommendation . . . 152
9.3.5 Asthma/Allergy recommendation system . . . 153
9.3.6 Discussion and looking back at the design goals . . . 158

10 Conclusions and Future Work 163

Bibliography 167

LIST OF FIGURES

Page

1.1 The emerging eco-system 3

3.1 Different Types of Concepts can be recognized in different data availability settings. Single media, such as images, results in concepts more in images than in the real world, but using different media it is possible to recognize concepts in the real world 38

4.1 Different phases in the framework: Situation modeling, recognition, and alerts 43
4.2 An E-mage showing user interest across mainland US in terms of number of tweets containing the term iphone on 11th Jun 2009 48
4.3 Different operators for situation recognition 51

5.1 Operands for Situation Modeling 55
5.2 Operators for Situation Modeling 56
5.3 Steps in Situation Modeling 59
5.4 Recursive approach for defining situation variables 59
5.5 Base model created for epidemic outbreaks 62
5.6 Situation model: Changes made in refinement phase 63
5.7 Situation model after the instantiation phase (details added in Red) . 64

6.1 The workflow of data from raw streams to situation descriptors . . . 66
6.2 Operator syntax format 72
6.3 Example: Filtering operation based on spatial predicate 73
6.4 Example: Filtering operation based on value predicate 73
6.5 Example: Filtering operation to normalize values to a range 74
6.6 Example: Aggregate operation on two streams using add function . . 75
6.7 Example: Classification operation based on linear thresholding 77
6.8 Example: Characterization operation based on spatial epicenter . . . 80
6.9 Example: Characterization operation based on average speed over 3 cycles 81
6.10 Example: Pattern matching operation based on a spatial pattern . . . 83
6.11 Example: Pattern matching operation using on a temporal pattern . 84

7.1 Screenshot of EventShop system 89
7.2 System Architecture of EventShop 91

7.3 System Architecture of Data Ingestor 93
7.4 System Architecture of Query Processor 101
7.5 Operator tree for a situation query 103
7.6 Mapping of Situation recognition Algebra to Media Processing Operations 105
7.7 Configuration Options for different operators in EventShop 106

8.1 Approach to recognizing macro situations, personalized situations, and sending alerts 114
8.2 Steps in personal situation modeling 118
8.3 A snapshot of the Personalized Alert Unit 122
8.4 Situation Model for Asthma threat level 124
8.5 Situation Model for personal threat level 125

9.1 Real World Events and their recognition via human sensor reports . . 129
9.2 Variation of frequency of hashtags and their ranks for different time durations 130
9.3 Variation of frequency of hashtags and their ranks for different geo-regions 130
9.4 Sample results of applying the queries 1-3, for iphone theme 133
9.5 Combination of operators for undertaking a business decision 134
9.6 Sample result for running query 3 for political theme I='Healthcare', and politician P='Obama' 135
9.7 Spatio-(temporal) E-mages representing average of Flickr images posted from each location across months in 2010. 137
9.8 Snow themed E-mages through the year based on number of images tagged with 'snow' coming from each location 137
9.9 Movement of greenery zones (brightest= 'most green') across US through the year 138
9.10 Answers for queries 1-2 for seasonal characteristics monitoring using Flickr data 138
9.11 Answers for queries 3 and 4 for seasonal characteristics monitoring using Flickr data 138
9.12 Situation Model for flood risk level 141
9.13 Situation Model for personal safety concern 142
9.14 Sample E-mages 143
9.15 Parameters for the data sources configured in the Flood Recommendation Application 143
9.16 Resultant situation classification and Personalized Alert configuration 144
9.17 Sample tweets sent out to users 144
9.18 'Wildfire' recognition model using satellite data 145
9.19 Recognition performance of satellite data based detector for Wildfires across California over last 2 years 146
9.20 'Wildfire' recognition model using social data 147
9.21 'Wildfire' recognition model using satellite + social data 148

9.22 Recognition performance of satellite + social detector for Wildfires across California over last 2 years 148
9.23 Wildfire recognition results over different timeframes 149
9.24 Model for movement patterns in hurricanes 150
9.25 Parameters for simulated hurricane data source 150
9.26 Hurricane Application: Spatial pattern matching 151
9.27 Hurricane Application: Temporal pattern matching 152
9.28 Situation Model for Epidemic Outbreaks 153
9.29 E-mage for (a) Reports on Flu (brighter indicates more reports), (b) Historical average, (c) Population 154
9.30 'Epidemic outbreak risk level 155
9.31 Situation Model for asthma threat level 155
9.32 Situation Model for personal threat level 156
9.33 Parameters for the data sources configured in the Allergy Risk Recommendation Application 157
9.34 Asthma Application: Sample E-mages 157
9.35 Sample result 157
9.36 Parameters for the Personal activity level data source 158
9.37 Sample Tweets sent out from EventShop 158
9.38 Summary of different applications discussed in this dissertation 160

LIST OF TABLES

Page

2.1 Survey of Situation definitions. Note: '○' indicates partial support. . 16
2.2 Survey of situation aware applications and their characteristics. . . . 25

3.1 Design goal of the proposed work 38
3.2 Survey of various related research areas and their characteristics. Legend: X = Well supported, 'o' = Partial support. 41

6.1 Terminology used for data representation and operators 70
6.2 Summary of various query operations. TES=Temporal E-mage Stream, TPS=Temporal Pixel Stream 84

7.1 Output formats for different types of queries 111

8.1 Examples of different data types 116
8.2 Summary of various query operations. TES=Temporal E-mage Stream, TPS=Temporal Pixel Stream 118

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Professor Ramesh Jain for the guidance and support he has given me over the years. Each day I strive to learn a bit more from him on how to become a better researcher, and a better person. It has been a rare privilege to learn from a man so wise, so bold, and yet so humble.

Next, I would like to thank my thesis committee members, Michael Carey and Gopi Meenakshisundaram, for their support and multiple helpful suggestions on my dissertation.

I would also like to thank my master's thesis advisor Mohan Kankanhalli, who introduced me the joys of conducting research, and Pradeep Atrey, who mentored me through the initial phases. I am also very grateful to Jiebo Luo and Dhiraj Joshi for introducing me to the world of industrial research.

I would like to thank my ESL lab mates (Ish, Hamed, Mingyan, Siripen, Setareh, Pinaki) for numerous discussions and collaborations. My special thanks go to Mingyan Gao for being such a wonderful collaborator. I would also like to thank Yufang Jin, from UCI Earth Science department who has helped me with multiple experiments related to this work.

I am also indebted to the UCI Bren Chair's funds which provided financial support while I worked on my dissertation.

Good friends are difficult to come by. I would like to thank my friends at IKG (Irvine Kay Goondey) for the spirited humor and lighter moments in life which kept me sane during the highs and lows of grad school life. My very special thanks go to Nilopa, Avi, Becca, and Vrishti, for their friendship and support throughout this phase of my life.

Finally, I owe everything to my family (my parents Dr Siyaram Singh, Urmila, and my siblings Abhishek, Kinshuk, and Vidushi) for placing me on this path in life. It is only due to their affection, hard work, and sacrifices, that I have come this far.

CURRICULUM VITAE

Vivek Kumar Singh

EDUCATION

Doctor of Philosophy in Computer Science **2012**
University of California, Irvine *Irvine, California*

Master of Computing **2005**
National University of Singapore *Singapore*

Bachelor of Engineering in Computer Engineering **2002**
National University of Singapore *Singapore*

RESEARCH EXPERIENCE

Graduate Research Assistant **2007–2012**
University of California, Irvine *Irvine, California*

Research Assistant **2006–2007**
National University of Singapore *Singapore*

TEACHING EXPERIENCE

Teaching Assistant / Reader **2007–2009**
University of California, Irvine *Irvine, California*

Lecturer **2002–2006**
Institute of Technical Education *Singapore*

SELECTED HONORS AND AWARDS

Selected as an 'Emerging Leader in Multimedia Research' **2009**
IBM Research Labs, NY

Best Paper Award **2009**
ACM Workshop on Social Media

Best Student Paper **2009**
IEEE Workshop on Situation Modeling

REFEREED JOURNAL PUBLICATIONS / BOOK CHAPTERS

Mechanism Design for Incentivizing Social Media Contributions **2011**
V. K. Singh, R. Jain and M. Kankanhalli, in Social Media and Modeling (Eds. Hoi, Luo, Boll, Xu, Jin, King), Springer-Verlag London Limited.

Towards Environment to Environment (E2E) multimedia communication systems **2009**
V. K. Singh, H. Pirsiavash, I. Rishabh, and R. Jain, in Multimedia Tools and Applications.

Adversary Aware Surveillance Systems **2009**
V. K. Singh, and M. Kankanhalli, in IEEE Transactions on Information Forensics and Security.

Coopetitive multi-camera surveillance using Model Predictive Control **2008**
V. K. Singh, P.K. Atrey and M. Kankanhalli, in Journal of Machine Vision and Applications.

REFEREED CONFERENCE PUBLICATIONS

Situation Recognition: An Evolving Problem for Heterogeneous Dynamic Big Multimedia Data **Oct. 2012**
V.K. Singh, M.Gao, and R. Jain, in *to appear in* ACM Multimedia Conference.

EventShop: From Heterogeneous Web Streams to Personalized Situation Detection and Control. **Jun. 2012**
M.Gao, V.K. Singh, and R. Jain, in ACM Web Science Conference.

Reliving On Demand: A Total Viewer Experience **Nov. 2011**
V.K. Singh, J.Luo, D. Joshi, M. Das, P. Stubler, and P. Lei, in ACM Multimedia Conference.

From Multimedia Data to Situation Detection **Nov. 2011**
V.K. Singh in *(Doctoral Symposium)* ACM Multimedia Conference.

Social Pixels: Genesis and Evaluation **Nov. 2011**
V.K. Singh, M.Gao, and R. Jain, in ACM Multimedia Conference.

Event Analytics on Microblogs **Apr. 2010**
V.K. Singh, M.Gao, and R. Jain, in Web Science Conference.

Structural Analysis of Emerging Event-Web **Apr. 2010**
V.K. Singh, and R. Jain, in IW3C2 World Wide Web Conference.

Multimodal Observation Systems **Oct. 2008**
M.K. Saini, V.K. Singh, R. Jain, and M. Kankanhalli, in ACM Multimedia Conference.

Adversary Aware Surveillance Systems **Jun. 2007**
V.K. Singh, and M. Kankanhalli, in IEEE Conference on Multimedia and Expo.

Coopetitive Multimedia Surveillance **Jan. 2007**
V.K. Singh, P. K. Atrey, and M. Kankanhalli, in International Multimedia Modeling Conference.

SELECTED WORKSHOP PUBLICATIONS

Social Life Networks for the Middle of the Pyramid. **Apr. 2011**
R. Jain, V.K. Singh, and M.Gao, in ACM Workshop on Social Media Engagement.

Motivating Contributors in Social Media Networks **Oct. 2009**
V.K. Singh, M. Kankanhalli, and R. Jain, in ACM Workshop on Social Media.

Toward Environment-to-Environment (E2E) Affective Sensitive Communication System **Oct. 2009**
M. Paleari, V.K. Singh, B. Huet, and R. Jain, in ACM Workshop on Multimedia Technologies for Distance Learning.

Situation-based-control for Cyber Physical Environments **Oct. 2009**
V.K. Singh, and R. Jain, in IEEE Workshop on Situation Modeling.

Towards Environment to Environment (E2E) multimedia communication systems **Oct. 2008**
V.K. Singh, H. Pirsiavash, I. Rishabh, and R. Jain, in ACM Workshop on Semantic Ambient Media.

A Design Methodology for Selection and Placement of Sensors in Multimedia Surveillance Systems **Oct. 2007**
G. Sivaram, K.R. Ramakrishnan, P.K. Atrey, V.K. Singh, and M. Kankanhalli, in ACM Workshop on Video Surveillance and Sensor Networks.

Coopetitive Visual Surveillance using Model Predictive Control **Oct. 2006**
V.K. Singh, P.K. Atrey, and M. Kankanhalli, in ACM Workshop on Video Surveillance and Sensor Networks.

ABSTRACT OF THE DISSERTATION

Personalized Situation Recognition

By

Vivek Kumar Singh

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 2012

Professor Ramesh Jain, Chair

With the growth in internet-of-things, social media, mobile devices, and planetary-scale sensing, there is an unprecedented opportunity to assimilate spatio-temporally distributed streams into actionable situations. Detecting situations in realtime can be used to benefit human lives and resources in multiple applications. However, the progress in the field of situation recognition, is still sluggish because: (a) the notion of situations is still vague and ill-defined, (b) there is a lack of abstractions and techniques to help users model their situations of interest, and (c) there is a lack of computational tools to rapidly implement, refine, and personalize these situation models to build various situation-based applications.

This dissertation computationally defines *situations* and presents a framework for personalized situation recognition by providing support for conceptual situation modeling, data unification, real-time situation recognition, personalization, and action-taking.

The proposed framework defines a situation as "An actionable abstraction of observed spatio-temporal descriptors", and identifies a data representation, a set of analysis operations, and lays out a workflow for modeling different situations of interest. Considering Space and Time as the unifying axes, it represents data in a grid-based *E*-

mage data structure. It defines an algebra of operations (*viz. Selection, Aggregation, Classification, Spatio-temporal Characterization,* and *Spatio-temporal Pattern Matching*) for situation recognition; and defines a step-by-step guide to help domain experts model their situations based on the data, the operations, and the transformations.

The framework is operationalized via EventShop - a web based system which lets users graphically select, import, combine, and operate, on real-time data streams to recognize situations for generating appropriate information and actions. EventShop allows different designers to quickly configure their situation models, evaluate the results, and refine the models until a satisfactory level of performance for supporting various applications is achieved. The detected situations can also be personalized and used for undertaking control actions via Situation-Action rule templates.

The framework has been used to build multiple applications including flu monitoring and alerting, wildfire recognition, business decision making, flood alerts, asthma recommendation system, seasonal characteristics analysis, and hurricane monitoring.

Thesis statement:

We computationally define the notion of situations and present a framework for personalized situation recognition by providing support for conceptual situation modeling, data unification, real-time situation evaluation, personalization, and action-taking.

Chapter 1

Introduction

We are living in an 'age of abundance' [30]. Humanity is more connected than ever before, and there is an unparalled availability of data and computing resources. With the growth trends in social media, multimodal mobile sensing, and location driven sensing, increasingly larger parts of human life are getting digitized and becoming available for sense making. Real world phenomena are now being observed by multiple media streams, each complementing the other in terms of data characteristics, observed features, perspectives, and vantage points. Many of these multimedia streams are now available in real-time and increasingly larger portion of these come inscribed with space and time semantics. The number of such media elements available (e.g. Tweets, Flickr posts, sensor updates) is already in the order of trillions [89], and computing resources required for analyzing them are becoming increasingly available. We expect this trend to continue, and even get accelerated with the growing adoption of mobile devices.

These data now provide an opportunity for their aggregation and composition to recognize the evolving situations in all parts of the world. The situational information

derived can be used to provide information, answer queries, and also take control actions. Providing tools to make this process easy and accessible to the masses will impact multiple human activities including traffic, health, business analysis, political campaign management, cyber security monitoring, disaster response, crisis mitigation, and homeland security.

However, the progress in generating actionable insights from diverse data streams is still slow and the field of situation-based-computing in its infancy. This dissertation identifies and tackles some of the important challenges in the field of situation recognition. Specifically, it computationally defines the notion of situations and presents a framework for personalized situation recognition that provides support for conceptual situation modeling, data unification, real-time situation evaluation, personalization, and action-taking.

We motivate the work based on the emerging eco-system, and the type of problems that is becoming increasingly relevant.

1.1 The emerging eco-system and a motivating application

As shown in Figure 1.1, the Cloud today connects and contains a variety of data streams related to multiple human functions like traffic, weather, and health. These data are in archived databases as well as real-time streams reporting attributes from different parts of the world. The real-time streams originate either from the traditional sensor/device based sources (e.g. PlanetarySkin, Satellite imagery), or the increasingly common human reporting mechanisms (e.g. Twitter and Facebook status updates). All these data can be aggregated in the Cloud and used for situation

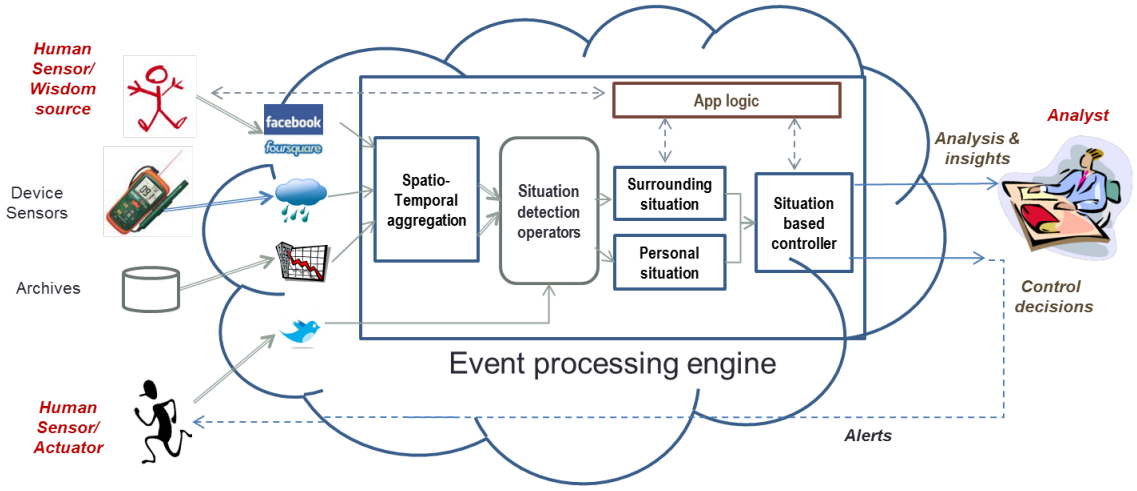

Figure 1.1: The emerging eco-system

recognition and action taking.

Humans play multiple roles in this eco-system. ***human sensors*** can describe different aspects of a situation, many of which are not yet measurable by any hardware sensors. Millions of users are already acting as ***human actuators*** which are getting daily alerts, advices, and recommendations for undertaking different actions. This trend will only increase [89, 30]. As envisioned in the 'wisdom of the crowds' [60] concept - or Wikipedia as a system - different users can act as ***wisdom sources*** and work towards completing different tasks including the configuration of applications for different situations [97]. Lastly, the centralized ***Analysts*** can visualize, and analyze different situations to undertake important macro-decisions affecting their country, state, county, or corporation.

Together this eco-system allows for the creation of unprecedented data as well as collective intelligence for supporting multiple applications. Let us consider one representative application in this eco-system.

1.1.1 Motivating application: Flu risk based recommendations

Alice, a part-time *Ushahidi.com* volunteer and full time mother is worried about the evolving Flu Epidemic situation. She wants to create an evolving map of Flu Epidemic risk for people across the United States. Not knowing how to define Epidemic risk, she approaches her friend Bob, who is an expert on communicable diseases. He explains that Flu Epidemic risk depends on multiple factors both personal and environmental. As a first step he advises her to focus on three environmental factors viz. temperature, nearby polluting factories, and the number of active flu cases reported. He advises her that the weather data can be obtained from a US Geological Services website, the number of active cases can be obtained from Twitter, and nearby factory data from *www.epa.gov*. He helps her create a conceptual blueprint of how these data values can be combined to classify the US into zones of low, mid, and high epidemic risk.

Alice uses the blueprint 'situation model' as a guide, and configures a mashup application which combines the different data streams and creates a heat-map representing flu epidemic risk in different parts of the US. Continuously being updated with real-time data, the visualization gives an intuitive snapshot of the flu situation evolving across US.

Charlie sees this map on *Ushahidi.com*, and decides to take this one step further by *personalizing* the situation recognition. He defines individual's personal risk level based on rules which combine the individual parameters (e.g. exertion level, sneezing frequency, temperature) for any person with the corresponding risk level in her surroundings. He defines action rules which urge the most vulnerable people to visit doctors; advise potentially vulnerable people to avoid exertion, and prompt users in healthy environments to enjoy the outdoors (e.g. "go jogging at the nearest park").

Thousands of users get this information, and based on verifying the data behind the suggestion, many decide to dust off their track pants and go for a run.

1.2 Difficulties in handling situations

Despite multiple recent efforts to understand and use situations, building an application like the one described here is still a hard and lengthy process (if not completely infeasible). This is due to multiple reasons, including the lack of understanding and operationalization of the concept of situations, heterogeneity of the data involved, real-time processing requirements, lack of geo-spatial data and abstractions, and dearth of computational infrastructure support. Let us consider some of these problems here:

Concept of situation is ill defined

Everybody understands and interprets situations in their own way (see Chapter 2 for a detailed discussion), and typically does so with a perspective of solving one problem – the one being encountered by the designers at that time. Lack of agreement on the semantics of situations as a concept is a fundamental problem that affects progress in each of the related areas. The absence of a standardized understanding results in fragmented efforts, which are not reusable, lack clear planning, and often fall short in terms of design maturity [27]. Application developers choose whichever technique is easiest to implement, at the expense of generality and reuse. This approach entails a number of risks. First, an availability-driven approach constrains the possibilities for designers by limiting the kind of applications they are able to design and the usages they can propose. Second, the details and shortcomings of the model may be carried

up to the application level and affect its usefulness and ability to continuously evolve.

Conceptual situation modeling: abstractions required

Currently, there are no tools to explicitly describe what the designer means by a particular situation e.g. "Allergy Outbreak". To recognize a concept one must 1) have an internal model of what it means, and 2) be able to externalize it using some constructs. Lack of such tools often implies that the situation models and data used in applications are driven by the acquisition mechanisms available.

Data representation, unification, and processing

Practical problems (like the one discussed) require a combination of information from different sources. This requires heterogeneous data to be converted into a common representation that is generic and does not need be redefined for every new data source selected. Furthermore the representation needs to capture enough semantic and computational detail so that it can support a variety of situation recognition tasks. For example, the discussed flu risk application needs a method to combine the data coming from Twitter stream, satellite, and pollution neighborhood. The data representation needs to capture the spatial semantics like neighborhood (e.g. to define the pollution effect of factories) and geography-driven-joins (e.g. for overlaying of data grids).

Situation evaluation at scale

Detecting situations involving data coming from all parts of the world requires scalable systems which can seamlessly handle huge volumes of data. Further, doing this

analysis in real-time to support situation based actions is hard problem. Currently there are no systems which can seamlessly handle data integration, indexing, and analysis for web-scale spatio-temporal data.

From situations to personalized alerts

As we saw in the example, situations need to be recognized both at the personal level and the macro level. Traditional situation recognition has focused on single large scale (e.g. over city, state, country) insights. The decisions once made were *broadcasted*. This was true from health warnings, to weather alerts, to advertisements. Today, we need tools to individually access each user's inputs and combine them with the surrounding situation recognized around her. Thus allowing each user to get a *personalized (unicast)* alert based on specific situation recognized for her. Tools which seamlessly support transition across these two very different levels (macro and personal) are not yet available.

Rapid implementation, validation, refinement, and deployment

As with any new problem, and more crucially so when trying to reach out to application designers with web-scale variations in the level of design experience, iterative development is key to creating usable situation-aware applications. Thus, applications need to be implemented in a way that makes it easy to modify different components. There are currently few guidelines, models, or engines that support this requirement.

1.3 Contributions

This dissertation contributes towards the problem of situation recognition. The main contributions are:

1. To computationally define the notion of 'situations'.
2. To describe a generic process to select, import, combine, and operate on real-time data streams to recognize personalized situations for generating appropriate information and actions.
3. To provide a Situation modeling kit which helps the application designers translate their mental models of situations into explicit, actionable, and computable modules.
4. To define a unified representation (E-mage) and situation recognition algebra for diverse Spatio-temporal data.
5. To provide a web based system (EventShop) that allows rapid validation and refinement of situation models to create multiple situation-aware applications.
6. To provide tools for personalization of situations to support action-taking.

We define a situation as: "*An actionable abstraction of observed spatio-temporal descriptors*". This definition emphasizes the characterization of situations based on measurable spatio-temporal descriptors. Focus on spatio-temporal data (which is the most common connotation associated with situations), scoping of problem only to observable data, and an emphasis on actionable abstractions (as defined explicitly by human domain experts) allows development of a computational framework to define diverse situations and take appropriate actions.

The framework focuses on the identifying abstractions required for modeling and recognizing situations. These abstractions are identified based on a survey of situation-based applications, their requirements, and a specific focus on spatiotemporal data. The focus on spatio-temporal draws upon the basic nature of the physical world, i.e. space and time are the independent axes which unify all physical phenomena. To remain computationally grounded, this framework assumes that combining and evaluating a large number of data streams measured across space and time can be used to determine situations. This means that an engine that is programmed using operators to detect features over vast number of data streams can be used to define and recognize[1] any arbitrarily complex situation.

The framework is instantiated via EventShop, a web based system that supports rapid validation and refinement of the generated models. It allows different users to quickly implement their situation models, evaluate the results, and refine the models until a satisfactory level of application performance is achieved.

The personalization of these situations is undertaken by defining the personal parameters required to make the situations relevant to any given user. The action-taking aspect is handled via Situation-Action rules, which allow combination of personal parameters and macro-situations to undertake different actions (e.g. sending out tweets to users in risky situations).

Identifying the requirements to support the building and evolution of situation-aware applications, results in a conceptual framework that both 'lowers the floor' [75] (i.e. makes it easier for designers to recognize situations) and 'raises the ceiling' (i.e. increases the ability of designers to build more sophisticated detectors).

[1]In this work we do not differentiate between *recognition* and *detection*

We demonstrate the usefulness of the approach via multiple applications including flu monitoring and alerting, business decision making, flood alerts, asthma recommendation system, seasonal characteristics analysis, and hurricane monitoring.

To the best of our knowledge, this work is the first systematic attempt toward recognizing situations in the context of large scale spatio-temporal multimedia streams. Tackling the related challenges implied a need for a collaborative effort coming from data stream processing and media analysis perspectives. Hence parts of this work were undertaken in collaboration with [40]. The principal areas of collaboration were defining the situation recognition algebra, and implementation of the EventShop system. The emphasis of this dissertation is on computationally defining situations and presenting a method for conceptual modeling of situations, data unification, real-time situation recognition, personalization, and action-taking. This also implies that issues like infra-structure support for handling web-scale data, and user experience design are beyond the scope of this dissertation. Lastly, the implemented toolkit (EventShop) is intended to be an evolving, extensible, open-source project. Specific features and options available are extensible, and will continue to evolve.

1.4 Outline of the thesis

Chapter 2 discusses the notion of 'situations' as broadly understood across different research fields and provides an operational definition for it. It also surveys a breadth of situation aware applications and identifies the functional requirements for a framework supporting them. Chapter 3 discusses the related work and scopes the focus of this dissertation. Chapter 4 describes a generic framework to recognize situations from heterogeneous streams. Chapter 5 describes a generic approach for situation modeling. Chapter 6 describes the data representation (E-mage Streams), and op-

erators defined to evaluate situations. Chapter 7 discusses how a graphically configurable tool (EventShop) can be used to easily prototype situation models. Chapter 8 discusses how situations can be personalized and used to send out alerts to large number of users. Chapter 9 discusses multiple applications which were created using the described framework and their comparison to ground truth where applicable. It rounds off with a discussion of the proposed framework at meeting the design goals identified in Chapter 2. Chapter 10 discusses the future challenges and concludes this dissertation.

Chapter 2

Understanding and using Situations

2.1 Defining *situations*

2.1.1 Previous definitions

Situations have been studied across multiple research areas like ubiquitous/pervasive computing [114, 104], building automation [29], mobile application software [108], aviation/air traffic control [37, 2], robotics [84, 64], industrial control [80], military command and control [103], surveillance [17], linguistics [13], stock market databases [51, 3], and multimodal presentation [76], under the garbs of situation modeling, situation awareness, situation calculus, situation control, and situation semantics. The interpretation of situation however is different across different areas and even across different works within the same area. Here we sample some of the situation definitions employed:

- *Endsley, 1988: "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future"[37]*
- *Mooray, 2005: "is a shorthand description for keeping track of what is going on around you in a complex, dynamic environment"[73]*
- *Adam, 1993: "knowing what is going on so you can figure out what to do"[2]*
- *Jeannot, Kelly, & Thompson, 2003: "what you need to know not to be surprised"[57]*
- *McCarthy, 1968: "A situation is a finite sequence of actions."[70]*
- *Yau, 2006: "A situation is a set of contexts in the application over a period of time that affects future system behavior"[114]*
- *Barwise and Perry, 1980: "The world consists not just of objects, or of objects, properties and relations, but of objects having properties and standing in relations to one another. And there are parts of the world, clearly recognized (although not precisely individuated) in common sense and human language. These parts of the world are called situations. Events and episodes are situations in time, scenes are visually perceived situations, changes are sequences of situations, and facts are situations enriched (or polluted) by language."[12]*
- *Dietrich, 2004: "...extensive information about the environment to be collected from all sensors independent of their interface technology. Data is transformed into abstract symbols. A combination of symbols leads to representation of current situations ... which can be detected"[29]*
- *Sarter & Woods, 1991: "accessibility of a comprehensive and coherent situation representation which is continuously being updated in accordance with the results of recurrent situation assessments"[88]*
- *Dominguez, Vidulich, Vogel, & McMillan, 1994: "the continuous extraction of environmental information along with integration of this information with previous knowl-*

edge to form a coherent mental picture, and the end use of that mental picture in directing further perception and anticipating future need"[31]

- *Smith & Hancock, 1995: "adaptive, externally-directed consciousness that has as its products knowledge about a dynamic task environment and directed action within that environment"[100]*
- *Dostal, 2007: "the ability to maintain a constant, clear mental picture of relevant information and the tactical situation including friendly and threat situations as well as terrain"[32]*
- *Merriam-Webster dictionary: "relative position or combination of circumstances at a certain moment"[71]*
- *Singh & Jain 2009: "the set of necessary and sufficient world descriptors to decide the control output"[95]*
- *Steinberg, 1999: Situation Assessment is "the estimation and prediction of relations among entities, to include force structure and cross force relations, communications and perceptual influences, physical context, etc"[103]*
- *Dousson, 1993: "set of event patterns and a set of constraints"[33]*

We clearly see some common traits as well as the dissimilarities amongst different definitions. Most telling perhaps is the observation by Jakobson et al that *" ... being a relatively new field, there is a clear lack of theoretic well-grounded common definitions, which may be useful across different domains."* [56].

We decided here to focus on the commonalities across definitions, and identified the following notions to reverberate across definitions:

1. **Goal based (GB):** Situations need to be defined for an application or a purpose.

2. **Space and time (ST):** capture and represent a volume of space and/or time.

3. **Future actions (FA):** support future prediction and/or action taking

4. **Abstraction (AB):** some form of perception, or symbolic representation for higher cognitive understanding.

Further while some definitions were **computationally grounded (CG)** in data (e.g. Endsley, Dietirch), others were abstract (e.g. Barwise, Merriam-Webster). We summarize the definitions based on these axes in the Table 2.1 .

Work	Goal Based	Space Time	Future Actions	Abstraction	Computationally Grounded
Endsley, 1988 [37]:		X	X	X	X
Mooray, 2005 [73]:		o		X	
Adam, 1993 [2]:	X		X		
Jeannot, 2003 [57]:	X				
McCarthy, 1969 [70]:			X		
Yau, 2006 [114]:	X		X		X
Barwise, 1971 [12]:		X		X	
Dietrich, 2003 [29]:				X	X
Sarter, 1991 [88]:		o		X	
Dominguez,1994 [31]:	X		X	X	X
Smith, 1995 [100]:	X	o	X	X	
Dostal, 2007 [32]:		o		X	
Merriam-Webster [71]:		o			
Singh, 2009 [95]:	X		X		X
Steinberg, 1999 [103]:	X		X	X	o
Dousson, 1993 [33]:		o	X	o	X

Table 2.1: Survey of Situation definitions. Note: 'o' indicates partial support.

2.1.2 Proposed definition

Based on observing the common traits as well as a focus on staying computationally grounded, we define a situation as:

"An actionable abstraction of observed spatio-temporal descriptors."

Going right to left, let us consider each of the terms used in this definition:

- **descriptors:** This follows the approach of quantifying an abstract /inexact notion based on sampling its characteristics [34, 77].
- **spatio-temporal:** The most common connotation associated with 'situations' (as well as this work's focus), is on spatio-temporal data.
- **observed:** As a computational concept the focus is only on the 'observable' part of the world. Meta-physical as well as physical aspects which cannot be measured by sensors are simply beyond its scope.
- **abstraction:** This signifies the need to represent information at a much higher level than sensor measurements or even their lower level derivations. Decision makers typically focus on higher (knowledge) level abstractions while ignoring the lower level details.
- **actionable:** The top level descriptors and abstractions need to be chosen based on the application domain, and the associated output state-space. Hence our focus is on creating a representation (e.g. classification) which maps the lower level details into one concrete output decision descriptor. Hence, we are not interested in *any* higher level abstraction, but rather the *specific* one which supports decision making in the application considered.

As can be noticed, this definition operationalizes the reverberating threads found

across different definitions in literature, and computationally grounds them.

2.2 Problem of situation recognition

As highlighted by the definition, the essential problem of situation recognition is that of obtaining actionable insights from observed spatio-temporal data. Just like any effort at concept recognition, this problem can be split into three phases viz. observing data, extracting features, and detecting concepts from the observed features. The unique nature of situation recognition problem is reflected in the spatio-temporal grounding of all data as well as features defined.

Data

Let us represent the data observed at spatio-temporal coordinate st about any particular theme θ as follows:

$$D_{st\theta} = \lambda(\theta, st) \tag{2.1}$$

where:
s represents the spatial coordinate of the observation, i.e. $s \in \Re^3$
t represents the temporal coordinate of the observation,
θ represents the application/sensor specific properties which are observed at the spatio-temporal coordinate, and
λ is the mapping function from the real world characteristics to the observation space.

Aggregating over space and time, the data about any particular theme can be referred to as $D_{ST\theta}$, and combining over all observed themes, the data D_{ST} can be represented

as:

$$D_{ST} = \{D_{ST\theta_1}, D_{ST\theta_2}, ..., D_{ST\theta_k}\} \tag{2.2}$$

Features

A spatio-temporal feature f_{ST} can be obtained via a function Ω applied on the observed data.

$$f_{ST} = \Omega(D_{ST}) \tag{2.3}$$

These features (e.g. growth-rates, geographical epicenters, raw values) capture different properties of the observed phenomena and are selected based on their ability to discriminate between the classes of interest.

The combination of features yields a feature-set F_{ST} represented as:

$$F_{ST} = \{f_{ST1}, f_{ST2}, , f_{STN}\} \tag{2.4}$$

Situations

Consequently, Situations can be derived via a function Ψ applied on the feature-set.

$$c_{ST} = \Psi(F_{ST}) \tag{2.5}$$

and $c_{ST} \in C$, where C is the situation universal set. It could be discrete classifications (focus of this work) or values in a certain range. Here:

$$C = \{c_1, c_2, ..., c_m\} \tag{2.6}$$

where c_1 through c_m are the admissible classes of situation.

To summarize, c_{ST} is the final spatio-temporal situation selected from the range of situations possible, obtained via function Ψ applied on the observed features, which in turn are obtained by applying a function Ω on the observed spatio-temporal data.

Thus the problem of situation recognition is to identify the right situation classification for a given set of observations i.e.

$$\Psi \circ \Omega : D_{ST} \rightarrow C \tag{2.7}$$

or alternatively:

$$c = \Psi(\Omega(D_{ST})) \tag{2.8}$$

This work aims to define a framework to tackle the situation recognition problem i.e. extract spatio-temporal features from the observed data, and use them for situation classification.

Note that in this work we will focus on 2 dimensions (latitudes and longitudes) for spatial coordinates i.e. $s \in \Re^2$, and consider the observed values be real numbers i.e. $D_{st\theta} \in \Re$.

2.3 Situation aware applications

To define a generic framework for building situation-aware applications, we first survey a variety of situation-aware applications – and identify their commonalities and differences.

Here we discuss 18 of the applications surveyed. These applications correspond to 6 each from three categories. First six are applications developed in-house by our team when we started exploring the notion of situation-awareness. Next 6 correspond to academic initiatives to solve situation related problems. Where needed, we focused on one specific application from large scale projects. The last 6 correspond to industrial or governmental efforts at building situation aware applications.

1) Flu monitoring and response: To monitor number of flu cases, and direct at-risk users to vaccination sites [92].

2) Political event analytics: To monitor changes in interest in different political figures, parties, and topics.

3) Business decision making: To identify the most suitable location for a new business store.

4) Allergy/Asthma recommendation: To monitor Allergy risk, and direct at-risk users to safe locations.

5) Seasonal pattern analysis: To monitor when the change in Fall colors occurs in New England.

6) Thailand flood response: To direct people stuck in unsafe areas to the nearest shelters.

7) City of Ontario disaster mitigation (RESCUE @ UCI): To provide information about open shelters, Spills, fire, earthquakes, and road-closure [9].

8) Raining cabs (Senseable Cities Lab @ MIT): Visualize the impact of Rain on cab wait times in Singapore [61].

9) Mapping ideas (Geography Deptt @ SDSU): To analyze traces of different ideas from cyberspace to the physical geography [102].

10) Earthquake detection using Twitter (@ Univ. of Tokyo): Using twitter feeds to detect Earthquakes and send out alerts [87].

11) Smart classrooms (@ASU): Student mobile devices react to the situation in the classroom to initiate connections and collaborations [113].

12) Military situation awareness (@ Univ. Bundeswehr - Munich): Assessment of risk at different locations on the battleground [66].

13) Plant hardiness zone map (@ US Deptt. Of Agriculture): To help gardeners and farmers identify where and when plants grow best by breaking up geographical areas into zones.

14) Nokia 'Situations': To change the settings (e.g. sounds, alerts) on a mobile phone based on the contextual parameters (e.g. time, location).

15) Breast cancer application (@VSolveIt): To identify disease hot-spots, and suggest clinical trial zones for breast cancer.

16) US drought impact reporter (@National Drought Mitigation Center): To provide daily drought risk assessments across United States.

17) Situation based industrial control (@Siemens): To monitor the state of different processes in an industrial application, and take actions based on the situations recognized.

18) Traffic condition reports (@Google): To classify freeways traffic conditions based on the speed and the volume.

All the applications follow a general pattern where they import one or more sets of data, do some value addition, and provide the output in different formats. We noticed the following general themes in each of the aspects:

1. **Input Data:** Variety of data being used.

    - **Source:** Sensor based or human report driven (S, H)

      The data used can be either sensor driven, human reports driven. We consider collected data (e.g. Census) to be human reports for this discussion.

    - **Timespan:** Real-time or Archived (R, A)

      The applications either continually evolved with new incoming data or used a snapshot of archived data. Very few works used both, but it was

seen.

- **Diversity:** Homogeneous data or heterogeneous data (H,T)

  The data could be homogeneous, or very disparate.

2. **Value addition:** Processing of data streams

   - **Integration:** Shallow or Deep (S or D)

     Many applications focused on visually placing different data types on a common format (typically map). These can be considered shallow mashups. While useful for information consumption, they do not integrate different sources to derive newer information [22]. Deep mashups on the other hand allow inter operation between different data types for defining newer variables which affect the situation.

   - **Operations used:** Arithmetic Logic, Value Projection, Nearest neighbors, Spatio-temporal properties (AL, VP, NN, P)

     We noticed 4 broad categories of operations. Those for doing arithmetic, or logical combination, Projection of values based on user location, identification of nearest neighbors, and spatio-temporal property extraction (e.g. peak, growth rate, volume).

3. **Output:** provided to the end users or analysts.

   - **Visualization:** Maps, Charts, Timelines, or Text (M, C, T, TX)

     The data could be presented in multiple formats. Maps were by far the most common visualization tool.

   - **Actions/ Alerts** (AC, AL)

     Potentially such systems could take actions on their owns, but we saw most large scale systems involving humans focus on alerts to them.

- **Querying** (Q)

  Many applications allowed users to query the system to identify values (typically for a particular location).

We summarize these observations in the Table 2.2.

Applications	Inputs			Value addition		Output		
	Source	Timespan	Diversity	Integration	Operations	Viz	Actions	Querying
1) Flu	H	R	H	D	AL, VP, NN, P	M, T	AL	
2) Political	H	R	H	D	AL, P	M, T		Q
3) Business	H	A	H	D	AL, VP, P	M		Q
4) Allergy	S , H	R	T	D	AL, VP, NN, P	M, T, TX	AL	
5) Seasonal	H	A	H	D	AL , P	M		Q
6) Flood	S , H	R	T	D	AL, VP, NN, P	M, T, TX	AL	
7) Disaster mitigation	S , H	R	T	S	AL, NN, P	M		Q
8) Cabs	S, H	A	T	S		M		
9) Mapping Ideas	H	A	H	D	AL	M , T		
10) Earthquake	H	R	H	D	AL, VP, NN, P	M, T	AL	
11) Smart Classrooms	S	R	T	S	AL, NN, P		AC	
12) Military Sit. Awareness	S, H	R	T	S	AL, P	M, T		Q
13) Plant hardiness	S	R	T	D	AL,P	M		Q
14) Nokia 'Situations'	S	R	T	D	AL, P		AC	
15) Breast Cancer	H	A	T	D	AL, VP, P	M		Q
16) Drought Impact	S	R	T	D	AL, P	M		Q
17) Industrial control	S	R	T	D	AL, P	M	AC	
18) Traffic condition	S	R	T	D	AL, P	M		Q

Table 2.2: Survey of situation aware applications and their characteristics.

LEGEND:

Source: Sensor based or human report driven (S,H), **Timespan:** Real-time or Archived (R, A), **Diversity:** Homogeneous data or heterogeneous data (H,T), **Integration:** Shallow or Deep (S or D), **Operations:** Arithmetic Logic, Value Projection, Nearest neighbors, Spatio-temporal properties (AL,VP,NN,P), **Visualization:** Maps, Charts, Timelines, or Text (M, C, T, TX), **Actions/ Alerts:** Action-taking, alerts (AC, AL), **Querying:** (Q)

A first glance at the table demonstrates the sheer diversity and applicability of situation-aware applications on multiple aspects of human lives. We also notice that while they are disparate, there exist certain commonalities across applications. We revisit the diversity, and the commonalities identified (in particular those in data operations) when selecting the generic abstractions for building situation based applications.

2.4 Design goals for framework to build situation-aware applications

After considering the nature of different situation-aware applications, we proceed to identify the requirements of an effective framework for supporting such applications. We identify 3 design goals:

1) Expressive power

2) Lower the floor

a. Reduced time to build

b. Lower CS expertise required

3) Raise the ceiling

a. Better designed situation detectors.

b. Provide personalization options

The concepts 'Lower the floor' and 'raise the ceiling' are inspired by [75]. Let us discuss what we mean by each of them.

2.4.1 Expressive power

Given the diversity of applications observed in different applications, a framework designed to recognize situations across them needs to be versatile. This implies that the framework needs to focus on the commonalities, and also start with aspects which are common across applications. The last mile specific issues can be left to the individual application designers where required.

2.4.2 Lower the floor

The framework should resonate with the ideals of [19] in that, " ... new breed of applications, often developed by nonprofessional programmers in an iterative and collaborative way, shortens the traditional development process of edit, compile, test, and run. Situational applications are seldom developed from scratch; rather, they are assembled from existing building blocks".

To truly allow web scale innovation, and cater to the "Long tail of (situation) applications" [106], we need to make sure that the situation detectors are easy to build and do not presume CS expertise. The user input needs to be a declarative specification of *what*. The procedural details of *how* need to be abstracted away wherever possible.

2.4.3 Raise the ceiling

The framework should not only support situation recognition but also raise the quality of the detectors defined. We consider two different aspects of this raising of the ceiling. First is the design process of the applications. The framework should include design guidelines and wizards to ensure that the designers do not fall into common

early mistakes. For example, the designers should not start this process bottom-up i.e. focus on the data sources available, and think what they can recognize with it; but rather go top-down i.e. start with the goal and identify the data sources required. Similarly, we want to make addition of different data streams, operators, and features to involve minimal cost. Hence the complexities of operator implementation or writing data wrappers should no longer be a factor in influencing which affordances are provided by an app. Once the design process selects certain modules they should be available at minimal cost.

Second is the ability to support personalization. Traditional situation recognition and decision making has focused on single large scale (e.g. over city, state, country) decision-making. Today, we need tools to individually access each user's inputs and combine this with the surrounding situation for personalized decision making.

2.5 Components required for the framework

In order to support the design goals discussed, we have identified the three important components required for such a framework.

2.5.1 The building blocks

To increase the expressive power and to 'lower the floor', we need to identify common building blocks which can be used to build a wide variety of applications. The building blocks need to be built upon abstractions which are commonly understood by all application developers, and are applicable across different applications (e.g. Space and Time).

2.5.2 Modeling approach

To 'lower the floor' we need to provide a set of guidelines so that a new application designer is not overwhelmed by the task at hand. Building a system to handle the 'Flu epidemic in USA' might appear to be too vague and daunting for a new user. But with some guidance through the design process, the users can break their problem into modular, explicit, and computable chunks. Equally importantly, the resulting guidelines can help 'raise the ceiling'.

2.5.3 Rapid prototyping toolkit

The end goal of the framework is to build working applications for personal and societal good. Hence providing a toolkit (graphical or API based), which allows the users to quickly translate the models into working applications will tremendously 'lower the floor'. On the other hand, an ability to rapidly reiterate and redefine the applications will help 'raise the ceiling'. Further, an ability to personalize the recognized situations and configuring action alerts will help raise the ceiling.

Chapter 3

Related Work

The proposed work lies at the intersection of multiple active research areas. Here we discuss the related work in context of the problems studied. We first survey the related areas which tackle situation related problems. Next, we paint a timeline of how the focus on recognizing different concepts (e.g. objects, events, situations) from multimodal data has varied over time. Lastly, we discuss other attempts at building end-to-end toolkits to support multiple (situation-aware) applications. As already discussed the notion of situation is interpreted very differently across different areas, and here we try to be as inclusive as possible.

3.1 Situation awareness across research areas

Multiple application and research areas tackle situation related problems. Certain areas are more active than others, and here we simply consider some representative efforts in each area to draw out the holistic landscape.

GIS (Geographical Information Systems)

Geographical Information Systems research focuses on developing tools for understanding and assimilating geography driven information. Applications studied include business intelligence, climate modeling, disaster relief, and city planning. Over the years, this area has led to progress in better understanding of satellite data, data modeling, geographical analysis operators, and analysis tools (e.g. ArcGIS) [58]. The focus of this research area has been on creating sophisticated spatial analysis tools. For example [7] describes a collection of spatial analysis operators suitable for supporting different GIS applications. However, the field has historically focused on disk-based archived datasets, and aspect of time and real-time data are only beginning to make inroads into research [91].

Active Databases and Stream processing

Interest in understanding events in data streams started in database community in the form of stream databases or continuous queries. With the increasing number of applications in need of real-time data stream processing to allow timely response to complex situations, different information stream processing frameworks have been proposed. Such tools have been applied to multiple applications including wireless sensor networks [68], financial ticker analysis [51], RFID reading stream investigation [10], traffic data management [8] and click stream inspection [47]. Similarly, research on Active database systems has resulted in systems which can recognize, evaluate and react to database situations [20, 21, 42, 86].

Event-Condition-Action (ECA) rules have been proposed as the general problem formulation in active database systems. Primitive events and simple temporal operators are designed to describe events or event sequences in database systems. To deal

with the applications where diverse data sources, high data rate and large number of queries are possible, data *stream* management systems (DSMS) have been developed [1].

Complex Event Processing (CEP) systems [24, 107, 111, 67] push forward the conventional pub/sub works by providing users with logical and temporal operators for describing composite events. Commercial vendors, such as IBM, Oracle, Tibco, Coral8, and StreamBase, have built event processing systems, which integrate functionalities of both DSMS and CEP systems. A comprehensive survey on data stream processing can be found at [43].

Although there have been multiple efforts at data stream processing, some important aspects of data streams have not received enough attention.

1) Data Type: As mentioned, DSMS are designed to handle streams of generic data type. However, these systems have been focusing only on structured or semi-structured text data. No existing framework has attempted the integration of data streams in textual, visual, audio and other multimedia formats. With the development of mobile devices, sensor networks and social networks, the majority of all data in the world is in rich media (e.g. audio-video, images) formats.

2) Spatial Data: Although CEP systems explicitly process event streams, spatial attributes of events have been purely regarded and analyzed as symbolic data instead of geographical data. This simplified data model may not be appropriate for analyzing complex spatio-temporal patterns.

3) Scalability: Limited by the data type and modeling approach, the data volumes that could be managed by existing systems is significantly restricted. With the advent of big data, data stream processing systems which are able to handle heterogeneous data streams at scale will need to be redesigned.

This dissertation focuses on issues 1) and 2). A related effort [40] addresses some of

the challenges in scalability.

Aviation and Military applications

Pioneering works by Endsley et al [38] have focused on creating better situational awareness for fighter pilots, and air traffic controllers. Other works focus on an action driven approach and support creating actionable inference (e.g. [2]) . (Refer to Chapter 2 for a detailed discussion on interpretations of 'situations'.)

Military applications often use the JDL (Joint Directors of Laboratories) Model for information fusion [103]. Situations are considered a Level 2 abstraction in this model. In this model, situations capture a higher level of abstraction than sensor fusion, but lower than 'impact assessment' and 'resource management'. Jakobson et al [56] have broadly studied the problem of 'situation management'. They consider the situation model as a knowledge level view created using a domain based ontology. Such a situation model requires both human and signal level intelligence. Such a situational model can work in an 'investigative', 'control' or 'predictive' phase depending on the application's focus on past, current or future actions. Greenhill et al. have developed SDL (Situation Description Language) [45], which is a full language implementation undertaken to help submarines develop a tactical understanding of the situation at hand.

Robotics and AI

John McCarthy discussed situation recognition as one of the most fundamental problems in AI in 1968 [70]. He defined the field of situation calculus. A major wave of enhancements, implementation and formalization was done by the cognitive robotics group at Univ. of Toronto led by Reiter [63]. Situational calculus has a major issue

known as the frame problem i.e. it cannot be used to derive the non-effects of actions [83]. There is also a closely related set of works on event calculus [62], which describes how events and their effects can be represented using a logical language. The specific constructs used by them (and situation calculus) are *Actions, Fluents and Situations*. While actions are used in their normal sense and situations have been discussed above, Fluents are statements whose truth values may change over time. Reiter's group has applied Situation Calculus to a number of applications, most notably those dealing with robotics, and planning in controlled environments [26].

Linguistics

In linguistics, situation semantics is used to refer to a mathematically based theory of natural language semantics introduced by the mathematician Jon Barwise in 1980 [13]. Barwise and Perry began with the assumption that humans use language in limited parts of the world to talk about (i.e., exchange information about) other limited parts of the world. He called those limited parts of the world, 'situations'. A potentially interesting line of work on Semiotics and "Situation based control" was done by Pospelov in Russia. Semiotics as a science of signs explores the syntactic, semantic and pragmatic aspects of signs. The main text however was published in Russian and the details remain inaccessible. Based on citation in [56], situational control theory [79] was based on semiotic models of the domain developed in linguistics and language psychology. Pospelov considered situations as states of the relations between objects referred to at some point in time. Pospelov's situation formalism was based initially on graph theory and finite state machines, and later formal relational expressions close to FOL (First Order Logic).

Context-based systems

Yau and Liu [114] proposed the use of situation modeling to allow environments to adapt their behavior based on them. They make use of an OWL ontology to describe situations. Specific application based situation ontologies have also been discussed for disaster control in [99] and surveillance [74]. Wang et al. [109] have used situations to automatically configuring mobile phones' sound/vibration settings.

Computational Social Science

Multiple efforts under the names of web mining, social network analysis, reality mining have studied similar problems. While they may not explicitly use the term 'situation', they do focus on the same problem – that of understanding aspects of human life and the world by observing the data footprint being created. Blogscope [11] was an influential effort at mining blog content to understand emerging topics and events of interest. Pentland and colleagues have been leading an effort at understanding human dynamics via analysis of phone-logs and social-sensors [4, 35]. Mining phone usage patterns to understand social behavior has also been explored in [44] and has been shown to be a proxy for city scale sensing in [82]. Recently, Twitter has been used for analyzing presidential debates [28], detecting earthquakes [87], and predicting the stock market [14]. There have also been efforts at providing toolkits like Twitris [53], TwitInfo [69], and Truthy [81] for Twitter data visualization and analysis.

Media processing and applications

Multiple media (image, audio, video, sensor) processing efforts have tried to focus on the problem of sense-making. The research on different aspects of media processing

has led to a rich collection of tools and techniques relevant for in-silo media processing. Research at combining different modalities has focused on information fusion aspect, and only slowly started to reason and think in the real-world, rather than the media silos.(See Section 3.2 for more detailed discussion.) Multiple 'situation-aware' applications have been discussed in the media processing literature.

For example, Brdiczka et al. [17] have described the use of situation models to aid video surveillance. They characterize environments based on events and entities (person, place and objects). The considered 'situations' were walking, browsing, fighting, waiting, and object left. Crowley [23] has described the use of situation models for observing human activities. According to this work, human activities follow loosely defined scripts, and it aims to provide "a conceptual framework in which scripts for human activity are described as scenarios composed of actors and objects within a network of situations". Shirehjini [76] has described a situation modeling example in the multimedia presentation domain, but focused specifically on the 'situations' of "Setup, Welcome, Presentation, Explanation, Question-Answer, Discussion, and Conclusion". The parameters identified for capturing the situation were "Activity, Location, Interaction, Media, Device, Environment and Audience states (e.g., light and noise setting, user movement, number of person, level of attention, interruptions)". These parameters served as distinguishing characteristics for mentioned situations at a high-level and did not depend on a specific sensor type or situation recognition technologies. Dietrich et al. [29], look at situation modeling from a building automation perspective.

3.2 Progress in the field of 'concept recognition' from multimedia data

Media (image/ audio/ video) processing research has made major progress on the tasks of object recognition, scene recognition, event recognition, and trajectory recognition. Each of these concept recognition problems focuses on analyzing observable media to recognize an application-centric concept (e.g. 'chair', 'office', 'intrusion'). Dealing with the specifics of the media required sophisticated techniques for pixel analysis, noise reduction, signal processing, time-frequency transformations, and machine learning techniques. The complexities of dealing with each of these areas led to multiple research papers focusing just on analysis of a particular media e.g. images. Similarly thousands of research papers have been written on each of the problems of object, scene, event, and activity recognition.

The first well known approach at visual recognition was by Roberts and colleagues in 1960s [36], and the first effort on Event recognition was made by Jain and colleagues in 1980s [50]. In this work we channel the lessons learnt from recognizing intra-media concepts (*i.e. those which manifest themselves, and can be recognized within a single media object e.g. a tree, or a chair in an image*), to define and recognize evolving concepts (*i.e. those which occur in real world, are constantly evolving, and inherently manifest themselves over heterogeneous multimedia streams from numerous sources*).

As shown in Figure 3.1, Situation recognition builds on and extends object recognition, scene recognition, activity and event recognition, and complex event processing. The challenges in Situation recognition are very different from those in object or event recognition. For example, we can no longer just *accept* heterogeneity, or allow multiple data streams; we need to *expect* them and capitalize on them. We need to focus on recognition of real world phenomena based on their footprints across multiple het-

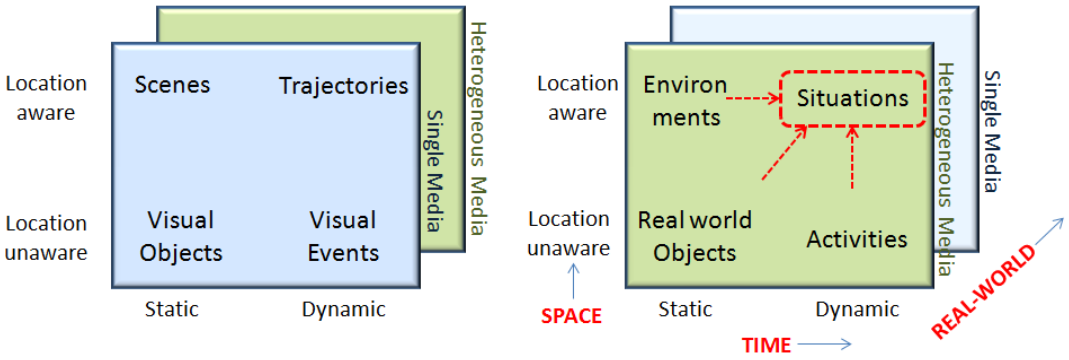

Figure 3.1: Different Types of Concepts can be recognized in different data availability settings. Single media, such as images, results in concepts more in images than in the real world, but using different media it is possible to recognize concepts in the real world

Approach	Type of features used	Flexibility at run-time
Computer Vision	Rigorous	Low
Databases	Basic	High
This work (design goal)	Rigorous	High

Table 3.1: Design goal of the proposed work

erogeneous media. This allows for solving practical human problems by correlating data ranging from social media, to sensor networks, and satellite data.

For doing this we build upon techniques rigorously developed by computer vision and multimedia researchers which aggregated pixel values for identifying low to mid-level features (moments, color, shape, texture, area, edges) to recognize higher level concepts (boy, face, rose) in applications. However, the computer vision research field has mostly focused on rigorous feature extraction for the classification problems; its recognition models (e.g. 'chair' detector) tend to be opaque and un-editable at run time. The database field, on the other hand, has focused on transparency of information; information about any entity or concept can be obtained flexibly at run-time by declaratively defining a 'model' based on attributes of the data.

In this work we exploit the positives of both the sides, and create models for situations which use rigorous features, but yet are re-configurable at run time. Situation models

are created by domain experts using a set of declarative operators. Under the hood, the system automatically implements these spatio-temporal operators by leveraging relevant computer vision operators, and it presents the evaluation results in real-time to the user. Thus the system supports sophisticated situation recognition, but is reconfigurable on-the-fly to support multiple queries and applications.

3.3 Toolkit support for situations, context, and data analytic applications

One of the design choices made early in this work was that of its focus. Efforts could be directed towards automating the process of situation recognition-much in line with the dominant trend in Computer Vision Research on concept recognition. Alternatively the work could focus on building a human-in-the-loop framework which can at run time be flexibly be configured to match and recognize different situations. We did not want the system to make a 'cold start' and re-learn the classification model each time new concept needs to be recognized, and hence we decided to provide a toolkit which allows different users to create different situation aware applications using the framework.

Toolkits supporting several applications need to integrate heterogeneous data types and support multiple operations for their integration. The field of web Mashups is driven by very similar motivations. Mashups try to bridge the disconnect between the domain experts and the computing experts. By allowing very easy integration of different web data sources and operations as *widgets* they want to allow creation of sophisticated web applications. Based on a White paper by IBM [19] now " ... businesses can remix information from inside and outside the enterprise to solve sit-

uational problems quickly. Individual users can create these situational applications themselves, by 'mashing' together multiple information sources into a lightweight Web application that is 'good enough' to solve a situational problem that pops up".

Visual mashup languages like Yahoo Pipes [39], and MashArt [25] have tried to make it easier for non-Information System experts to combine different web streams. They, however, still stop at basic operators (no spatio-temporal analytics), and even those are often considered difficult for real domain experts to be effectively used [22]. Other attempts like Popfly [46], DataMasher[1] (Winner of OpenData competition), 'Mobile Web Mashups' [106], and IFTTT [2] also attempt to support web mashup environments for application creations. However, as seen in Chapter 2, their support still tends to be limited in terms of the heterogeneity of data supported or the kind of operations possible.

In this work we focus on declaratively providing sophisticated spatio-temporal analysis operators on data coming in from a wide variety of structured and unstructured multimodal data streams. In order to do so we draw upon and extend concepts from multiple research areas. As summarized in Table 3.2, multiple areas have looked at different aspects related to this framework. To the best of our knowledge, this work is the first end-to-end effort which combines heterogeneous data streams, explicitly considers human sensors, performs data-analytics, studies situations, considers spatial semantics, handles real-time streams, supports personalized control, and provides a generic toolkit to build multiple applications.

[1] $http://www.datamasher.org/$
[2] $http://www.ifttt.com/$

Area	Combine hetero data	Human sensors	Data analytics	Define situations	Location aware	Real-time streams	Person-alization	Control Actions / Alerts	Toolkits
Situation Awareness	X			X	o	o	o	X	X
Situation Calculus				X					
Web data mining	o	X	X		o		X		X
Social media mining	o	X	X		o		o		X
Multimedia Event recognition	X		o		o	o			
Context-based computing	X	X	X	o		o	X	o	X
Complex Event processing/Active DB	X		X	o		X		X	
Geographical Info. Systems	X	o	X		X				o
Mashup toolkits e.g. Y! pipes, ifttt	X	X	o			X		X	X
This work	X	X	X	X	X	X	X	X	X

Table 3.2: Survey of various related research areas and their characteristics. Legend: X = Well supported, 'o' = Partial support.

Chapter 4

Overall Framework

This chapter sketches the workflow for modeling and building situation aware applications using the framework. It discusses the design features of the framework and gives an overview of each component rather than zooming into the details (which are presented in the following chapters).

As shown in Figure 4.1, the overall process can be split into 3 stages. 'Situation Modeling' is the process of conceptually describing situations of interest using generic building blocks. They yield explicit, computable blue-prints of data sources, operators, and the logic required to recognize any situation. The 'Situation Recognition' stage physically implements these models by applying different operations on the data to derive situational information. In the third stage, this information can be used for drawing insights or be personalized to generate alerts.

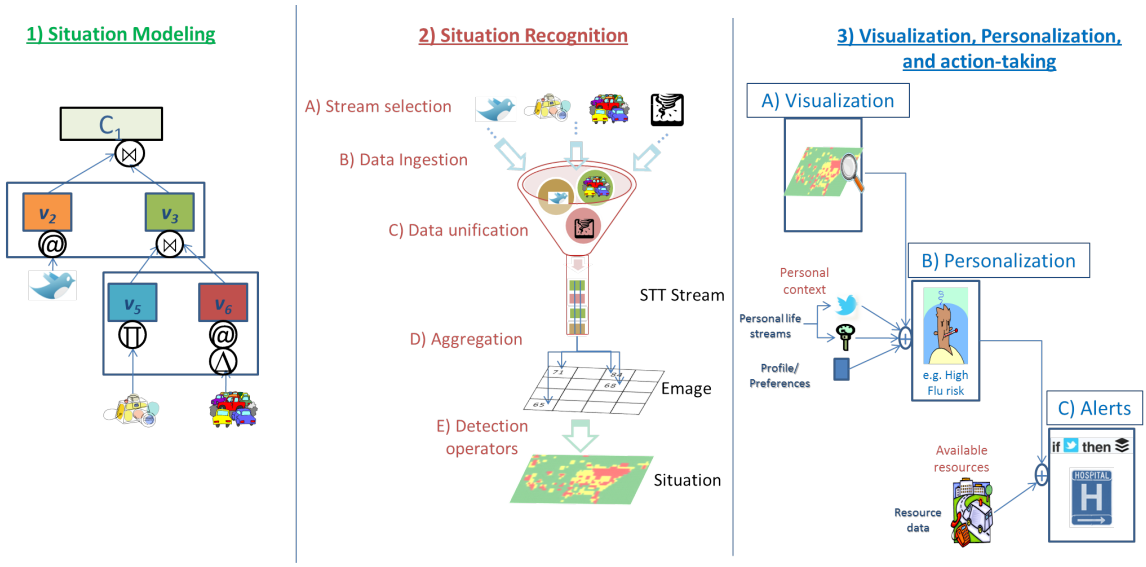

Figure 4.1: Different phases in the framework: Situation modeling, recognition, and alerts

4.1 Design features of the framework

Based on the discussion in Chapter 2, we have identified some design features essential to the framework for supporting situation-based applications.

Using humans as sensors

The framework explicitly considers humans as sensors. Humans can describe aspects of a situation which are not yet measurable by any hardware sensors. Humans can describe perceptions, emotions, impressions (e.g. for business products), counter state censors (as seen in Iran elections), be first respondents (e.g. Hudson river landing), emergency reporters (e.g. Haiti earthquake rescues), and even pass unconfirmed/unofficial information reports (rumors, merger-info, scandals). The growing importance of multimodal user contributions (e.g. Twitpic, Twaudio, Flickr), and location based services (e.g. Foursquare, Gowalla, Yelp) clearly highlights this. Social media networks arguably thus represent a large self-organizing sensor network, more

sophisticated than any other sensor network currently deployed.

Consider space and Time as fundamental axes

In designing different aspects of the framework (data representation, operators, building blocks) we consider space and time to be the fundamental axes. This draws upon the basic nature of the physical world i.e. space and time are the independent axes which unify all physical phenomena. This also builds on and extends the concepts from the fields of GIS, cartography, satellite sensing, and location-based computing.

Support real-time situation evaluation

Today most business data is stored for later analysis, but the value of the data decreases over time; data has a half-life[1]. The more often we collect data and the more local the data is, the shorter is its period of relevance. The prominence of real time data in Twitter, Facebook, Satellite, and Stock-market streams also highlights this. Hence, in this work we explicitly consider data-streams (and not static or on-disk data) to be our primary data model. All the operators and representations discussed work on streams and a standing query notation.

This underscores the fact that practical situations need to be recognized in real time to save lives and resources.

[1] $http://www.numenta.com/grok\_info.html$,
$http://slashdot.org/topic/bi/business-data-has-a-half-life-of-usefulness-nucleus-research/$

Generate personalized actionable situations

Everybody's personal situation is different. Hence the framework needs the capability to generate personalized situations and corresponding alerts to be really useful to a variety of users.

4.2 Situation Modeling

The first step in developing situation aware applications is that of conceptually modeling the situation of interest. In this phase, the application designers (domain experts) can systematically externalize their internal models of the situations of interest [94]. Conceptual modeling allows for better designed, modular, and flexible systems which are more likely to match the application requirements [27]. The conceptual models also allow for the identification of the data sources and the operators required. A generic toolkit to guide this process consists of the following components:

1. The 'building blocks'
    - Operators
    - Operands
2. A prescriptive approach for modeling situations using the operators and operands.

The ***operands*** in the framework are data and knowledge level constructs onto which different operators are applied. The ***operators*** are a set of declarative algebra of operations which capture a wide variety of functionality required for spatio-temporal analysis. The set of operators defined in this work are *Filter, Grouping, Aggregation,*

Spatio-temporal Characterization, Spatiotemporal Pattern matching, Transform, and Learning.

The approach for creating situation models involves the application designers taking a higher-level, (possibly vague) situation characterization and splitting it into a set of more concrete features. Such a process is repeated recursively until each of the descriptors is defined only in terms of the actual data sources and operators available. Once completed, the process yields blue-print plans to combine different data streams into situations.

4.3 Situation Recognition

In this stage the conceptual models are operationalized to generate output from the incoming data. The overview of the process of moving from heterogeneous streams to situations is shown in Figure 4.1 (phase 2). This process involves 5 steps:

1) Stream selection

2) Data ingestion

3) Data unification

4) Aggregation

5) Situation evaluation

Relevant data streams are identified by the domain experts, based on which the relevant wrappers ingest the data. A unified spatio-temporal format records the data originating from any spatio-temporal coordinate using its numeric value. Aggregating such data results in two dimensional data grids (called E-mages) which can be extended over time into E-mage Streams. The situational descriptor is defined as a

function of different spatio-temporal features derived using operations applied on the E-mage Streams.

This process follows from the discussion in Section 2.2, which explained how situation recognition can be split into the problems of obtaining features from diverse spatio-temporal data, and using the features to evaluate situations.

4.3.1 Data stream selection

The situation models act as blue prints which identify the data streams and operators required to recognize a situation. The data streams can originate from a person's mobile device, social network updates, stand-alone sensors, satellites, websites or archived web data sources. The framework supports as many different types of raw data as may be relevant.

4.3.2 Data ingestion

The ingestion of data from different sources is based on wrappers which bring the external data into the system. For computational purposes all data streams are normalized to numeric streams, but the underlying 'raw' data is also maintained while required by the application. Each data stream has a set of associated spatio-temporal parameters which need to be configured.

4.3.3 Data unification

The heterogeneous data streams are unified based on focusing on the commonalities across them. Each data stream is considered to be reporting certain thematic ob-

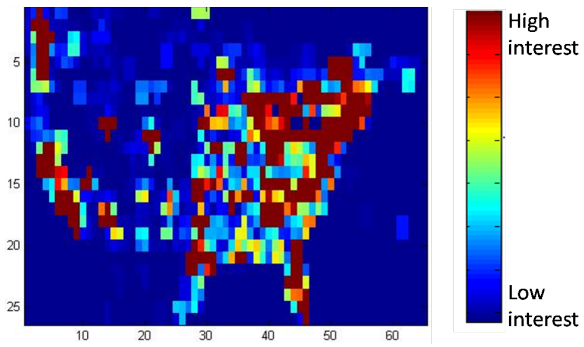

Figure 4.2: An E-mage showing user interest across mainland US in terms of number of tweets containing the term iphone on 11th Jun 2009

servations from different spatio-temporal coordinates. Hence a STT or Space, time, theme (i.e. where-when-what) tuple is used to organize all types of data [93].

A STTPoint is represented as:

$$STTPoint =< latitude, longitude, timeStamp, theme, value > \tag{4.1}$$

By extension, a flow of STTPoints becomes a STT Stream.

4.3.4 Spatiotemporal Aggregation

Spatial data can be naturally represented in the form of spatial grids with thematic attributes. Values in STTPoints that are collected in a time window over STT stream can be combined and aggregated to form a two-dimensional data grid. The data grid together with related STT information is called an E-mage. E-mages capture the semantics and notion of spatial neighborhood very elegantly, and geographical-joins [52] between data streams reduce to simple overlaying of grids. An example of an E-mage is shown in Figure 4.2.

A flow of E-mages forms an E-mage Stream, which serves as a first-class citizen i.e.

fundamental data-structure in the framework.

These gridded representations spatio-temporal data are very analogous to images, pixels, and videos which have been studied by media processing researchers for a long time. We exploit these analogies from multiple perspectives.

1. Visualization: The grid-like representation allows for intuitive visualization and aids situation awareness for a human user. Humans are quite used to seeing satellite image and GIS data on similar interfaces.

2. Intuitive Query and mental model: The correspondence of the human mental model of spatio-temporal data with the query processing model, makes it easier for humans to pose queries, and understand the results.

3. Data Analysis: Such a representation allows for exploitation of a rich repository of media processing algorithms which can be used to obtain relevant situational information from this data. For example, well developed processing techniques (e.g. filtering, convolution, background subtraction) exist for for obtaining relevant data characteristics in real time.

4. Efficiency: The pixel/image based representation reduces the run-time processing requirements from potentially millions of XML data feeds to a rectangular grid representation of known size. This allows the *run time* query process to work on just the e-mages (which can be directly stored in the main memory due to much smaller size), rather than the entire raw data corpus. The process of creating e-mages out of the raw data can be undertaken separately without affecting run-time performance.

5. Privacy preservation: Such an aggregation approach aids applications that need to maintain *individual* user privacy. Spatio-temporal 'binning' allows the higher level algorithms to work on the aggregate representations, without focusing on individuals.

4.3.5 Situation evaluation

The situation at any location is characterized based on spatio-temporal descriptors determined by applying appropriate operators on E-mage Streams. The supported operators are *Filter, Grouping, Aggregation, Spatio-temporal Characterization*, and *Spatiotemporal Pattern matching* [41].

These operators are designed to be declarative to allow end users to describe their data needs rather than procedurally handling the details of manipulating the data. The aim of designing this algebra is to retrieve relevant spatio-temporal-thematical data (E-mages, E-mage Streams, or their attributes) by describing their characteristics [105].

An overview of the operators is shown in Figure 4.3.

4.4 Situation visualization, personalization, and alerts

The evaluated situation information can be used for visualization, analysis, and control actions.

4.4.1 Situation visualization

Spatio-temporally aligned situation data can be visualized on maps, timelines, and reported as text. The outputs of different queries are either a temporal stream of E-mages (which can be overlaid on maps), or STTPoints (which can be shown on a map and a timeline).

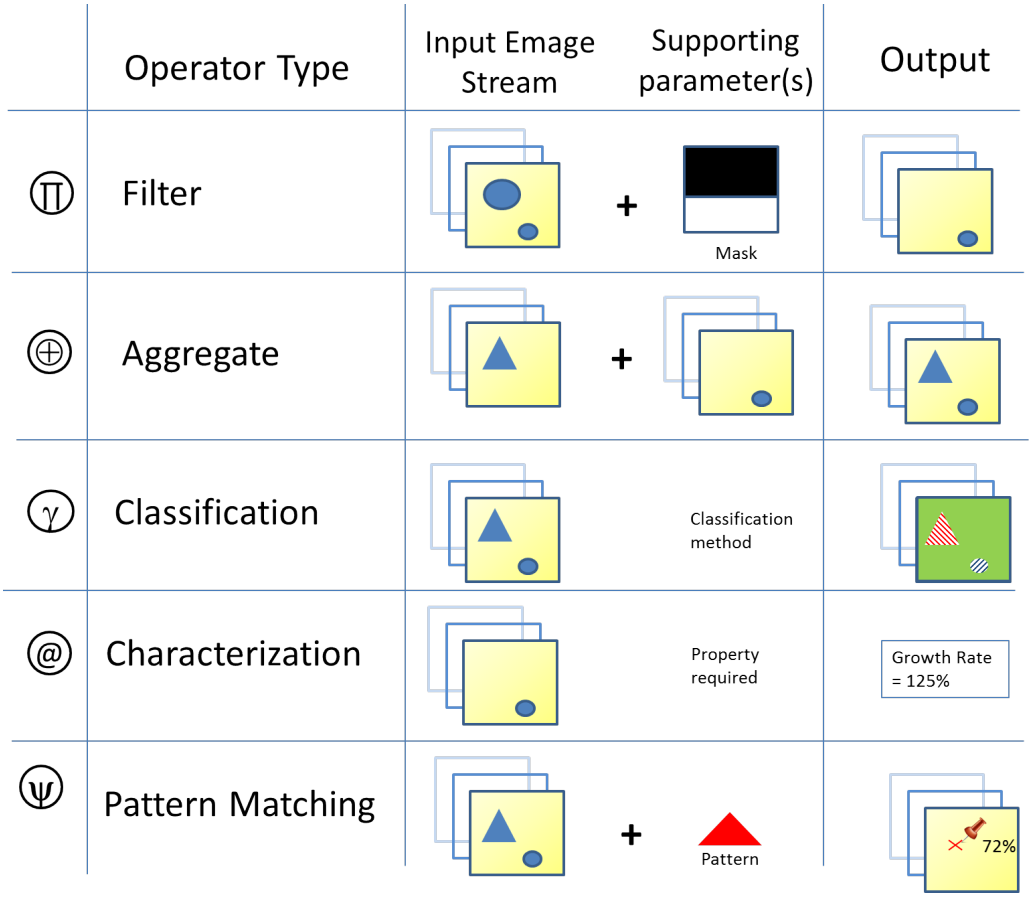

Figure 4.3: Different operators for situation recognition

4.4.2 Personalization and Alerts

The recognized situation can be personalized to each user by projecting out the situation classification value for user's location and combining it with personal parameters. The individual parameters can be personal data streams (e.g. activity level) and action recommendations can be undertaken using approaches similar to E-C-A (Event-Condition-Action) [72]. The spatio-temporal coordinates associated with each data stream are used to direct users to nearest location satisfying certain conditions. A combination of macro-situations and personal parameters can be used to configure different Situation-Action templates. Multiple such templates can be registered to provide customized alerts to all recipients.

Chapter 5

Situation Modeling

Situation modeling is the process of conceptually defining what constitutes an actionable situation in the application designer's domain[1]. It allows the designer to externalize what she means by a specific situation of interest (e.g. an 'Epidemic'). Building this model in terms of conceptual building blocks rather than directly implementing them in code has multiple advantages. First, the application designers get to focus on the 'Big-Picture' rather than getting bogged down by the implementation details. Next, such a process encourages a goal-driven thinking rather than an availability driven thinking [27]. Lastly, the modeling in terms of generic blocks allows for easy reuse of components across applications.

The output of such a process are S2S (Situation-to-Sources) diagrams which act as blue prints for the situation detectors and succinctly identify the different data sources and operators required to implement the detectors in various situation-based applications.

The generated models can also be used as knowledge representation tools [101]. They

[1]Hence, here the intended 'user' is a domain expert/ application designer.

capture and retrieve an otherwise intangible asset - an expert's knowledge which can then be shared across organizations or even the whole Web. Such tools also allow the redirection of human focus to where it needs to be - defining the application logic; rather than the procedural details, and automatable tasks.

To aid the creation of different situation models and their implementation, the framework provides:

1. The 'building blocks'
    - Operators
    - Operands
2. A prescriptive approach for modeling situations using the operators and operands.
3. Steps and guidelines for refining the models so that they are well-defined, explicit, and ready for translation into code.

5.1 Operators and Operands

The constructs employed need to be well defined and quantifiable. They also need to be generic enough to capture the common requirements across multiple applications. Lastly, specialized building blocks catering to specific applications should be avoided, as too many blocks lead to higher cognitive load and may confuse the application designers using them.

Considering these (often competing) considerations, the framework defines the following set of operands and operators.

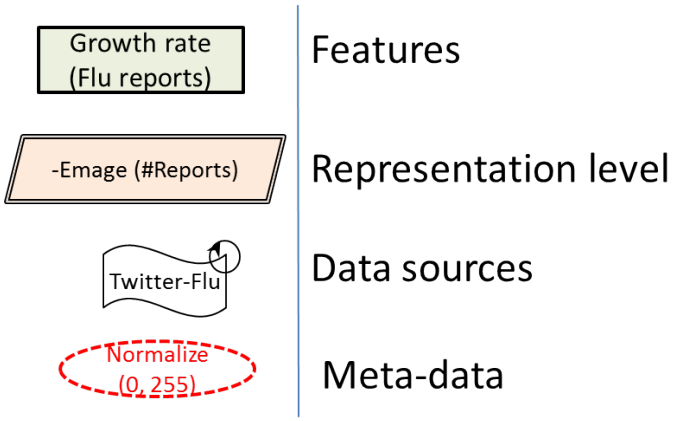

Figure 5.1: Operands for Situation Modeling

5.1.1 Operands

The operands in the framework are data and knowledge level constructs onto which different operators are applied.

The operands defined (also see Figure 5.1) are:

1. **Feature:** Any spatio-temporal descriptor that contributes towards defining the overall situation. (e.g. growth rate of Flu reports)

2. **Representation level:** The level at which data needs to be analyzed viz. individual data nugget(STTPoint), spatial aggregation (E-mage), or spatio-temporal aggregation (E-mage stream)

3. **Data source:** The resource (e.g. URL) for obtaining the data.

4. **Meta-data:** Any additional details (e.g. spatio-temporal bounds, operator types, normalization bounds, output variables, thresholds) required for complete specification of the features or operators.

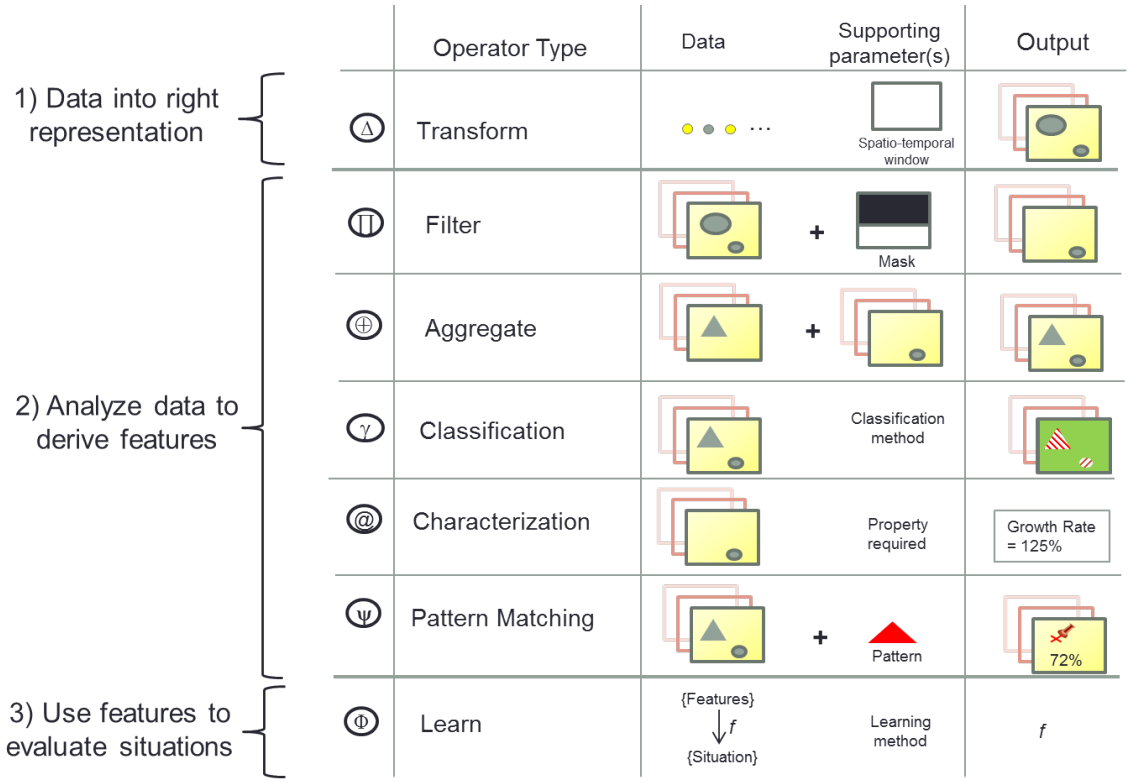

Figure 5.2: Operators for Situation Modeling

5.1.2 Operators

The operators are applied to the operands to bring data to the right representation, analyze data to derive appropriate features, and to use features to evaluate situations. The operators defined (refer to Figure 5.2) are:

1. **Transform (Δ):** This allows the data at any layer to be transformed into the next (higher) layer. It can be for:
   - Data source to STT (Level 0 to Level 1): The wrappers to translate heterogeneous data into STT data nuggets.
   - STT to E-mages (Level 1 to Level 2): Combining STT nuggets into an aggregated E-mage representation

2. **Filter (Π):** This allows for selection of data based on space, time, theme, or value parameters.

3. **Aggregate ($\oplus$):** This allows for data to be combined based on mathematical operators.

4. **Classification (γ):** This operator segments the values into different categories.

5. **Characterization [spatio-temporal] (@):** This operator handles derivation of relevant spatio-temporal attributes (e.g. epicenter, density, shape) from any data stream.

6. **Pattern Matching [spatio-temporal] (Ψ):** This operator allows users to study how closely the captured phenomena match known patterns or related historical data.

7. **Learn (Φ):** This operator automatically identifies the function to combine different features into the appropriate situation classification. The function can be identified using computational techniques (e.g. Machine Learning [6]) once the expert identifies a 'learning' data source which contains samples of different feature values and the expected classification result. This operator is useful in cases where the relative weight of the features on the situational classification is not known to the expert.

The first 5 operators are are chosen based on the survey of commonly operations in situation-based-applications presented in Chapter 2, as well as a survey of commonly applied operators on other grid like data structures (i.e. images). Together they provide the affordances to combine, analyze, and classify different data streams. The

'transform' operator corresponds to the data unification, and aggregation steps (refer to Section 4.3) required to bring data into the unified E-mage Stream format, and the 'learning' operator supports scenarios where the experts themselves are not aware of the right parameters for combining different data-sources.

5.2 The Wizard for modeling situations

A designer can follow these steps to model situations (refer to Figure 5.3).

First, she needs to identify the output state space (i.e. range for the output descriptor). Next she needs to identify the spatio-temporal bounds being considered. The next step is identifying the relevant features useful in defining the situation output. If it is an imprecise classification type of problem, then the designer is required to identify the data source for 'learning' how the different features identified affect the situation classification.

For each of the features identified above, she needs to identify the data sources, representation levels required and the variables/themes needed for the transformation across levels. If the feature considered is atomic (i.e. detectable using well defined operations on a data stream) it is added to the model. In case the feature is not well defined yet, a recursive call is invoked onto the same algorithm to identify the relevant components and details for the one-lower level feature.

As shown in Figure 5.4, such a process repeats itself in a recursive manner (akin to depth first search) until the high level situation description has been partitioned into explicitly observable or computable components.

/* Input: Situation descriptor (possibly complex and vague) */

/* Output: List of intermediate descriptors, data sources, and representation levels required */

Get_components (v){

1) Identify output state space
   - Numeric or Classes
2) Identify S-T bounds
3) Identify component features; $v = f(v_1, \ldots, v_k)$
   - If (type = imprecise)
     - identify learning data source, method
4) ForEach (feature v_i){
   - If (atomic)
     - Identify Data source.
       - Type, URL, ST bounds
     - Identify highest Rep. level reqd.
     - Identify operations.
   - Else
     - Get_components(v_i)

}

Figure 5.3: Steps in Situation Modeling

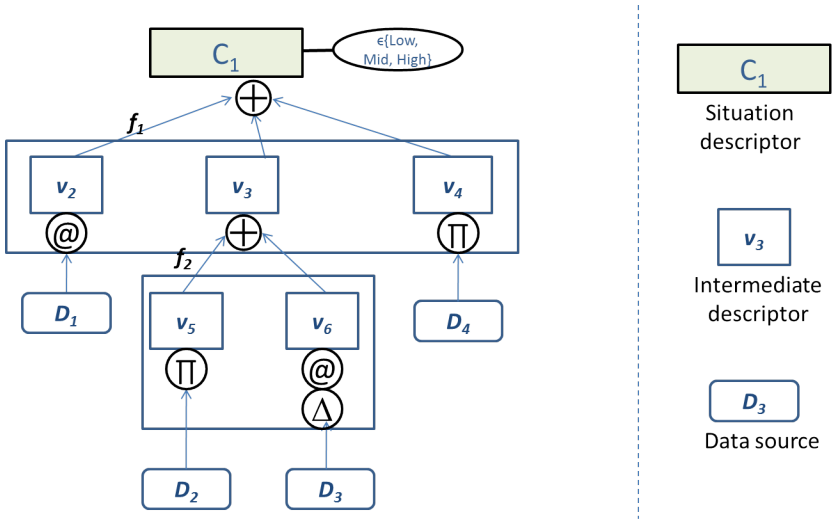

Figure 5.4: Recursive approach for defining situation variables

5.3 Enhancing and Instantiating the model

5.3.1 Refining the model

A model created by the process above captures a baseline representation of an application designer's model of a situation of interest. Just like E/R modeling [24] this process may require multiple iterations before the experts agree on a 'good' model. Suggested criteria to accept a 'good' model are:

1. Do Features identified provide enough discriminative power to the model?

2. Does the data stream chosen capture the semantics of the feature selected?

3. Are there any cyclic reasoning or cyclic dependencies in the features selected?

5.3.2 Instantiating the model

This phase acts as a middle ground between the conceptual models and the physical implementation. Hence it should be undertaken only after the model 'conceptually' satisfies all the design requirements. This phase involves adding the relevant details to make the operators computably explicit i.e. contain enough detail to be translated into code if required. This requires the following steps:

1. Provide necessary parameters for operators (e.g. Operator types, normalization bounds, thresholds) in the model.

2. Refine, if necessary.

The details of parameters required to qualify each operator depend also on the implementation tool employed. (The complete list of parameters required for implementation using EventShop is discussed in Chapter 7). This again undescrores the role of this stage as an intermediary between conceptual models and actual implementation.

Note though that even while configuring these parameters, platform specific details (e.g. implementation language, language related issues, data types, memory management) are not part of this model. Those decisions fall under the purview of the developers of the implementation tool, e.g. EventShop, rather than the domain experts who build models and use EventShop to test them. Some experts may also choose to stop after designing the conceptual models and pass them onto IT support personnel to do the translation into actual applications.

5.4 Example: Modeling Epidemic Outbreaks

Let us illustrate the process of situation modeling by considering 'epidemic outbreaks'. Given as-is, 'epidemic outbreak' is a vague undefined notion. In fact not even all experts agree on what constitutes an Epidemic. Here we discuss the workflow for one possible modeling of epidemics. Following Section 5.2, we first identify the output state space (i.e. required classification into low, mid, and high risk of outbreak). Next, (see Figure 5.5) we identify the spatio-temporal bounds being considered: *USA, with a spatial resolution of 0.1 latitude X 0.1 longitude, and re-evaluation to be made every 5 minutes.* The next step is identifying the relevant features which define the situation output. Lets define 'epidemic outbreaks' as a classification on 'growing unusual activity'. While this is a single feature, it is not atomic (i.e. cannot be derived directly using one data source). Hence we follow the process recursively, and try to model 'growing unusual activity'. This feature is defined based on two component features: 'Unusual activity' and 'Growth Rate'. It turns out that 'Unusual activity' is also not atomic, and needs to be split into the features of 'historical activity level' and 'growth rate'. Let's assume that the historical activity level is available from a curated database and current activity level can be measured based on the frequency of

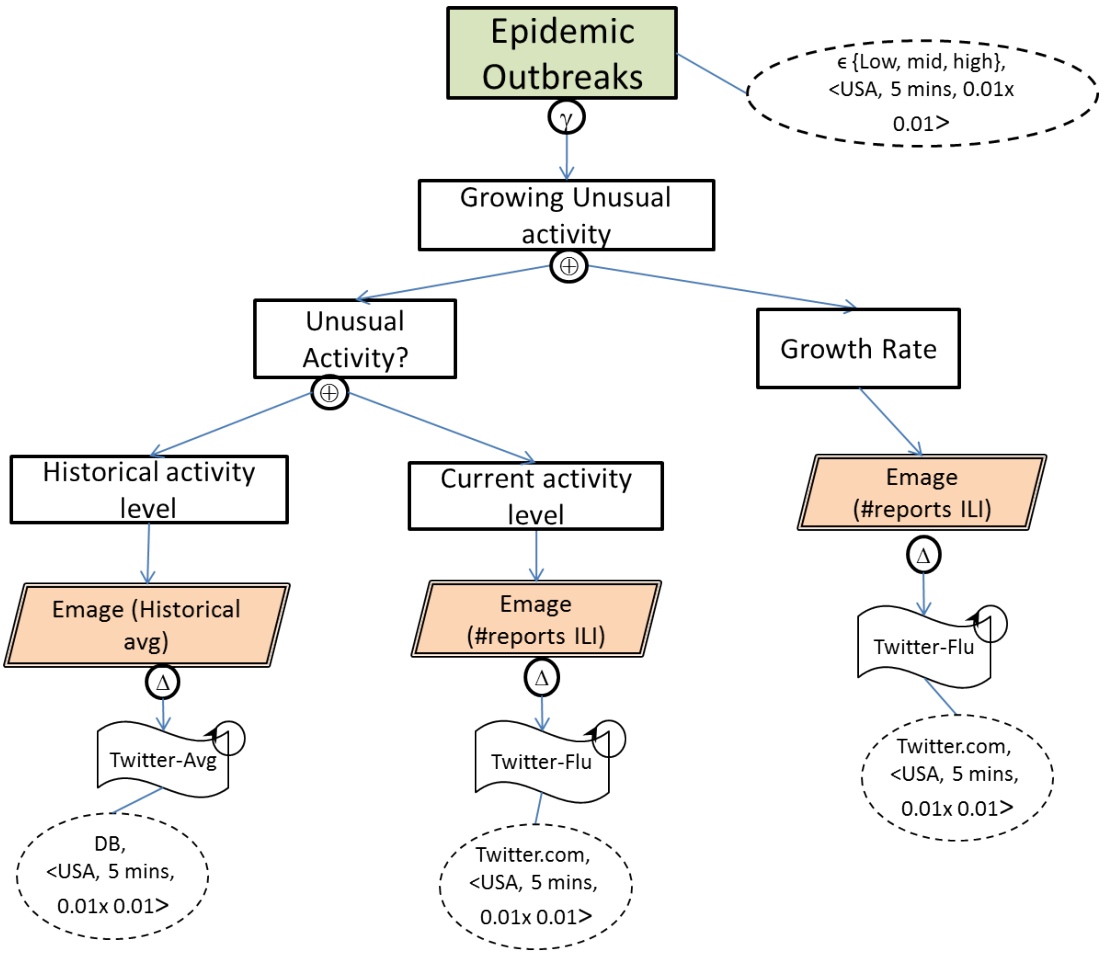

Figure 5.5: Base model created for epidemic outbreaks

terms indicating ILI (Influenza-Like-Illness) on Twitter stream. Similarly, the growth rate can be measured from the Twitter stream.

Hence, now we have three ('leaf node') features which can each be defined using a single data source and hence the modeling is complete. In effect we have split a vague concept (epidemic) into features such that each of them can be derived from a known data source.

In practice, application designers may not be satisfied with the first created model. For example, let's consider 'current activity level' which has been defined based on the number of Influenza-Like-Illness (ILI) reports observed from each location. It

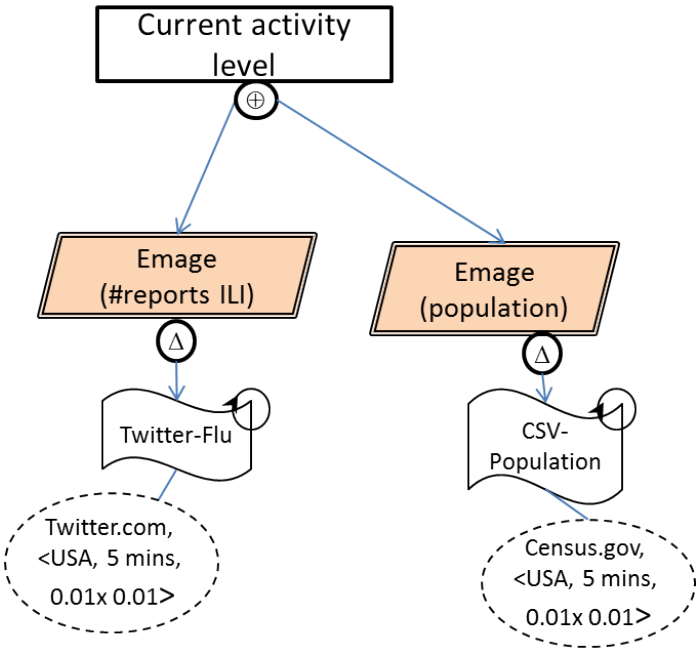

Figure 5.6: Situation model: Changes made in refinement phase

may be better to regularize this value based on the population at each location. This leads to changes in part of the model shown in Figure 5.6.

The last phase in model creation is that of 'instantiating' it. This step involves adding the relevant details about the exact operation to be performed and the associated parameters. Further, certain parameters/ data sources may need to be refined. Here we scale the population data onto the range [0,100] to be comparable to the incidents reported. Figure 5.7 shows the model with the relevant details added (e.g. 30, 70 as the thresholds for classification). Once this step is complete, the created model can be evaluated using EventShop (refer to Chapter 7) or a similar situation evaluation toolkit.

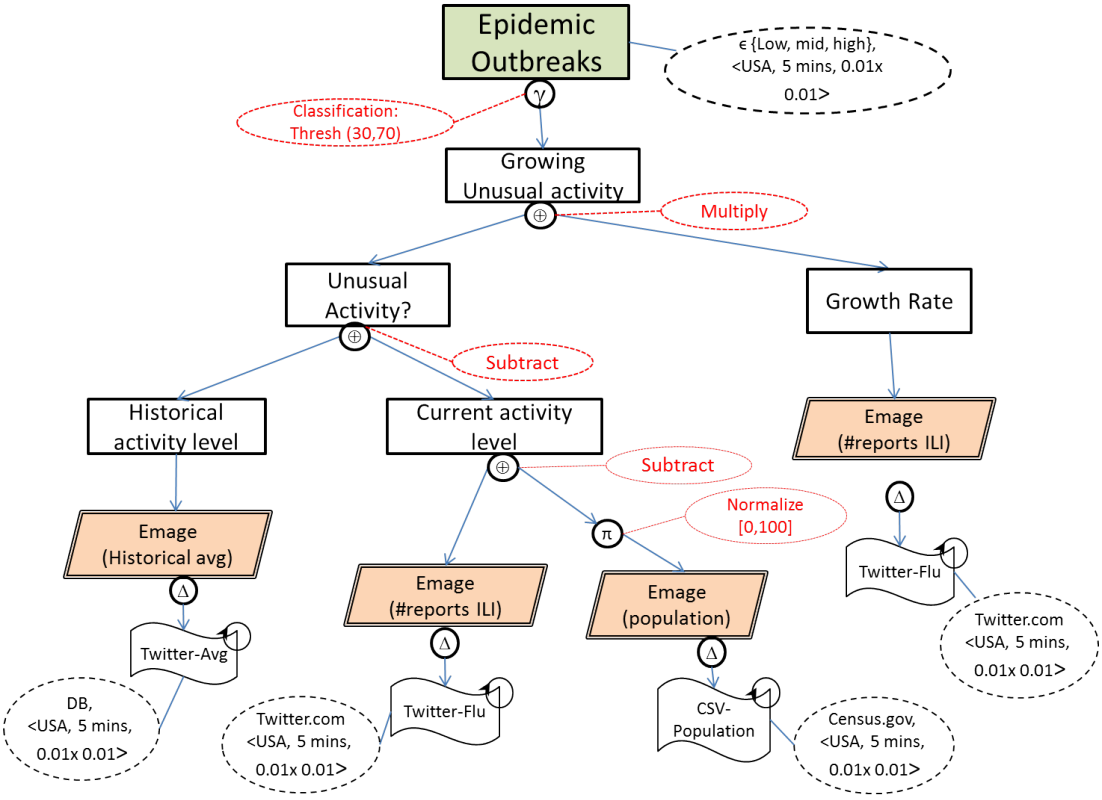

Figure 5.7: Situation model after the instantiation phase (details added in Red)

Chapter 6

Data Representation and Situation Recognition Algebra

The translation of the conceptual models onto applications requires computational support. As discussed in Chapter 1, this requires development of newer tools and techniques which can unify data, maintain spatial semantics, and support realtime analysis. Here we discuss the data representation proposed to combine different data streams and operators designed to support analysis of these streams into situation descriptors.

The overview of the process of moving from heterogeneous streams to situations is shown in Figure 6.1. The unified STT format employed (level 1) records the data originating from any spatio-temporal coordinate using its value. Aggregating such data results in two dimensional data grids (level 2). At each level the data can also be characterized for analytics. The situational descriptor (level 3) is defined by the user (application expert) as a function of different spatio-temporal characteristics.

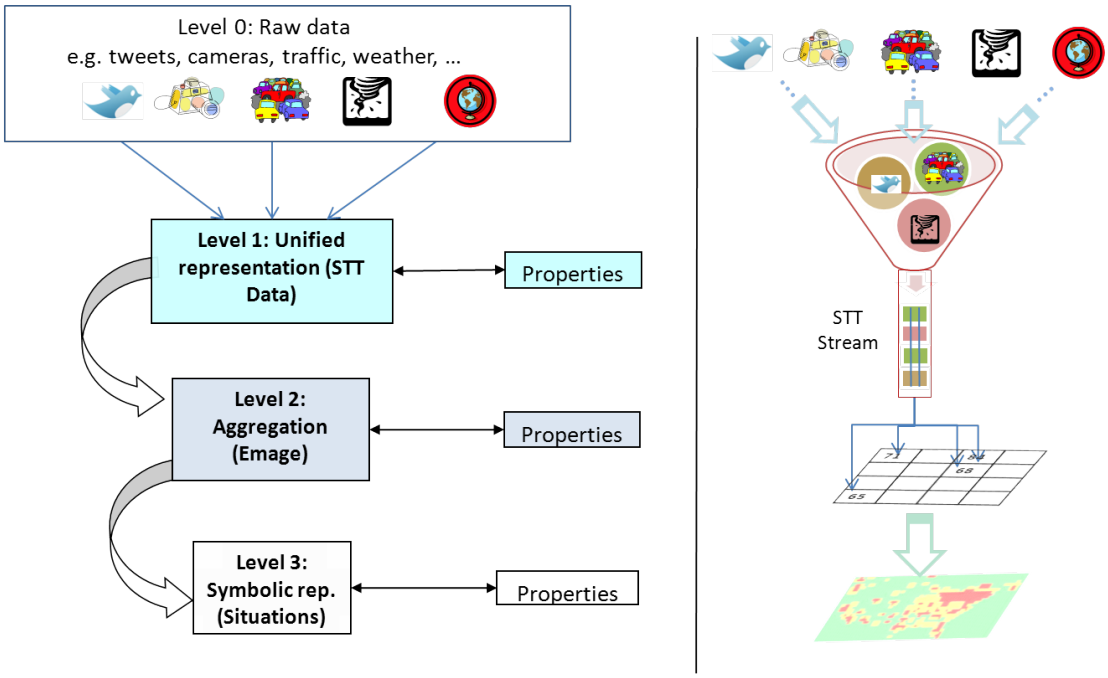

Figure 6.1: The workflow of data from raw streams to situation descriptors

6.1 Data representation

6.1.1 Data unification

Heterogeneous streams can be unified by focusing on the commonalities across them. We use Space, time, and theme (STT) as the fundamental axes to organize different types of data. All incoming data points are converted to, and represented as STT-Points. Each STTPoint has its coordinates in space and time, a theme, a value, and a pointer to raw data from which the value is derived.

$$STTPoint =< st_{coord}, \theta, value, pointer > \tag{6.1}$$

where:

$$st_{coord} =< lat, lon, alt, t > \tag{6.2}$$

wherein:

lat, **lon**, **alt**, represent the geographical latitude, longitude, and altitude (respectively) of the location corresponding to the data.
$lat \in [-90, 90]$, $lon \in [-180, 180]$, $alt \in [0, \infty]$

t represents the timestamp corresponding to the data.
$timestamp \in [0, \infty]$

θ represents the 'theme' e.g. 'Swine Flu', 'iPhone', 'Obama', to which the data corresponds. This theme is assigned by the application designer.

value belongs to a value set V which belongs to $\mathbb{N}$ i.e. natural numbers.

pointer is a link to the actual data (e.g. raw tweets from which interest level is derived). Certain application designers may want to keep a pointer to the actual data to support detailed analysis when required. This obviously has a trade-off with the privacy of an individual user, and may not be supported in all applications.

For the rest of our discussion here, we will focus on 2-D (latitude and longitude) spatial information, theme, and numerical values for each $STTPoint$ and ignore the altitude component, as well as the data pointer. Also, it is often desirable to consider only the spatial dimension of a STTPoint. We define a pixel as:

$$pixel =< \theta, x_1, x_2, value > \tag{6.3}$$

where x_1, and x_2 are the two dimensional spatial coordinates.

6.1.2 Spatiotemporal aggregation

STTPoints can be aggregated across space and time to yield different data structures.

E-mage

For each theme θ, let V be a value set and X a two dimensional point set. An V-valued e-mage on X is any element of V^X, which is a map from X to V. Given an V-valued e-mage $g \in V^X$, V is called the range of g and X the spatial domain of g. Here we use grid as the data structure representation of the e-mage. So

$$g = \{(\theta, \mathbf{x}, v(\mathbf{x}))|\mathbf{x} \in X = \mathbb{N}^2,\ and\ v(\mathbf{x}) \in V = \mathbb{N}\} \tag{6.4}$$

where:
$\mathbf{x}$ is the 2 dimensional spatial coordinate i.e. $\mathbf{x} = [x_1\ x_2]$, and
$v(x)$ is the value at coordinate $\mathbf{x}$.

We use $|g|$ to indicate the size of an e-mage, which is $(width, length)$.

Temporal E-mage Stream

Taken over time, a stream of E-mages can be defined as:

$$TES = ((t_0, g_0), ..., (t_i, g_i), ...) \tag{6.5}$$

where t_i is the timestamp of e-mage g_i.

Temporal Pixel Stream

Similarly a flow of pixels over time can be defined as a temporal pixel stream:

$$TPS = ((t_0, p_0), ..., (t_i, p_i), ...) \tag{6.6}$$

where p_i is a pixel.

6.2 Analysis operators (Situation recognition algebra)

As shown in Figure 6.1, the situation at any location is characterized based on spatio-temporal descriptors determined by using appropriate operators at level 2. The framework defines a set of generic operators to support this analysis. The operators work on the data representations described in the previous section. Temporal E-mage Streams (TES) act as first class citizens, and operators are applied in a standing-query format (i.e. they are continuously re-applied on each newly generated E-mage in any TES). The defined algebra builds upon and extends efforts in image operation algebra [49, 85].

We summarize the data representation terminology in Table 6.1 for easy reference while discussing the operators.

6.2.1 Filter (Π)

This operator acts as a filter to choose only the subset of data which is relevant to the user application. The user can define a predicate P which is applied every time

Term	Description	Representation
TES	Temporal E-mage Stream	$TES = ((t_0, g_0), ..., (t_i, g_i), ...)$
t	timestamp	
g	e-mage	$g = \{(\theta, \mathbf{x}, v(\mathbf{x}))\|\mathbf{x} \in X = \mathbb{N}^2, \ and \ v(\mathbf{x}) \in V = \mathbb{N}\}$
θ	theme of g	
X	Spatial domain of g	
V	Value range of g	
p	pixel in the e-mage	$pixel =< \theta, x_1, x_2, value >$
$\mathbf{x}$	spatial coordinate of p	$\mathbf{x} = [x_1 \ x_2]$
$v(x)$	the value at coordinate $\mathbf{x}$	

Table 6.1: Terminology used for data representation and operators

a new E-mage is generated in the TES. Predicate on e-mage g is a function on pixels $P(p)|p \in g$.

$$\Pi_P(TES) = ((t_0, \Pi_P(g_0)), ..., (t_i, \Pi_P(g_i))) \tag{6.7}$$

The predicate can define spatial coordinates or values. This operation also allows for normalization i.e. scaling the values to lie within a specified range.

Predicate on Space

A spatial predicate P_R works on each e-mage and uses a point lattice corresponding to region $R \subseteq X$. If the spatial coordinate of the pixel p is in R, then the value is copied over, else it is set to zero. The formal definition is

$$\Pi_{P_R}(g_i) = \{(\theta', \mathbf{x}, y)| \ y = v(x) \ if \ \mathbf{x} \in P_R; else \ y = 0\} \tag{6.8}$$

where θ' is the theme assigned to the new e-mage. It can be provided explicitly by the application designer or be generated automatically e.g. copied from incoming data stream, or be a hyphenation between the incoming theme and the operator. For the

remaining discussion in this chapter, we focus on the derivation of other components of g, and simply assume that the theme is copied over from the incoming data stream.

Predicate on Values

A value predicate P_v is a value comparison on pixel value $v(x)$. If the value satisfies the comparison, $P(v(x))$ is true; otherwise, false.

$$\Pi_{P_v}(g_i) = \{(\theta, \mathbf{x}, y) | \mathbf{x} \in X, and\ y = v(x)\ if\ P(v(x))\ is\ true; else\ y = 0\} \quad (6.9)$$

Value normalization

A predicate can also be applied to scale the values in the e-mage to lie within a new range $r = \{r_{min}, r_{max}\}$.

$$\Pi_{P_r}(g_i) = \{(\theta, \mathbf{x}, y) | \mathbf{x} \in X, and\ y = f(r, v(x))\} \quad (6.10)$$

where:

f is the function used to scale the values into the value range r.

Example:

1) Filter Temporal E-mage Stream TES based on a spatial predicate P_R

$\Pi_{P_R(region=M)}(TES)$

The format for the operators assignment is shown in Figure 6.2. The 'Operator cat-

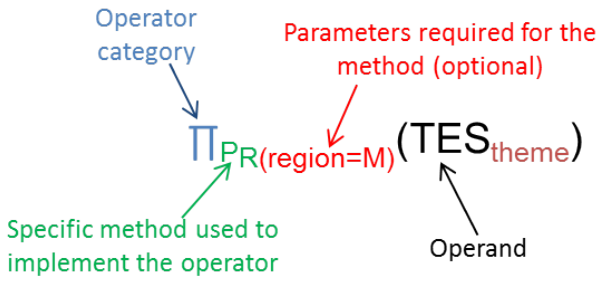

Figure 6.2: Operator syntax format

egory' identifies the main class of operation (i.e. *Filter, Aggregation, Classification, Characterization, or Pattern Matching*). 'Specific method' identifies the options used within the category (e.g. Predicate on Space, Predicate on Values, for Filter operation; max, min, average for Aggregation operation). Each such method also has an optional set of parameters (e.g. a region mask for Spatial Predicate) which the user must provide for describe the details of the operator. Here we assume that the system implementing the operators will be able to interpret any of the optional parameters discussed. The 'operand' identifies the data structure onto which the operator is applied. Note that sometimes we may add a theme to the representation (e.g. TES) as a short hand notation identify the data.

To illustrate the effect of this query we show how it would affect an active e-mage g_i of the TES. As can be seen in Figure 6.3, only the pixel values corresponding to lower half (based on the mask M) are maintained, and the rest are set to zero.

The empty cells in the Figure correspond to zero values at those pixels.

2) Filter Temporal E-mage Stream TES to maintain only values which are greater than 4.

$\Pi_{P_v(value>4)}(TES)$

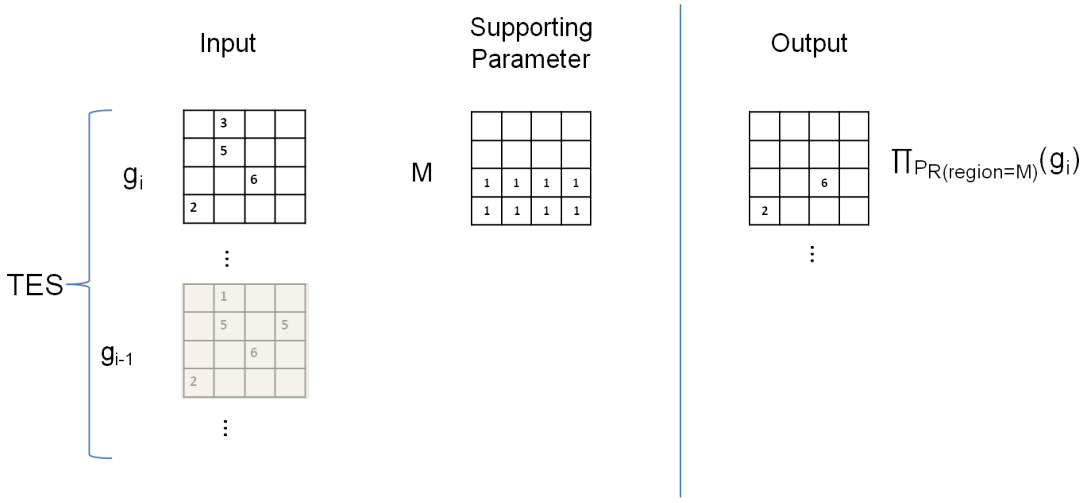

Figure 6.3: Example: Filtering operation based based on spatial predicate

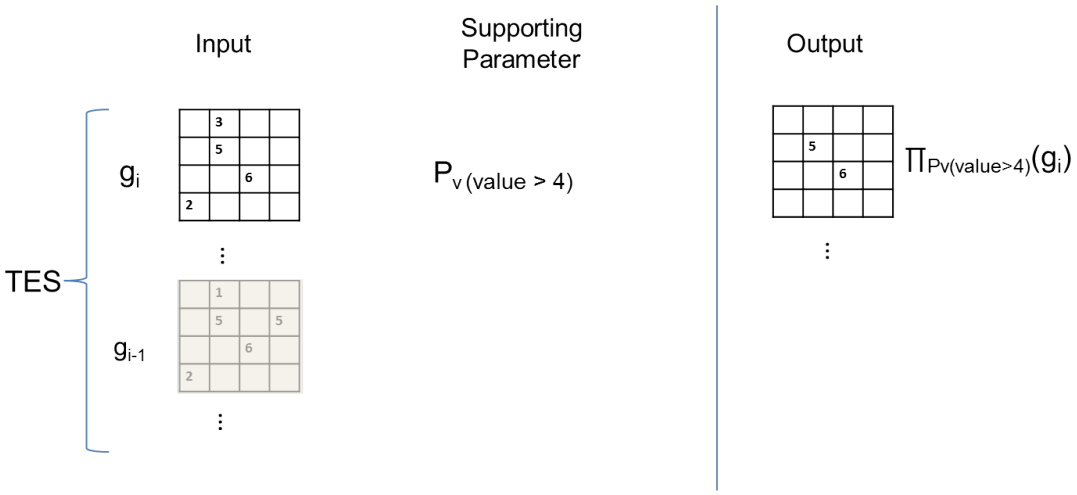

Figure 6.4: Example: Filtering operation based on value predicate

The effect of the operation is shown on a sample E-mage g_i in Figure 6.4. Notice that only the pixel values matching the criteria ($value > 4$) are maintained, and the rest are set to zero.

3) Filter Temporal E-mage Stream TES to scale the values into the range $r = \{0, 60\}$. $\Pi_{P_r(range=[0,60])}(TES)$

The effect of the operation is shown in Figure 6.5. Notice that the pixel values get

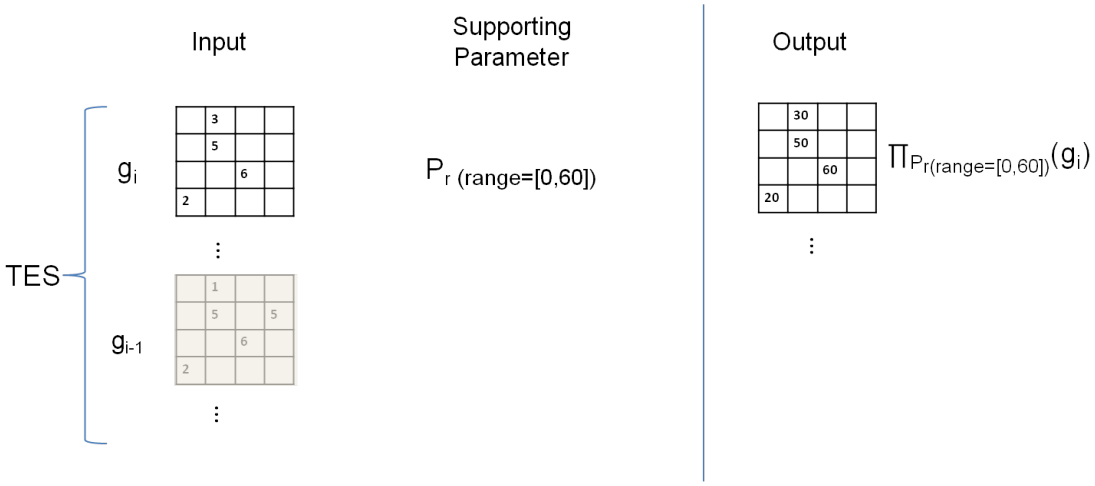

Figure 6.5: Example: Filtering operation to normalize values to a range

mapped from an initial range of 0 to 6, to the new range of 0 to 60.

6.2.2 Aggregation ($\oplus$)

Aggregation operations work on combining two (or more) E-mage streams.

Let us first consider the aggregation across two E-mages. The aggregation function takes two e-mages g_1 and g_2 as input, and generates a new e-mage g_3 as output. We assume that sizes of the e-mages are the same, i.e. $|g_1| = |g_2| = |g_3|$.

$$\oplus_f(g_1, g_2) = g3 = \{(\theta, \mathbf{x}, y) | x \in X, and\ y = f(v_1(x), v_2(x))\} \tag{6.11}$$

where:

v_1 and v_2 are the values at the same coordinate $\mathbf{x}$ in e-mages g_1 and g_2, and

$f \in \{add, subtract, multiply, divide, max, min, average, sum, convolution, and, or\}$.

Now, the definition can be extended to handle multiple TES, where e-mages at the

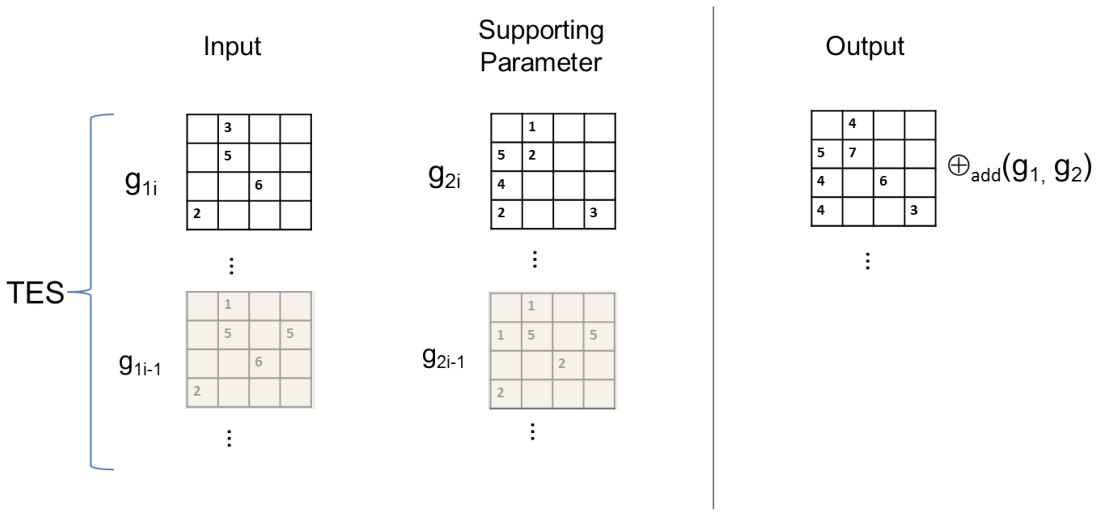

Figure 6.6: Example: Aggregate operation on two streams using add function

corresponding timestamps in different TES are processed as follows.

$$\oplus_f(TES_1, ..., TES_n) = (t_0, \oplus_f(g_{10}, ..., g_{n1})), ..., (t_m, \oplus_f(g_{1m}, ..., g_{nm})) \tag{6.12}$$

where g_{ij} is the E-mage at timestamp j in TES_i. While most of the operations $\{+, *, max, min\}$ can handle multiple inputs, some $(-, convolution)$ can handle only two inputs at a time.

Example:

1) Aggregate Temporal E-mage Streams TES_1 and TES_2, using addition.

$\oplus_{add}(TES_1, TES_2)$

The effect of the operation on sample e-mages is shown in Figure 6.6. Notice that the value at each pixel in the output e-mage is an arithmetic addition of the values at the corresponding pixels in the two input e-mages.

6.2.3 Classification (γ)

This operation segments pixels of a given e-mage g in TES into k different classes based on function f.

$$\gamma_f(g_i) = \{(\theta, \mathbf{x}, y) | \mathbf{x} \in X,\ y = f(\mathbf{x}, v(x), g_i)\ and\ y \in [1, k]\} \tag{6.13}$$

where:

f is the function which uses a combination of the pixel's location and value parameters and the global E-mage characteristics to assign the pixel to the right class.

k is the number of classes admissible.

The possible functions for f include kmeans, linear thresholding, affinity propagation, min-cut [90] and so on.

Over time this operation can be applied to each new e-mage and generate a classification for its pixels.

$$\gamma_f(TES) = ((t_0, \gamma(g_0)), ..., (t_i, \gamma(g_i))) \tag{6.14}$$

Example:

1) Classify Temporal E-mage Stream TES into 3 segments based on linear thresholding with thresholds set as 1 and 4.

$\gamma_{Linear\ Thresholding(thresholds=\{1,4\})}(TES)$

The effect of the operation on a sample e-mage is shown in Figure 6.7. Notice that the pixel values have been to set 1, 2 , or 3 which identify the class to which those

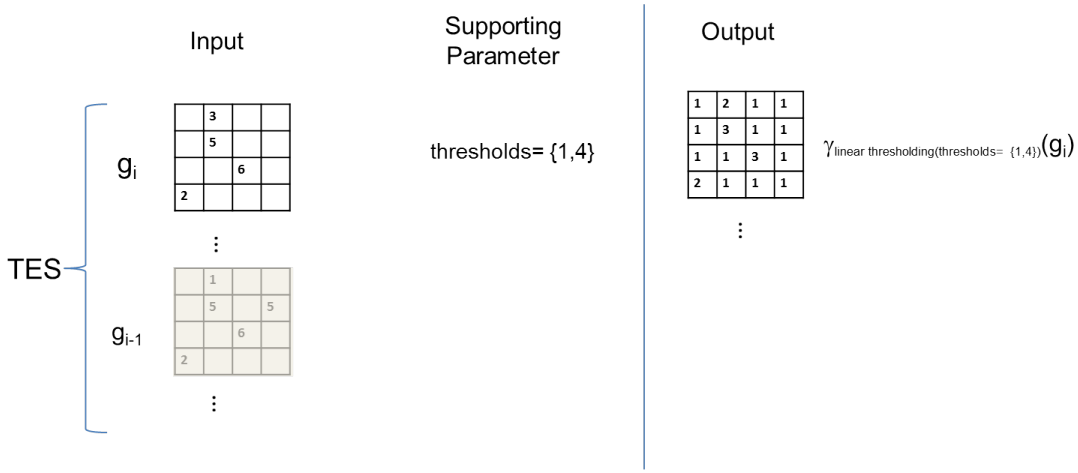

Figure 6.7: Example: Classification operation based on linear thresholding

pixels have been assigned. The class assignment corresponds to the thresholds set.

6.2.4 Characterization (@)

Spatial

E-mage characterization operation takes the active e-mage g in TES, and computes a pixel which identifies selected spatial property of the e-mage g based on a function f.

$$@_f(g) = p =< \theta, x_1, x_2, y >, \ where\ (x_1, x_2, y) = f(g), and$$
$$(x_1, x_2) \in X, \ and\ y \in \mathbb{N} \quad (6.15)$$

The function f can be selected from $\{count, max, min, sum, avg, epicenter, density\}$.

For functions such as max, min, $epicenter$, the coordinates (x_1, x_2) identify the location where the corresponding property is reached. For instance, when using max, the spatial coordinates and the value of the output pixel correspond to the E-mage

pixel whose value is the largest among all others in g.

However, other functions such as *count*, *sum*, *avg*, *density*, *shape*, are global features [98], and do not correspond to any specific location within the e-mage. Hence, the function produces only a value y, and the coordinates (x_1, x_2) are set to (0,0).

Over time this operation can be applied to each new e-mage and generate a corresponding pixel. Hence this operation can spatially characterize an evolving Temporal E-mage Stream (TES) and generate a Temporal Pixel Stream.

$$@_f(TES) = ((t_0, @_f(g_0)), ..., (t_i, @_f(g_i))) \tag{6.16}$$

Temporal

Temporal characterization operation takes a temporal pixel stream TPS, and computes a pixel output p_o which identifies the selected temporal property of the TPS in a time window tw based on a function f. The time window corresponds to the last k time cycles in the TPS yielding a windowed temporal pixel set $WTPS_i$.

$$WTPS_i = w(TPS) = \{(t_{i-k+1}, p_{i-k+1}), ..., (t_i, p_i)\} \tag{6.17}$$

where:

w is the windowing function.

t_i is the current time cycle,

p_i is the active pixel, and

k is the size of the time window.

As mentioned, the output pixel is computed based on the $WTPS$.

$$@_f(WTPS_i) = p_o =< \theta, x_1, x_2, y >, where\ x_1 = p_i.x_1,\ x_2 = p_i.x_2,$$
$$y = f(WTPS_i)\ and\ y \in \mathbb{N} \quad (6.18)$$

where $f \in \{displacement, distance, speed, acceleration, periodicity\}$.

Over time, different output pixels get generated yielding a new temporal pixel stream TPS_o

$$@_f(TPS_i) = TPS_o = ((t_i, @_f(WTPS_i)), (t_{i+1}, @_f(WTPS_{i+1})), ...) \quad (6.19)$$

Examples:

1) Characterize the epicenter of the Temporal E-mage Stream TES

$@_{spatial.epicenter}(TES)$

Note that the functions belonging to the spatial category are preceded by a keyword 'spatial.'.

The effect of the operation on a sample e-mage g_i is shown in Figure 6.8. Notice that the (x_1, x_2) coordinates for the output pixel have been set to (2,2) which correspond the epicenter of e-mage g_i. The coordinate system here starts with top left pixel being at (1,1). As epicenter is a spatial property, the value component $v(x)$ does not carry much importance, and here we assume that the function computing the epicenter sets it to zero.

2) Find the average speed taken over 3 cycles, for the epicenter of the Temporal E-mage Stream TES.

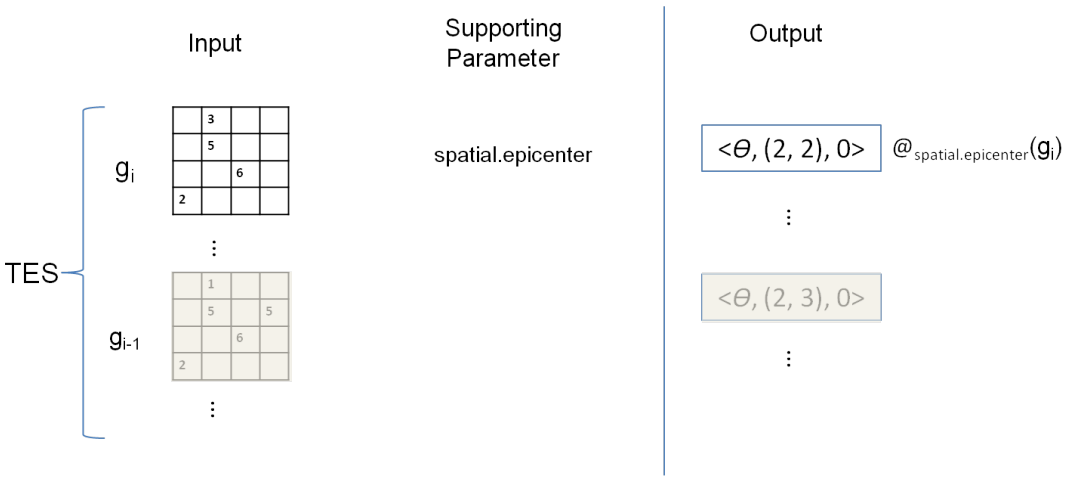

Figure 6.8: Example: Characterization operation based on spatial epicenter

$@_{temporal.speed(tw=3)}(@_{spatial.epicenter}(TES))$

Note again that the functions belonging to the temporal category are preceded by a keyword 'temporal.'. The effect of the operation on a sample TPS is shown in Figure 6.9. Notice that the value $v(x)$ of the pixel has been set to 0.47, based on the displacement observed over 3 time cycles. The coordinate values have simply been copied over from p_i.

Note also that this query builds upon the output of the previous query which yields a TPS with the epicenters.

6.2.5 Pattern Matching (ψ)

Pattern matching operation compares the similarity between a TES/TPS and a pattern, which can be defined from historical data or chosen from a library of relevant patterns (i.e. kernels) [98].

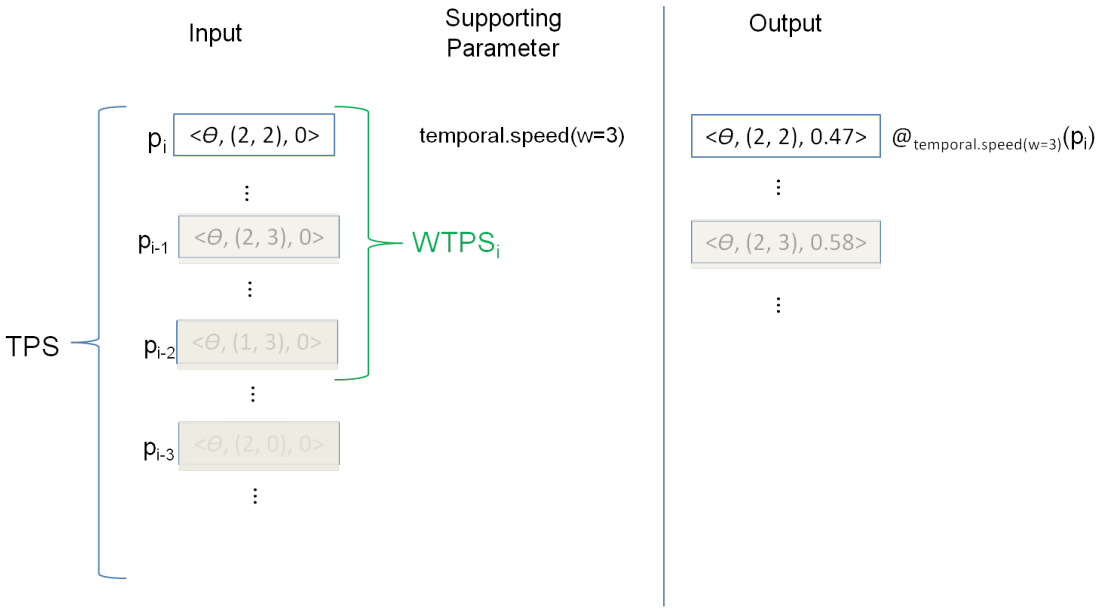

Figure 6.9: Example: Characterization operation based on average speed over 3 cycles

Spatial Pattern Matching

Spatial pattern matching compares every e-mage g_i in TES with a pattern e-mage κ, and defines a temporal pixel set where value at each pixel p_i represents the computed similarity.

$$\psi_f(g) = p =< \theta, x_1, x_2, y >, \ where \ (x_1, x_2, y) = f(g, \kappa), \ and$$
$$(x_1, x_2) \in X, \ and \ y \in \mathbb{N} \quad (6.20)$$

where f is a function which computes the similarity between the E-mage g and the kernel κ. Possible functions for f include *convolution, correlation, normalized correlation* and so on.

Over time this yields a temporal pixel stream.

$$\psi_f(TES) = ((t_0, p_0), ..., (t_i, p_i), ...) \quad (6.21)$$

Temporal Pattern Matching

Temporal Pattern matching operation takes in a temporal pixel stream TPS and a candidate temporal pattern κ to compute a pixel output p_o whose value identifies how similar κ is to TPS over a time window tw, based on a function f. The time window tw corresponds to the last k time cycles in the TPS stream yielding a windowed temporal pixel set $WTPS_i$, which is expected to be the same size as κ i.e. $|\kappa|= k$.

As earlier,

$$WTPS_i = \{(t_{i-k+1}, p_{i-k+1}), ..., (t_i, p_i)\} \tag{6.22}$$

where t_i is the active time interval, and p_i is the active pixel.

The output pixel is computed based on the $WTPS$.

$$\psi_f(WTPS_i) = p_o =< \theta, x_1, x_2, y > \ \ where\ x_1 = p_i.x_1,\ \ x_2 = p_i.x_2,$$
$$y = f(WTPS_i, \kappa)\ and\ y \in \mathbb{N} \tag{6.23}$$

where $f \in \{correlation, normalized\ correlation\}$.

Over time, different output pixels get generated yielding a new temporal pixel stream TPS_o

$$TPS_o = ((t_i, \psi_f(WTPS_i)), (t_{i+1}, \psi_f(WTPS_{i+1})), ...) \tag{6.24}$$

Examples:

1) Where and with how much similarity is a pattern matching with κ found in the

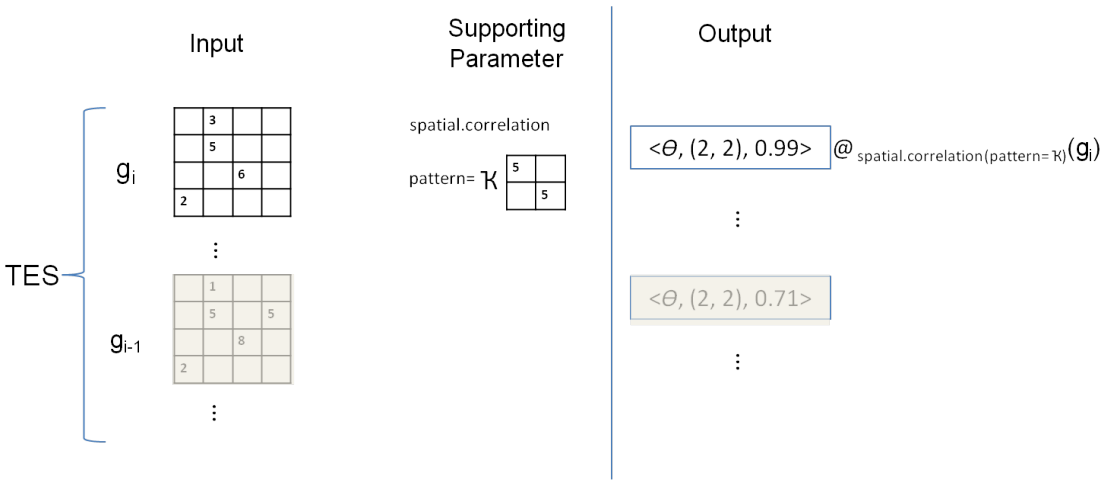

Figure 6.10: Example: Pattern matching operation based on a spatial pattern

Temporal E-mage Stream TES?

$\psi_{spatial.correlation(pattern=\kappa)}(TES)$

The effect of the operation on a sample e-mage g_i is shown in Figure 6.10. Notice that the coordinate (x_1, x_2) of the output pixel have been set to (2,2) which is the location of highest match, and the value has been set to 0.99 which represents the degree of similarity found.

2) How similar is the value increase in TES to the pattern κ?

$\psi_{temporal.correlation(pattern=\kappa,tw=3)}(TES)$

The effect of the operation on a sample pixel stream is shown in Figure 6.11. Notice that the value of the output pixel has been set to 0.99 which represents the degree of similarity found. The coordinate values (x_1, x_2) have simply been copied from p_i.

A summary of all the operators defined and the corresponding input and output data structures are shown in Table 6.2.

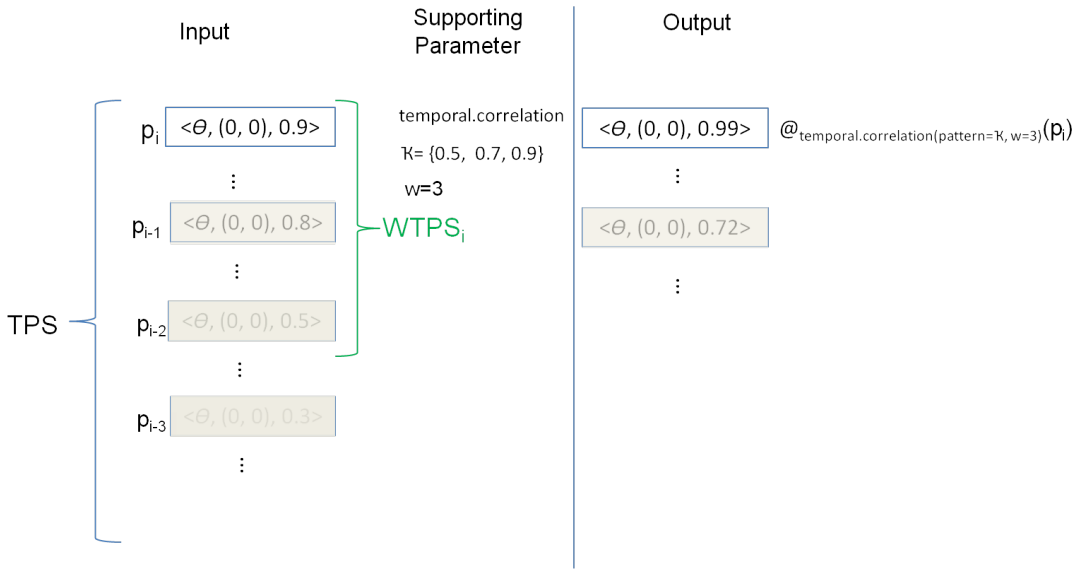

Figure 6.11: Example: Pattern matching operation using on a temporal pattern

S.No	Operation	Input	Output
1	Filter Π	TES	TES
2	Aggregate $\oplus$	$K \cdot$ TES	TES
3	Classification γ	TES	TES
4	Characterization @:		
	- Spatial	TES	TPS
	- Temporal	TPS	TPS
5	Pattern Matching ψ:		
	- Spatial	TES	TPS
	- Temporal	TPS	TPS

Table 6.2: Summary of various query operations. TES=Temporal E-mage Stream, TPS=Temporal Pixel Stream

6.2.6 Combining operators to create composite queries

The operators presented are designed to be declarative and basic building blocks which can be combined to support arbitrarily complex analysis. Here we see some examples of combinations of operators to pose different queries which can be used directly by analysts to derive insights or used as building block features for complex situation recognition.

1) Select e-mages for the theme 'Obama' corresponding to the region USA.

$$\Pi_{P_R(region=USA)}(TES_{Obama}) \tag{6.25}$$

Note that here we assume that any system implementing this operation will have access to spatial mask corresponding to the USA region.

2) Identify three different clusters for each E-mage above.

$$\gamma_{kmeans(k=3)}(\Pi_{P_R(region=USA)}(TES_{Obama})) \tag{6.26}$$

3) Show me the cluster with most interest in Obama.

$$\Pi_{P_v(value=3)}(\gamma_{kmeans(k=3)}(\Pi_{P_R(region=USA)}(TES_{Obama}))) \tag{6.27}$$

Such queries need not only be about political figures, but could also be about a hurricane e.g. 'Katrina'. Hence, we can identify the cluster showing highest level of activity related to theme 'Katrina' and go on to identify its epicenter, and how it moves over time.

4) What is the speed for high interest cluster for 'Katrina' e-mages?

$$@_{temporal.speed(tw=3)}(@_{spatial.epicenter}(\Pi_{P_v(value=3)}(\gamma_{kmeans(k=3)}(\Pi_{P_R(region=USA)}(TES_{Katrina})))) \tag{6.28}$$

5) How similar is pattern above to an exponential growth pattern generated with parameters: base=2, growth rate=3?

$$\psi_{temporal.correlation(pattern=generated,\ type=exponential,\ base=2,\ growthrate=3)}$$
$$(@_{temporal.speed(tw=3)}(@_{spatial.epicenter}(\Pi_{P_v(value=3)}(\gamma_{kmeans(k=3)}(\Pi_{P_R(region=USA)}(TES_{Katrina})))))) \tag{6.29}$$

Again we assume that the system implementing the query understands the function parameters (e.g. *pattern =generated, type=exponential, base=2, growthrate=3*).

Chapter 7

EventShop: Toward Interactive Situation Recognition

Based on the framework, we have developed a web-based platform called EventShop (http://auge.ics.uci.edu/eventshop/) that provides an easy way for different users to experiment with different data streams and recognize situations. This system operationalizes the various concepts promulgated in the framework. It provides an easy way to test, and refine situation models (Chapter 5), and does so using the data representation and operation algebra presented in Chapter 6.

EventShop provides operators for data stream ingestion, visualization, integration, situation characterization, and sending out alerts. The system can be graphically configured to interactively recognize different situations and undertake corresponding actions. It adopts a modular approach to make the system reconfigurable for different applications 'on the fly'. A simple graphical interface makes it accessible to non-technical users[1]. Hence, for the first time it provides non-technical users an

[1]The intended users are application designers. They need not be computer-science experts, but they are expected to have access to the application logic i.e. Situation Models

opportunity to experiment with real-time data streams coming from all parts of the world and integrate them for diverse applications; thus making one concrete step toward democratization of the process of situation-driven app-building.

EventShop includes a front end User Interface and a back end stream processing engine. EventShop draws inspiration from PhotoShop and provides an environment that allows users to apply different filters and operators to experiment with multiple layers of data until they are satisfied with the processing result. Just like PhotoShop moved image processing from *specialists* to *common-person* domain, EventShop aims to make real-time data processing and action-taking capabilities easy and available to all. EventShop is designed to allow its users to experiment with data sources and formulate queries by combining a rich set of operators without worrying about the underlying technical details.

Screenshot from EventShop is shown in Figure 7.1. The basic components are:

(a) **Data-source Panel:** To register different data sources into the system.

(b) **Operators Panel:** To show the different operators that can be applied to any of the data streams.

(c) **Intermediate Query Panel:** A textual representation of the intermediate query currently being composed by the user.

(d) **Registered Queries:** A list of completed queries registered with the system.

(e) **Results Panel:** To see the output of the query (which can be presented on a map, timeline, as a numeric value or a combination of them).

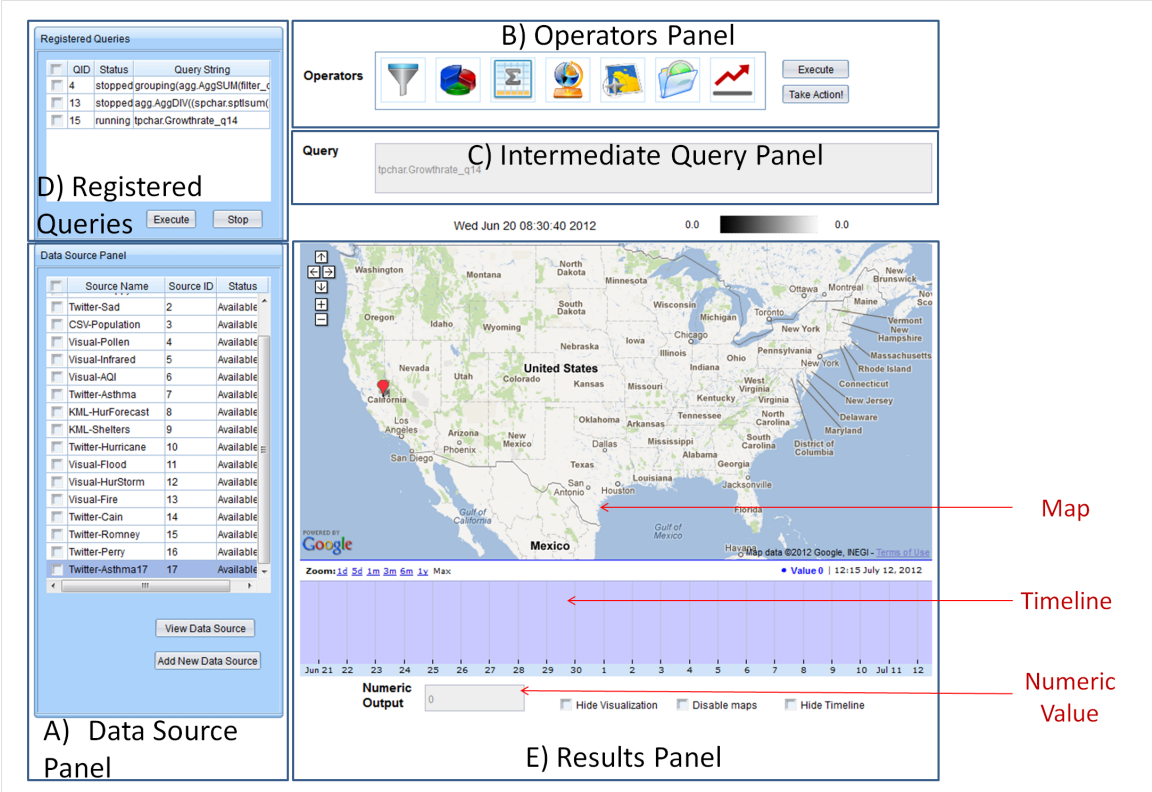

Figure 7.1: Screenshot of EventShop system

7.1 System design: overview

The EventShop system architecture is shown in Figure 7.2. EventShop provides a graphical interface that allows end users to register new data stream sources and formulate queries by combining a rich set of built-in operators. Users are also provided with a GUI tool that allows them to send personalized alerts to relevant people. In the back end, data sources and queries requested from the front end are stored into data source and query databases. Based on the information of registered data sources, EventShop continuously ingests spatio-temporal-thematic data streams and converts them to E-mage streams. Meantime, directed by the registered queries, EventShop pulls E-mage streams from data ingestors in to the query processor, which then processes the E-mage streams in each of the instantiated query operators. Besides being converted to E-mage streams, the raw data stream (e.g. tweet stream) is also persisted into raw data storage. This raw data can be combined with situation query results to define action conditions in the Personalized Alert Unit.

The front end GUI of EventShop is implemented in JavaScript, and sends requests to the back-end controller through a set of Ajax calls. The back-end controller to respond to these requests is implemented using Java Servlets. The Data ingestor component is implemented in Java. The implementation of runtime operators makes use of the OpenCV package [16] and is written in C++.

In the following discussion, we focus on the description of EventShop's back end design. The back end system consists of three major components: data ingestor, stream query processor, and personalized action alert unit. We will defer the detailed discussion on personalized action alert unit to the next Chapter.

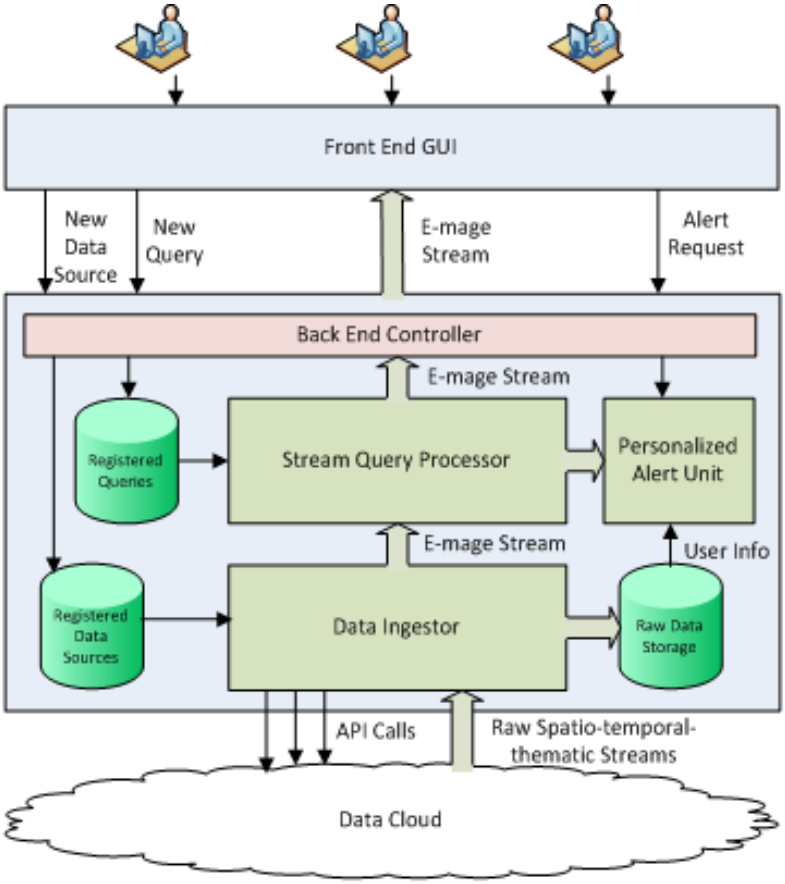

Figure 7.2: System Architecture of EventShop

7.2 Data Ingestor

Each time a new data source is registered and inserted into the data source database, the back end controller creates a new instances of STTPointIterator as well as EmageIterator (as shown in Figure 7.3) and adds them to the data ingestor. The data ingestor then connects to the data source and takes the raw data stream as input and relies on these iterators to convert the raw data stream into an E-mage stream. Note that the focus of EventShop is not on very sophisticated analysis of a single stream (e.g. not on sentiment or natural language processing based analysis of a twitter stream, or 'object recognition' in a Flickr stream) but rather on the ability to combine, analyze, and interpret different streams to recognize situations which can not be recognized by individual data streams. EventShop comes with ingestors for deriving values from some commonly used data-sources, and it expects more sophisticated individual wrappers to be developed by third party contributors as required. Note though that these third party contributors will be different from non-technical application designers (e.g. health care expert) who will focus only on configuring applications via wrappers made available by (first-party or third-party) developers.

Details of iterators are described in Section 7.2.2.

7.2.1 Data Sources

A data source registered by end users needs to include the following information to enable the data ingestion process:

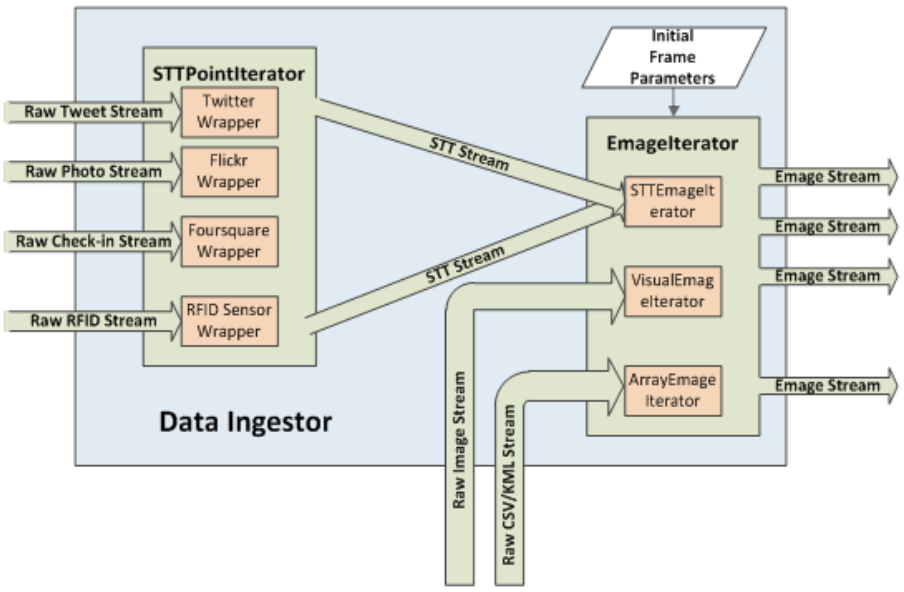

Figure 7.3: System Architecture of Data Ingestor

1. Theme: which is the central topic discussed in a data source. For example, hurricane, asthma, population, temperature, shelter locations etc.

2. Resource Locator: which is the API access link of a data source. For instance, Twitter opens its Stream and Search API which allow users to query the tweet stream. Sensor data, such as traffic sensors deployed by PeMS (Caltrans Performance Measurement Systems), is also often available online for user access as well.

3. Data Type: Data sources provide raw data streams in different formats.

   - **STT data streams:** In the case of Twitter, Facebook, and a variety of sensors, a single STT data point (e.g. a tweet, a status update, or a temperature reading) is the unit that is generated and appended to a data stream.
   - **Geo-image streams:** Some data sources aggregate their spatio-temporal-thematic data and provide them only in geo-image format, e.g. pollen

count data ($http://pollen.com/images/usamap.gif$). These geo-images get updated across time windows (e.g. every 5 mins for NASA satellite images, and every 24 hours for pollen count data).

- **Array Data Streams:** Alternatively, some data sources provide the data collected in a time window in an array format, such as in CSV(comma-separated values) or KML(Keyhold Markup Language) structures. This type of data gets updated across time windows and essentially forms an array stream.

A data source needs to declare itself as one of the types above.

4. Type Specific Parameters: Type specific parameters are also required in the data collection process.

   - For a raw *spatio-temporal-thematic data stream*, users need to specify the attributes that they are interested in. For Twitter and other social media sources, this can be a bag of keywords that covers a specific theme. For traffic and temperature sensors, attributes such as average speed, lane occupancy rate, max. temperature, and other measures recorded at sensors, can be specified.
   - For a *geo-image stream*, users can either specify the geo coordinate system adopted by the original data sources or provide transformation matrices that can be used to convert original geo images to E-mages which follow the equirectangular projection system [112]. To map image colors to values in E-mages, users can choose between converting images to grey-scale E-mages or assigning values to certain bins of image colors.
   - For an *array data stream*, depending on the data format, users are required to specify field names or tag names from where the spatial coordinate and value of a data point can be extracted.

5. Frame Parameters: The output of the data ingestor is an E-mage Stream. Parameters that are sufficient for specifying the size, resolution and generation frequency of an E-mage (or a frame, borrowing the concept of 'frame' from video) are necessary in the creation of an E-mage stream. A set of frame parameters specifies the size of the time window (e.g. 10 seconds, 1 hour, 1 day), the synchronization time point (e.g. creating an E-mage at every 10th second, or at 6AM everyday), the resolution in latitude (e.g. 0.01 lat), resolution in longitude, spatial boundary including southwest and northeast latitude and longitude values (e.g. for the US, the southwest point is (24, -125), and the northeast point is (50, -66)). That is,

$$FP =< window, sync, latUnit, longUnit, swLat, swLong, neLat, neLong > \tag{7.1}$$

The set of frame parameters that is used to guide the framing of a raw spatio-temporal-thematic data stream to create an E-mage stream for a data source is called its Initial Frame Parameters. We will later introduce the notion of Final Frame Parameters, which are required by each formulated query.

7.2.2 Iterators

As shown in Figure 7.3, a data ingestor is comprised of two types of iterators, STT-PointIterator and E-mageIterator.

1. **STT Point Iterator**

   A STTPointIterator is a generic iterator that generates one single STTPoint for each raw data point in spatio-temporal-thematic data stream, and outputs the

STTPoint at each 'next()' function call. We call a specific STTPointIterator for a data source a *wrapper*. For example, one can create a wrapper for a hurricane related tweet stream, or create a wrapper for a Foursquare check-in stream at a hospital. For example, the currently implemented Twitter wrapper parses each incoming tweet in the Twitter stream of a specific topic θ (e.g. hurricane), through the meta-data associated with the tweet, and identifies the time and location of the tweet, to generate one STTPoint i.e. (tweet location, tweet time, θ, 1). The value in the resulting STTPoint is decided by the logic adopted in the wrapper which can be simple (e.g. a count value) or sophisticated (e.g. a user sentiment value). EventShop expects all values to be real numbers. EventShop expects all observed values to be real numbers i.e. $v \in \mathbb{N}$. The responsibility of maintaining the data in the right units, and normalized range lies with the application designer, who can encode these when designing the application logic.

In the future, we will open APIs that allow third party developers to implement their own wrappers to ingest more data streams.

2. **E-mage Iterator**

   Guided by the initial frame parameters, an E-mageIterator generates one E-mage at each time window, and stores the E-mage in its internal buffer until query processor pulls the E-mage by calling 'next()' function. As described above, based on the data source type, E-mages can be generated from STT streams, geo-image streams and array streams.

   - **STT E-mage Iterator.** By iterating and combining collected STTPoints from the STT stream of a data source, the STTE-mageIterator generates an E-mage stream for the data source. Based on the initial frame parameters and the spatial and temporal coordinates stored in a STTPoint, a STTPoint is spatially mapped to a cell in the result E-mage. Suppose the

original STTPoint is collected at (lat, long), and the target cell is cell (i, j). Now given the initial frame parameters FP,
$i = \left\lceil \frac{long - FP.swLong}{longUnit} \right\rceil$, $and\ j = \lceil \frac{lat - FP.swLat}{latUnit} \rceil$ The value at the cell(i, j) is normally the sum of values of all the STTPoints that are mapped to the cell. Depending on the applications, however, the aggregate could also be $max, min, average$. The system defines a set of aggregates from which users can select to combine values.

- **Geo Image Stream Iterator.** If data points from a data source are already aggregated as a geo-image, an E-mage can also be created or imported directly from the STTPoints in these geo-image formats. The result E-mage has $\lceil \frac{neLat - FP.swLat}{latUnit} \rceil$ rows, and $\left\lceil \frac{neLong - FP.swLong}{longUnit} \right\rceil$ columns, and the value at a cell(i, j) in E-mage is computed from the original or normalized values at the pixels of the original geo-image that are projected to this cell. Computation of the projected area depends on the geo coordinate projection system.

- **Array Stream Iterator.** Similar to STT E-mage Iterator, each single value in the array is associated with a spatial coordinate that can be used to decide the E-mage cell to which the value is mapped. This value extraction part is continuously repeated at the end of each time window to obtain newer values from the resource location.

7.2.3 Handling different types of data

The list of data types supported by EventShop is extensible. However the current version provides built-in wrappers for the following data types.

Twitter

Users can specify a 'bag of words' to gather tweets related to any topic. The Twitter API supports the selection of tweets based on these terms as well as the geo-location of the tweets. We use Twitter4j (http://twitter4j.org) as an intermediary library for creating the wrapper. The value at the STTPoint is set to the count of the tweets containing the bag of words. More sophisticated wrappers, e.g. based on sentiment analysis or natural language processing can be developed by interested parties and plugged into EventShop as required.

Flickr

Similar to Twitter, users can specify a 'bag of words', and the value is set to the number of new posts related to a particular topic being posted from any location. We use FlickrJ (http://flickrj.sourceforge.net) as an intermediary library to create the wrapper.

Simulator

In order to simulate data for some of the types not currently available (especially at very high streaming rates), we have implemented a wrapper simulator. It creates semi-random data based on Gaussian distributions, four of which are centered around New York, Seattle, Los Angeles and San Francisco areas.

Geo-image streams

EventShop supports the ingestion of web data made available as maps or geo images. Examples of such data include satellite imagery, weather maps (*http://www.weather.com*), pollen count data (*http://pollen.com/images/usa˙map.gif*), pollution related data (*http://www.aqi.gov*), and data from many other map like presentation interfaces. Translation of such data into the equi-rectangular E-mage format requires a transformation matrix [110]. To aid this, the users can either specify the geo coordinate system adopted by the original data sources or provide their own transformation matrices. To aid the users, we have built a separate tool to determine such matrices. To map image colors to values in E-mages, users can choose between converting images from grey-scale E-mages or assigning values to certain bins of image colors.

CSV (Comma separated values)

EventShop can also ingest other general data uploaded by the users or available on a Web URL. This data needs to be a specified format i.e. contain columns corresponding to the location, time-stamp, and the value parameters.

KML (Google's geographical data format)

KML (Keyhole Markup Language) is a specialized XML format with a schema created to support various geo-constructs. The provided wrapper can parse this data using standard XML libraries. The KML format is expected to use specified tags for space, time, and value parameters.

MySQL archives

In order to be inclusive to offline (archived) data, the system also supports reading data from MySQL databases. The additional parameters required are:
$< HistoricalBeginTime, EndTime, TableName, DataType, DataRefreshRate >$.
The 'DataTypes' supported are 'Data table', 'Visual, aggregated data', and 'Region, aggregated data'. The data tables are expected to be in STT (i.e. spatial location, timestamp, theme, value) format. A 'Visual aggregated data' query points to the location for the archived geo-image, which is thereafter handled the same as the geo-images discussed above. The 'Region, aggregated data' option combines the values read from the table with a system-maintained description (in KML) of the boundaries of that region. The system currently only supports US states and metropolitan areas within California.

The MySQL wrapper works on JDBC (Java Database Connectivity) and could be modified to support other database systems (e.g. Oracle, MS SQLServer) as required.

7.3 Stream Query Processor

In contrast to traditional one-time queries issued to a database, a query in EventShop is a 'standing query'. A standing query needs to be registered and instantiated in the system *before* relevant data streams flow into the system and get processed.

The system architecture of the EventShop query processor is shown in Figure 7.4. For each operator of each registered query, the back end controller creates an operator instance that performs the computation. Next, back end controller connects these operator instances as defined in the logical operator tree of the query and forms a runtime operator tree in the stream query processor. Then the controller pulls E-

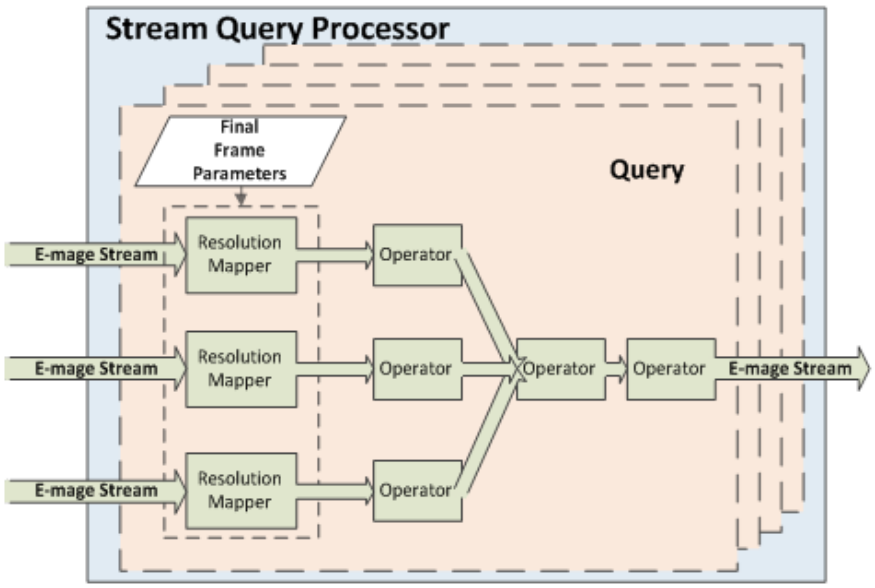

Figure 7.4: System Architecture of Query Processor

mage streams from the E-mage Iterators of the appropriate data sources, and feeds the streams to the runtime operator tree. Each operator instance processes the E-mage streams pulled from upstream operators and buffers the results in its internal buffer. The E-mage stream output from the last operator instance is the output of the entire query.

We now discuss the details of the process of query formulation and the operators supported by EventShop.

7.3.1 Query

A query registered by an end user needs to specify two pieces of information, a Final Frame Parameters and a logical operator tree.

Final Frame Parameters

Frame parameters specify the size, resolution and generation frequency of an E-mage (refer to Equation 7.1). When multiple queries need to access the same data source but combine them with other data sources at different spatial bounding boxes, resolutions or time windows, the E-mages pulled from data ingestors need to be first mapped to E-mages which conform to the final frame parameters. Therefore, every query specifies a set of final frame parameters, which should be followed by all input E-mage streams.

EventShop system has a *Resolution Mapper (RM)* component that takes an E-mage stream of a data source, the corresponding initial frame parameters, and the final frame parameters as requested by a query as its inputs; it then generates a new E-mage stream following the final frame parameters. In the RM component, EventShop allows users to select the strategy to perform this frame mapping. If E-mages at coarser frame parameters are mapped to finer frame parameters, the set of available strategies includes *interpolation, repeat,* and *split*. If the mapping is performed from finer to coarser frame parameters, the users can choose strategies like *sum, max, min, avg,* and *majority* for conversion.

Logical Operator Tree

The logical operator tree for a given query specifies the operators used and their order. Nodes in the operator tree represent configured operators (refer to Section 7.3.2), and directed edges between nodes denote the the E-mage flows from upstream to downstream operators. In EventShop, users sequentially configure operators as part of issuing their queries. An example of a logical operator tree is shown in Figure 7.5. In this example, the overall situation query involves the application of

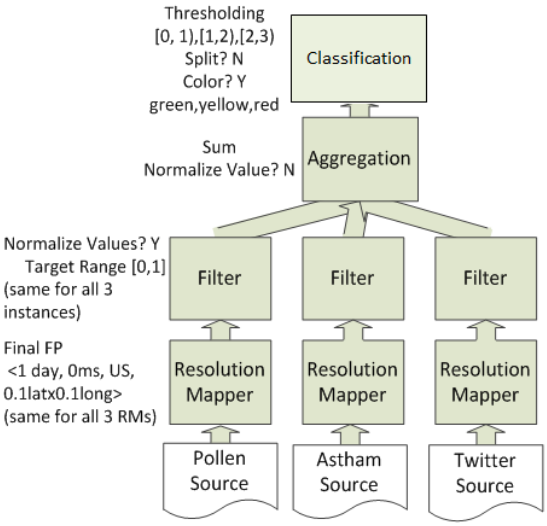

Figure 7.5: Operator tree for a situation query

'Filter' operation on three data sources, followed by their 'Aggregation', and finally a 'Classification' on the aggregated value.

This operator tree is parsed, instantiated and converted to a runtime operator tree by the back end controller. The runtime operator tree is then added to the stream query processor to do the actual processing.

7.3.2 Operators

Operators take E-mage streams from upstream operators as input, combine and process them, and create a new E-mage stream as output. Physically, operators are instantiated as E-mage Iterators. Each operator in the system maintains a buffer that stores the E-mages processed by it. Downstream operators continuously check and pull the next E-mage from the buffers. The final E-mages are stored in the buffer of the last operator until they are pulled by the user front-end.

We have implemented the operations described in Chapter 6. These operators are defined in a closed algebraic structure that can be easily combined to form complex

queries. In the future, we plan to provide interfaces to allow users to define their own customized E-mage operators.

We have implemented the query operators using an underlying media processing engine. Each query operator corresponds to one or more classes of media processing operations (under OpenCV) as shown in Figure 7.6. As can be seen, multiple query operations (e.g. 'Characterization', and 'Pattern matching') may employ the same media processing operation (e.g. 'Convolution') in terms of the underlying implementation. For example both 'Circularity' (which is a 'characteristic' from a user perspective) and Pattern matching with a library of Kernels, use the convolution operation. However, they are different operators from a user perspective.

A summary of the operators currently supported in EventShop is presented as Figure 7.7. Here, we discuss the detailed parameter settings of each operator. While the list of options/ configurations for each operator is extensible, here we review just the currently implemented options.

Filter

The Filter operator takes an E-mage stream as input, and filters each E-mage in the stream based on: *a value range, a spatial bounding box, or a time interval.* Additionally, values of an E-mage can be normalized into a new range of values by using the Filter operator. This allows the value ranges of different data streams to be made comparable before undertaking any further analysis.

S.No	Query Language Operator	Media processing Operator Category	Media processing Operator Details
1.	Filter		
	-Spatial	Arithmetic	AND with the spatial mask
	-Temporal	Arithmetic	AND with the temporal mark
	-Thematic	Arithmetic	=
	-Value	Arithmetic	AND, >, <, =
2.	Aggregation		
	-Max, Min, +,-,%,*	Arithmetic	Max, Min, +,-,%,*
	- NOT, OR, AND,	Logical	NOT, OR, AND
	-Convolution	Convolution	Convolution
3.	Classification		
	- Predefined segments count	Segmentation	K-means
	- Predefined segment boundaries	Segmentation	thresholds
4.	Characterization		
	i) Spatial		
	- Count, Min, Max, Sum, Average, Variation	Statistical	Count, Min, Max, Sum, Average
	- Coverage	Arithmetic	Count
	- Epicenter	Arithmetic	Weighted average
	- Circularity	Convolution	Scale free convolution with known circular kernel
	- Growth rate	Arithmetic	+, -, %
	ii) Temporal		
	- Displacement, Distance, Velocity, Acceleration, Growth rate	Arithmetic	+, -, %, *
	- Future estimation	Arithmetic	Multiplication with Kernels based on users choice e.g. linear, progression exponential growth
	- Periodicity	Convolution	Auto correlation i.e. Self convolution with time-lagged variant.
5.	Pattern Matching		
	- Scaled Matching	Convolution	Convolution with user defined or pre-defined Kernels
	- Scale free Matching	Convolution, Statistical	Maxima from Loops of Convolution with different image sizes.

Figure 7.6: Mapping of Situation recognition Algebra to Media Processing Operations

Operator	Input	Output	Parameters
Filter	E-mage Stream	E-mage Stream	• Value range predicate • Time interval predicate • Spatial bounding box predicate • Normalize Values? Y/N Yes, Target Value Range
Aggregate	K * E-mage Stream	E-mage Stream	• Aggregate: {max, min, sum, avg, sub, mul, div, and, or, not, xor, convolution} • Normalize Values? Y/N Yes, Target Value Range
Classification	E-mage Stream	E-mage Stream	• Grouping Method ❖ K-Means ✓ Number of segments ❖ Thresholding ✓ Threshold of each segment • Split? Y/N • Color? Y/N ❖ Yes, Color code for each segment
Characterization : Spatial	E-mage Stream	STTPoint Stream	• Spatial Characteristics: {max, min, avg, sum, epicenter, coverage}
Characterization : Temporal	STTPoint Stream	STTPoint Stream	• Time Window Size • Temporal Characteristics: {displacement, velocity, acceleration, periodicity, growth rate}
Pattern Matching: Spatial	E-mage Stream	STTPoint Stream	• Pattern Input Method ❖ From file ❖ Create a new one ✓ Number of rows ✓ Number of columns ✓ Distribution ▪ 2D Gaussian ▪ 2D linear • Normalize pattern size? Y/N • Normalize pattern value? Y/N
Pattern Matching: Temporal	STTPoint Stream	STTPoint Stream	• Time Window Size • Pattern Input Method ❖ From file ❖ Create a new one ✓ Sampling rate ✓ Pattern duration ✓ Distribution ▪ Linear ▪ Exponential ▪ Periodic • Normalize pattern size? Y/N • Normalize pattern value? Y/N

Figure 7.7: Configuration Options for different operators in EventShop

Aggregate

The Aggregate operator combines E-mages from two or more E-mage streams using arithmetic and logical operations. The E-mages must have the same dimension, and E-mages are aggregated per cell. For example, for each (i, j) pair, the *sum* aggregate adds cells(i, j) of the E-mages from each input stream and stores the value at cell(i, j) of the new E-mage. Again, note that the frame parameters followed by these E-mages need to be the same to be combined. In addition, the system allows one or more operands of an aggregation operator to be a scalar value or a 2D pattern, in which case every E-mage in E-mage streams can be combined with the scalar value (i.e. the same value is assumed at each cell) or the pattern.

Aggregates supported are $max, min, sum, avg, sub, mul, div, and, or, xor$, and *convolution*. Some aggregates, such as *sub* and *div*, can only be applied between two E-mages. Users can also choose to normalize values of the resulting E-mages.

Classification

The Classification operator segments each E-mage in an E-mage stream based on the chosen method. The methods currently supported are *kmeans* and *linear thresholding*. For *kmeans*, users need to specify the number of groups, and the appropriate value threshold is automatically selected by the system. For *linear thresholding*, users provide the thresholds for each class. Users can specify whether to split the segmented E-mage into multiple E-mages or create a single E-mage with cell values corresponding to the assigned segment. If the result E-mage is not split, the users can select a color code for visualizing each segment.

Characterization

EventShop supports two types of Characterization operation - spatial and temporal.

(i) **Spatial**

The spatial characterization operator works on a single E-mage stream and computes a spatially relevant property of each E-mage in the E-mage stream. The output is an STTPoint stream. Each STTPoint in the output stream stores the spatial coordinate where the measure is taken along with the associated value.

The following characterization measures are allowed: $max, min, sum, avg, epicenter$, and $coverage$. For some characteristics spatial-location is not relevant (i.e. $sum, avg, coverage$) and thus set to (0,0).

(ii) **Temporal**

The temporal characterization operator takes a window of STTPoints from a STTPoint stream and computes its characteristics like *displacement, velocity, acceleration, periodicity, and growthrate*. The output of this operator is again a STTPoint stream.

The time window (in seconds), over which the measure is taken for each of these options needs to be chosen by the user.

Pattern Matching

Again, EventShop supports two variants of this type of operator - spatial and temporal.

(i) **Spatial**

The spatial pattern matching operator takes E-mages from an E-mage stream and a two-dimensional pattern as its inputs and it tries to match the pattern in the E-mage. The output of this operator is an STTPoint stream. The output STTPoints record the location of the highest match along with the corresponding similarity value.

The 2D pattern for the spatio-temporal pattern matching operator can be input in two ways, either *uploaded* from an image file or *generated* by the system. Image files in most of the common image formats, such as bmp and png, are allowed as input. (See Section 9.3.3 for an example based on the use of an uploaded image to detect a hurricane pattern in satellite data.)

The system also allows users to create some commonly used 2D patterns. Users need to specify the resolution (i.e. number of rows and number of columns of the pattern) and a spatial distribution (including *Gaussian2D* and *Linear2D*). To generate a Gaussian pattern, the center point coordinate, x,y variance and the amplitude are needed. For a 2D linear pattern, a starting point, starting value, directional gradient and value gradient are required.

Users can again choose to normalize the pattern resolution as well as the values for matching purpose.

(ii) **Temporal**

The temporal pattern matching operator takes a 1D pattern and a window of STTPoints from an STTPoint stream as input and tries to match the pattern over the window of STTPoints. The output of this operator is an STTPoint stream where STTPoint stores both the center position of the sub window of STTPoints where the pattern matches with the highest similarity value and the similarity value itself.

Users specify the window size in seconds, over which the pattern is to be

matched.

The pattern can be input from a file (CSV format) or generated by the system. Currently, we allow *linear, exponential, and periodic patterns*. For generating a pattern, users specify the *sampling rate* and the *duration*. For example, the sampling rate could be 1 value per 5 seconds, and the whole pattern duration is 30 seconds. (See Section 9.3.3 for an example of temporal pattern matching.) For a linear pattern, the parameters include its *slope* and *Y-intercept*. Exponential patterns need the *base value* and the *scale factor*, and Periodic patterns use *frequency*, *amplitude*, and *phase delay*.

Similar to spatial pattern matching, users can choose to normalize the *pattern size*, *pattern value*, or *both*.

Note that in the current system, we handle spatio-temporal operators (e.g. temporal pattern matching on velocity of epicenter of a hurricane) by applying temporal operators on the outputs of spatial operators. See Section 9.3.3 for an example.

7.4 Presentation of results

The results of an Eventshop query are presented on the results panel (refer to Figure 7.1). 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 7.1.

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 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 7.1: Output formats for different types of queries

The presentation of an EventShop output E-mage on the Google Map interface for easy visualization requires two adaptations:

1. **Value normalization:** The output E-mage values are scaled between 0 and 255 to support natural visualization.

2. **Translation to Mercator Projection system:** Google maps uses the Mercator projection system, which is different from the Equi-rectangular projection system used by EventShop. Hence the E-mages must be projected to it. [2]

7.5 Discussion

EventShop acts as a platform to operationalize the proposed framework for situation recognition. It allows for easy translation of situation models into situation-based applications. Different models can be tested, revised, and enhanced until the results meet the application requirements (see Section 9.3 for case studies).

EventShop implements the operators described in Chapter 6, and it implements

[2]The x and y coordinates of a point on a Mercator map can be derived from its latitude κ and longitude λ as follows:
$x = R(\lambda - \lambda_0)$, and $y = R(ln(tan(\frac{\pi}{4} + \frac{\kappa}{2}))$.
The number λ_0 is the longitude for x = 0. R is the radius of the sphere of the Earth (6378.1 km) at the scale of the map as drawn, and κ and λ are given in radians.

the building blocks for the modeling approach described in Chapter 5 [3]. In effect EventShop provides all of the components (i.e. *data stream selection, ingestion, unification, aggregation, situation evaluation, visualization, and personalized alerts*) needed in order to support the generic approach to situation recognition discussed in Chapter 4 .

[3]Note: The operator for 'learning' (Chapter 5) has been left out of the current implementation.

Chapter 8

Personalization and Alerts

While macro-situations (e.g. 'epidemic level') are of interest to the analysts/policy-makers, individual users benefit most from realtime personalized alerts. Personalization ensures that the information is relevant to the end-user receiving it, and alerts allow them to take immediate actions based on the situations recognized. Hence the framework needs to provide an ability to:

1) *Personalize* the recognized macro situations, and

2) *Alert* users in real-time to take the appropriate actions.

This allows different situation-aware application built using the framework to not just passively observe, but rather act and respond to aid human lives and resources. This changes the focus from just analysis, or future planning, to actually undertaking real-time control to affect the evolving situation. Alerting people to respond in real time can allow them to evacuate risky areas, safeguard important resources, save lives of cattle and poultry, and escape unhealthy situations.

The approach for handling personalization and alerts is as shown in Figure 8.1. Different data streams are combined to recognize macro situations (i.e. situations detected

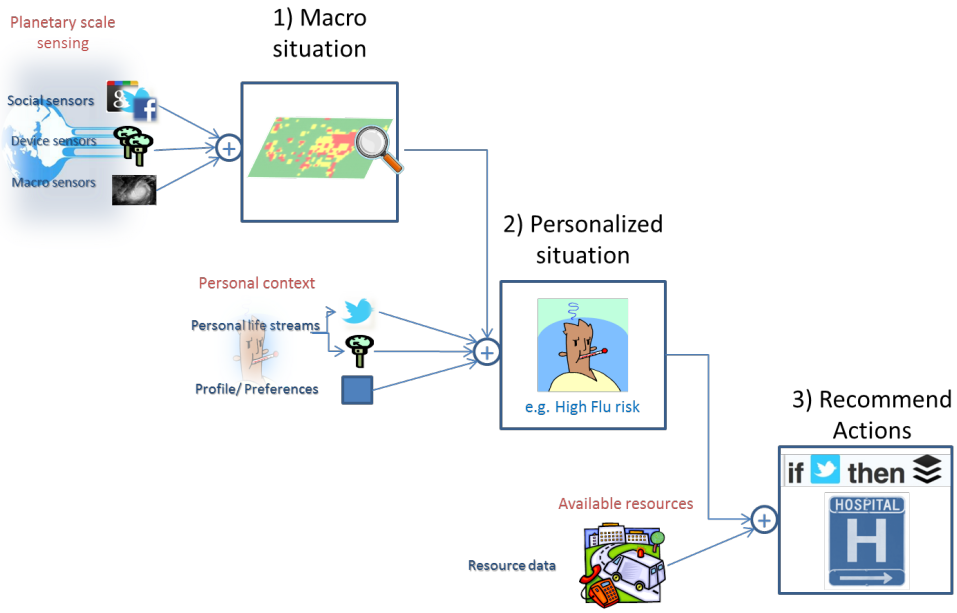

Figure 8.1: Approach to recognizing macro situations, personalized situations, and sending alerts

over aggregated spatio-temporal data). These macro situations can be combined with personal context, profile, and preferences for each user to recognize personalized situations. Each personalized situation can be mapped to relevant action recommendation. Each layer builds on the output from the previous layer. In this Chapter we focus on step 2 (personalizing situations) and step 3 (alerts).

8.1 Personalized Situations

Definition

Personalized situation: An actionable integration of a user's personal context with surrounding spatiotemporal situation.

To aid the modeling and recognition of personalized situations, we consider:

1) Data Types and representation

2) Operators

3) Modeling steps

8.1.1 Data types and representation

The data sources used for personalized situation recognition can be of three types. The first two are stream based and highly dynamic, while the third one changes infrequently.

1. *Macro Situation:*

   Situation values detected by combining multiple data streams from a 'macro' view point. These situation values can be projected onto the user's ST coordinates.

2. *Personal life streams:*

   - Sensor streams: Device based data streams reporting values about a particular user e.g. temperature, heart rate.
   - Personal media streams: Personal user account on social media streams can be used to detect various micro-events in the person's life e.g. coughing, sneezing, activity level.

3. *User Profile + Preferences:*

   - Profile: Any data about the user's personality which may be relevant to the situation classification.
   - Preferences: User's preference as applicable to application relevant parameters.

S.No	Data Type	Source	Nature	Examples
1	Macro Situation	Combination of streams	Streams	Pandemic risk index = high
2	Personal life streams	Sensors	Streams	Temperature = 99.1 F
		Personal media	Streams	Sneezing severity = mid
3	Profile + preferences	Profile	Persistent	Gender = male
		Preferences	Persistent	Preferred medical coverage = Kaiser Permanante

Table 8.1: Examples of different data types

The representation of all of these data types can be in the form of Temporal E-mage Sets, or Temporal Pixel Sets as discussed earlier in Chapter 6. We explicitly assume that each data nugget comes inscribed with spatio-temporal parameters.

8.1.2 Operators

The operators supported for combining these streams are similar to those defined for macro situation recognition (Chapter 6). However the focus now is to view the data from a single user perspective rather than spatiotemporal aggregates. This implies two important differences. First, STTPoint/Pixel streams, rather than Temporal E-mage streams, become the primary data representation. Second, this implies that operators focusing on the analysis of spatial aggregates (i.e. spatial characterization and pattern matching) will not be relevant here.

The semantics of the operations however remain very similar to those in Chapter 6. Rather than going through each operator in detail here we simply recap the relevant operators, and summarize their corresponding input and output data structures in Table 8.2.

To differentiate between the operators defined from a personal user perspective and those defined from a macro perspective, we apply a superscript 1 on the personalized situation recognition operators e.g. op^1.

Filter (Π^1)

This operator acts as a filter to choose only the subset of data which is relevant to the user application. This operator works the same as defined in Chapter 6 - except the focus is now on spatial filtering to generate a user-location based TPS from the TES input.

Aggregate ($\oplus^1$)

Aggregation operations work on combining two (or more) Pixel streams via arithmetic and logical operations like $+, *, max, min$.

Classification (γ^1)

This operator segments incoming data (TPS) into different categories based on functions like linear thresholding, kmeans, affinity propagation.

Characterization: Temporal ($@^1$)

Temporal characterization identifies the selected temporal property (e.g. displacement, distance, speed, acceleration, periodicity, growthrate) of the TPS in a selected time window.

Pattern Matching: Temporal (ψ^1)

Pattern matching operation compares the similarity between a TPS and a selected pattern based on functions like correlation, normalized correlation.

S.No	Operation	Input	Output
1	Filter Π^1	TES	TPS
2	Aggregate $\oplus^1$	$K \cdot$ TPS	TPS
3	Classification γ^1	TPS	TPS
4	Characterization $@^1$:		
	- Temporal	TPS	TPS
5	Pattern Matching ψ^1:		
	- Temporal	TPS	TPS

Table 8.2: Summary of various query operations. TES=Temporal E-mage Stream, TPS=Temporal Pixel Stream

```
/*  Input:  Personalized  Situation  descriptor  */

/* Output: List of intermediate descriptors, data sources, and operations required */

Get_components (v){

1)  Identify output state space
        • Numeric or Classes
2)  Identify component features; v =f(v1, …, vk)
3)  ForEach (feature vi){
        • If (atomic)
            • Identify Data source.
                • Type, URL, ST bounds
            • Identify operations.
        • Else
            • Get_components(vi)
}
```

Figure 8.2: Steps in personal situation modeling

8.1.3 Modeling Personalized Situations

The process of modeling the personalized situation is very similar to that described in Chapter 5. Basically the domain experts need to keep splitting the possibly complex and vague situation descriptor into its component features until each one of those can be derived by simply applying an operator on a data source.

Also note that this modeling needs to happen from a user-based (not macro) perspective, and after the macro situation modeling has been completed. Hence spatio-temporal bounds are no longer a required in this modeling, and at least one of the

features (i.e. $v_1, \ldots, v_k$) is expected to be the output of the macro situation classification at the user's location.

8.2 Alerts

The proposed framework provides an approach to send out alerts to people in different situations. The alert may need to provide different types of information to tackle problems in different scenarios. Here we adopt the perspective defined under Social Life Networks [55, 54], where the focus is on '*connecting people to the right resources based on personalized situations detected*'. In effect, it focuses on aiding most basic human needs (i.e. Level 1 of Maslow's hierarchy - food, water, health) via automated recognition of situations.

The problem of matching users to resources can be formulated as follows:

$$\Gamma : (U \times Z) \rightarrow (U \times R) \tag{8.1}$$

where:

$U = \{u_0, \ldots, u_M\}$ is the set of users,

$R = \{r_0, \ldots, r_N\}$ is the set of resources,

$Z = \{z_0, \ldots, r_O\}$ is the set of personalized situations,

and Γ is the function which takes users and their personal situations as input, and gives out appropriate $\{u, r\}$ i.e. user-resource pairs as outputs.

We consider the data types, representations, and the approach to undertake the matching.

8.2.1 Data types

The above mentioned perspective necessitates the use of three types of data sources

1. *Situation recognition sources:* Data sources used to recognize the macro or personal situation.
2. *User list sources:* Data sources which provide a list of users.
3. *Resource list sources:* Data sources which provide a list of resources (e.g. shelter spaces, ambulance, park). Note that detected situation E-mages can also act as resources (e.g. healthy situation area).

All of these data sources are expected to be available or transformable into the STT format (correspondingly TES/TPS) in this framework.

8.2.2 Situation-Action Rules

We adopt a rule based approach to match users to resources (i.e. to handle the Γ function of eq. 8.1). It works as follows:

IF u_i *is_in* z_j, THEN connect(u_i, $nearestLoc(u_i, r_k)$).

Supporting such rules requires *matching* the user to the right resource type r_k, and then *connecting* her to the nearest resource location. Let us consider each of these operations.

Matching

The matching process can be defined as follows:

$$is\_in(u_i, z_j) \implies match(u_i, r_k) \tag{8.2}$$

where:

u_i is the i^{th} user. z_j is a personalized situation, and r_k is the resource.

is_in returns a Boolean output stating whether the user u_i, falls under the personalized situation z_j, and

match creates a mapping between users and the resource types needed.

The framework expects the matching rules to be defined by the domain experts based on the application.

Find nearest resource

Similarly, identifying the nearest location for the r_l can be represented as:

$$nearestLoc(u_i, r_k) = \underset{Loc(u_i), Loc(r_{kp})}{argmin^{p=1...P}} dist(u_i, r_{kp}) \tag{8.3}$$

where:

$Loc(u_i)$ is the location of the user,

$Loc(r_{kp})$ is the p^{th} location for the k^{th} resource e.g. (k^{th} resource = 'Hospital'; $p^{th}location$='Hoag Hospital, 12, 4th St., Irvine, CA-92617')

P is the total number of locations for the resource r_k, and

dist is the Euclidean distance.

The nearestLoc function can be handled automatically by the system once the match-

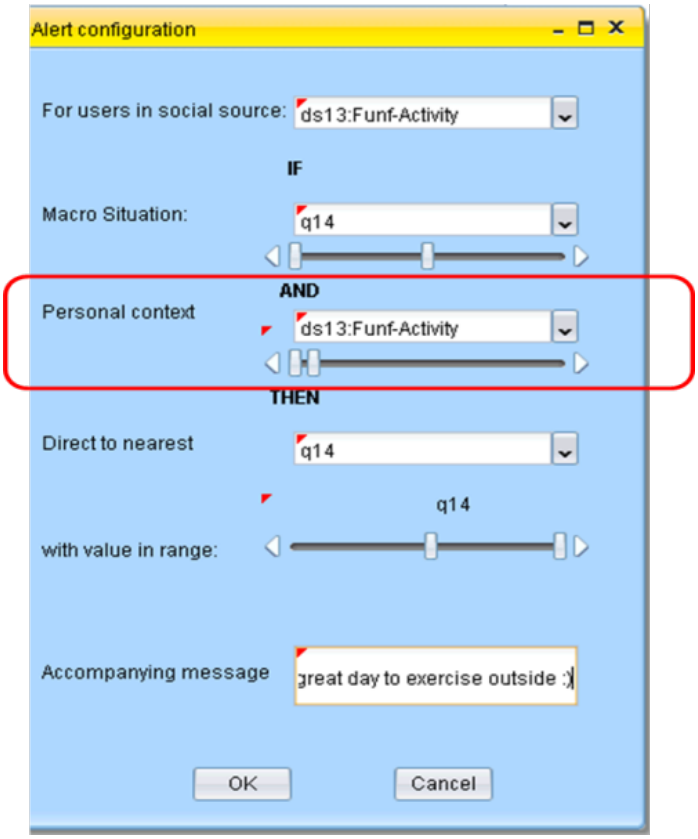

Figure 8.3: A snapshot of the Personalized Alert Unit

ing rules have been defined by the application designer. A collection of situation-action rules can be registered identify the right resource for each user.

These rules can be considered an adaptation of E-C-A (Event Condition Action) [72] rules, for handling situation based applications.

8.3 EventShop support for personalized alerts

Support for personalization and alerts is provided in EventShop via its Personalized Alert Unit. Figure 8.3 shows an example via a screenshot.

The Personalized Alert Unit works on Situation-action rule templates that can be

configured as follows. If the user belongs to a location with a prescribed macro-situation, AND she satisfies defined personal conditions, then she can be directed to the nearest location matching certain other situation conditions and sent an alert message. The configurable options are:

1) User data stream: The personal media stream to be used for identifying users and sending out alerts.

2) Macro-situation: As defined by a composed situation query and a value range on it.

3) Personal context: Defined to be a value range on the selected personal media/sensor stream.

4) Desired Resource: Defined as a single resource stream or a composite situation stream matching the proscribed value range.

5) Message: The alert to be sent out to the users.

Note that the current version of EventShop provides only limited support for personalized alerts. The 'And' operator is available to combine macro situation and personal data-stream, and Twitter is available to send out alerts. Based on the experience gained by implementing and testing multiple applications, we plan to extend these capabilities in the next development phase of EventShop.

8.4 Example: Asthma/ allergy recommendation system

Let us consider an application that recognizes the allergy risk level for different location in the US. It then advises highly-vulnerable people to stay indoors, while

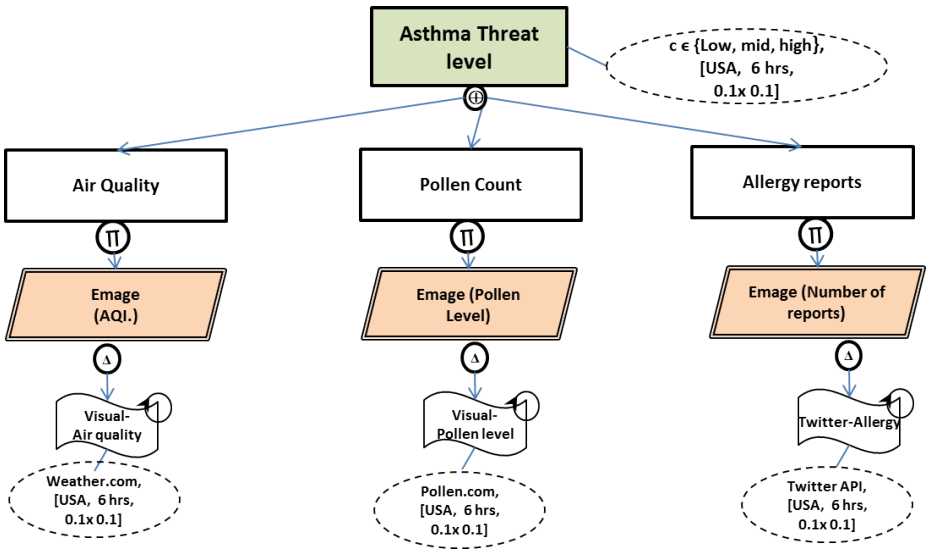

Figure 8.4: Situation Model for Asthma threat level

prompting those in healthy environments to enjoy the outdoors (e.g. to go jogging at the nearest park). Building such an end-to-end application requires three steps, which we detail next.

8.4.1 Defining macro situation

The first step is defining the macro-situation, which can be modeled following the discussion in Chapter 5. On an experimental basis we can define the allergy risk for an environment based on the pollen count, air quality, and the number of asthma reports on social media (human sensors) in the neighborhood and create a situation model as shown in Figure 8.4.

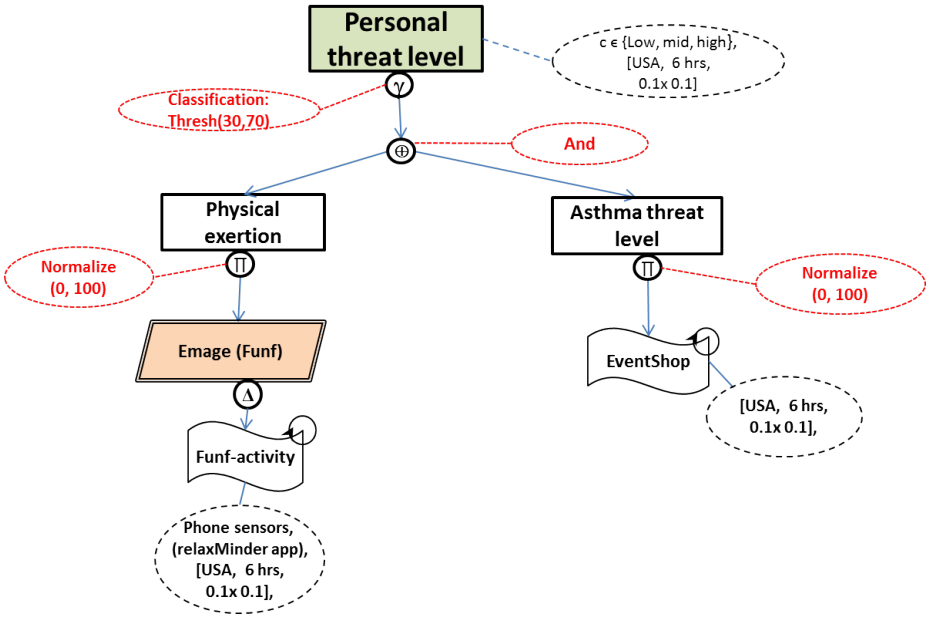

Figure 8.5: Situation Model for personal threat level

8.4.2 Defining personalized situation

For experimental purposes, the personal threat level can be defined based on combining the environmental threat level with the personal exertion level. The exertion level can be derived from the sensor stream from user's mobile device. This can be modeled as shown in Figure 8.5.

8.4.3 Situation-action rules

To send out alerts which will connect users to the appropriate resources, we configure two situation-action rules.

1) To encourage people in good situations to go outdoors:

IF u_i *is_in (Personal Asthma Threat = low), THEN connect*(u_i, *nearestLoc*(u_i, *Park*)).

where *(Personal Asthma Threat = low)* can be defined in EventShop based on a

combination of the macro asthma threat and the user's personal activity level. Additionally, we can configure an accompanying messages, e.g. 'Nice day to go jogging!', in the personalized alert unit as was shown back in Figure 8.3.

2) To encourage people in risky situations to stay indoors:
IF u_i *is-in (Personal Asthma Threat = High), THEN connect(*u_i*, n/a).*
with an accompanying message 'stay indoors!'

Once configured, this acts as a real-time situation monitoring and alerting application for allergy alleviation. Multiple such rules can be registered for defining various applications, and multiple such applications can be created using the framework.

Chapter 9

Evaluations

In this chapter we discuss experiments and case studies undertaken to validate the different ideas proposed in this dissertation. We split the discussion into three sections.

We first test two of the design features of this work (humans as sensors, and space and time as organization axes) to see if they make sense in the various real world scenarios. Next we check if the data representation (E-mages) and the situation recognition algebra is expressive and relevant enough to provide situational insights across multiple applications. Lastly we test the ability of the holistic framework to support a design process of creating situation-based applications - from situation modeling, to situation recognition, and personalized alerts. We pay particular attention to the use of these applications in real world scenarios, and we discuss the ability of the framework the meet the design goals set forth in Chapter 2.

9.1 Validating the design principles

9.1.1 Using humans as sensors

The use of humans as sensors in this framework builds upon a hypothesis that human reports on social media (e.g. Twitter, Facebook, Foursquare) are relevant and can be used for detecting real world events and situations.

To test this hypothesis we ran the following experiment. We selected a list of 10 important events that happened in the USA (or Canada) during the period of Nov 2009-Mar 2010. We tried to be diverse in both the category of events as well as the physical location. We ran the peak/ max (in time) and epicenter (in spatial location) characterization operators on a relevant Twitter corpus. The data in the corpus was obtained using two sources. The Twitter streaming API was used to download a portion (10%) of all public Twitter feeds. While some Twitter posts are directly GPS geo-coded, we geocoded the rest by using the 'home' location (e.g. San Francisco, CA) of the user by using an opensource geocoding service ($http://ws.geonames.org$). Only the tweets successfully geocoded were used for the experiment. We augmented this data set using location based queries for each location across the US for selected topics. To counter the effect of high Twitter user bases in different parts of the country we performed 'background subtraction' (with an e-mage composed by averaging the first day of the month posts for all topics from different locations).

As shown in table 9.1, we found that the temporal peak coincided with the actual event occurrence period for all 10 events. The spatial peak matched precisely for 7, and was a nearby big city for 2 more, out of 10 events. Only for the Winter Olympics event, was the peak (Toronto) not within a reasonable distance of the original location (Vancouver). While a more detailed analysis is required to understand such

S.No	Category	Event	Physical Date	Observed Temporal Peak	Physical Location	Observed Spatial Peak
1	Politics	Health Care Bill passed	2010-03-21	2010-03-21	38.89, -77.03 (Washington)	41, -74
2	Politics	California Prop 8, Trial Day 1	2010-01-11	2010-01-11	37.77, -122.41 (San Francisco)	38,-122
3	Society	Fort Hood Shootings	2009-11-05	2009-11-05	31.13, -97.78 (Fort Hood, TX)	33,-97
4	Society	SeaWorld Whale Accident	2010-02-12	2010-02-12	28.54, -81.38 (Orlando, FL)	29,-81
5	Sports	Winter Olympics Opening ceremony	2010-02-12	2010-02-12	49.24, -123.11 (Vancouver)	44,-79
6	Sports	Baseball World Series final	2009-11-04	2009-11-04	40.71, -74.00 (New York)	41, -74
7	Entertainment	Oscars	2010-03-07	2010-03-07	34.05, -118.24 (Los Angeles)	34, -118
8	Entertainment	South by Southwest festival	2010-03-12 to 2010-03-21	2010-03-15	30.26, -97.74 (Austin, TX)	30, -98
9	Tech. Conv.	CES 2010	2010-01-05 to 2010-01-07	2010-01-06	36.17, -115.13 (Las Vegas)	34,-118
10	Tech. Conv.	TED 2010	2010-02-10 to 2010-02-13	2010-01-10	33.76, -118.19 (Long Beach,CA)	34, -118

Figure 9.1: Real World Events and their recognition via human sensor reports

phenomena, we found the results to highlight a reasonable correlation between real world events and the related reports generated by humans.

Hence analyzing human reports can indeed be useful for detecting real world events and situations.

9.1.2 Space and time semantics of social media data

Another important design feature of the framework is the use of space and time as the axes to organize different types of data. While physical phenomena are known to exhibit spatio-temporal correlation [48, 5], we posit that social media data also exihibits spatio-temporal patterns just like real-world event data. Specifically, we test if social media data for various events also exhibit spatio-temporal power laws like those observed in seismic and other 'physically grounded' data streams.

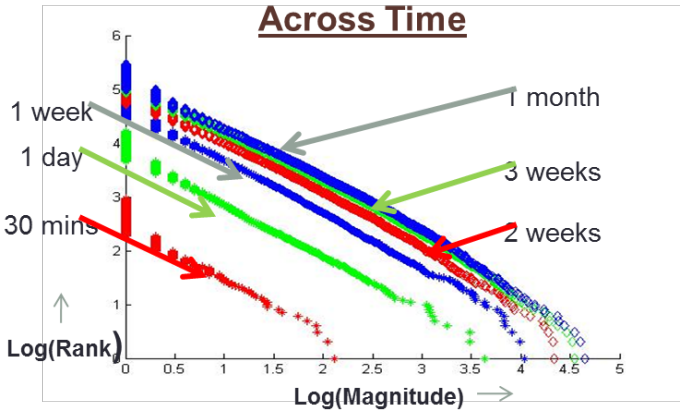

Figure 9.2: Variation of frequency of hashtags and their ranks for different time durations

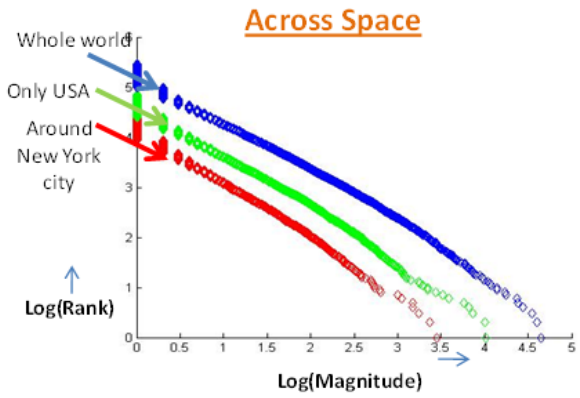

Figure 9.3: Variation of frequency of hashtags and their ranks for different geo-regions

Here we present the results of analysis of a corpus of 5.6 Million tweets over a period of one month based on the hashtags (taken to be proxy for event tags) associated with them [96]. We found that human report patterns on Twitter also exhibit spatio-temporal power laws (akin to Gutenberg-Richter's Law which has been well studied for physical phenomena like earthquakes [48]).

We ranked different event-tags based on their frequency of occurrence, and plotting the rank of the tags against the frequency of occurrence on a Log-Log scale gave a linear plot. More interestingly (See Figure 9.2), when we took subsets of the data

corpus for different time durations (30 mins, 1 day, 1 week, 2 weeks, 3 weeks, and 1 month), we observed separate but still linear (and with very similar slope) plots for each one of them. Further still (see Figure 9.3) when we took subsets of data based on location (whole world, just United States, and just a 10 latitude by 10 longitude block around New York city), each collection independently exhibited the Gutenberg-Richter law.

These observations provide support for the inherent spatio-temporality of social media data and their correlation across space and time with other 'physical' data streams.

9.2 Validating the data representation and analysis operations

We undertook experiments to validate the data representation (E-mage streams) and Situation recognition Algebra as being expressive enough and sufficiently relevant to gain situational insights across multiple applications. The three applications considered are for business intelligence, political event analytics, and seasonal pattern analysis. The aim is to test the expressiveness of the algebra in posing meaningful queries and to validate the data representation and operators with regard to their ability to yield appropriate results to those queries.

The results presented here are based on a large data corpus of ($>$ 100 million) tweets collected using the 'Spritzer' stream (since Jun 2009) and then the higher rate 'Gardenhose' stream since Nov, 2009. The seasonal pattern analysis application uses a collection of over 700,000 Flickr images. These applications were tested using a Matlab based implementation of the operators and did not involve personalized action-taking. The analysis was done on an offline corpus wherein Temporal E-mage Streams can

be considered as Temporal E-mage Sets.

9.2.1 Application: Business analysis

We considered some sample queries which can be asked by a business analyst, when dealing with spatio-temporal data, about a product of interest (P) (e.g. 'iPhone').

1. When did the interest peak in our product across USA?
   $@_{temporal.max}(@_{spatial.sum}(\Pi_{P_R(region=USA)}(TES_P)))$
2. Segment the above e-mages into three zones with varying interest levels.
   $\gamma_{kmeans(k=3)}(\Pi_{P_R(region=USA)}(TES_P))$
3. Where are the epicenters for each 'zone' of product interest?
   $@_{spatial.epicenter}(\gamma_{kmeans(k=3)}(\Pi_{P_R(region=USA)}(TES_P)))$
4. Show me the best location to open a new store for product P given the existing store locations as an e-mage S, and the 'catchment' area kernel for each store as an e-mage C.
   $@_{spatial.max}(\oplus_{convolution}(\oplus_{Subtract}(\oplus_{Add}(\Pi_{P_R(region=USA)}(TES_P)),$
   $(\oplus_{Convolution}(\Pi_{P_R(region=USA)}(TES_S), TES_C))), TES_C)$

The data considered in this experiment is for Jun 2 to Jun 15, 2009, which included the date of Iphone 3G version's release (Jun 8). The results of applying queries 1 through 3 are shown in Figure 9.4. As can be noticed (amongst other things), the peak of interest does indeed match with the actual release date.

For easier understanding, we visually illustrate (see fig. 9.5) query 4 and show a sample result. The query aims to find the best location to start a new store for 'iphone'. In this example, e-mages corresponding to the term 'iPhone' were *added* for

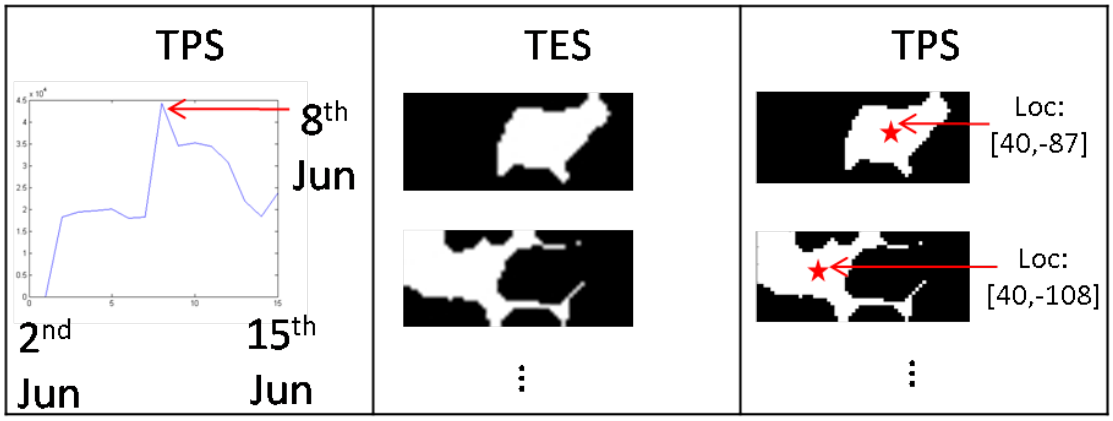

Figure 9.4: Sample results of applying the queries 1-3, for iphone theme

14 days. On the other hand, the *AT&T* retail store locations[1] E-mage was *convolved* with the catchment area (assumed to be a Gaussian Kernel C) for each store. The difference between the obtained 'aggregate interest' and 'net catchment area' E-mages was taken to be a representative of the 'under-served interest areas' where it makes sense to open a new retail store. To find out the best location for such a store, we undertook a *convolution* operation between the catchment area of a new store and this 'under-served interest areas' E-mage. The *maxima* operation on the obtained E-mage gave the most appropriate location. Note that the obtained answer in this case is not merely a pixel location but a physical geo-location with real life semantics. We obtained the details of the physical location via reverse-geocoding using *http* : *//ws.geonames.org*. Thus we were able to solve a sample real world business problem using a combination of primitive situation recognition operators as defined on E-mages.

[1] *AT&T* was the only legal service provider for iPhones in US at the time. We assume that iPhones can only be sold from these retail locations.

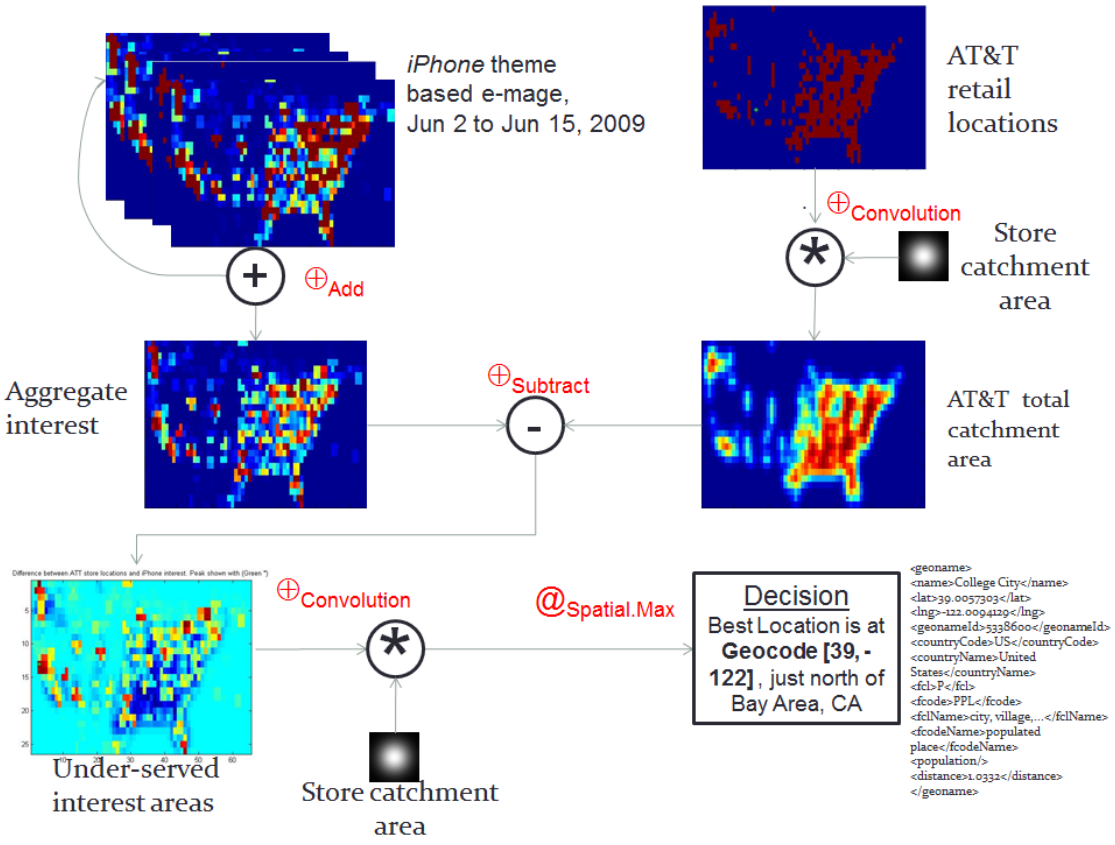

Figure 9.5: Combination of operators for undertaking a business decision

9.2.2 Application: Political event analytics

We consider sample queries relevant to a political analyst, or campaign manager, when dealing with spatio-temporal data about a personality of interest (P) (e.g. 'Obama') or issue of interest (I) (e.g. 'Healthcare').

1. When did the interest peak about personality P across USA?

   $@_{temporal.max}(@_{spatial.sum}(\Pi_{P_R(region=USA)}(TES_P)))$

2. What is the periodicity of the interest in this personality?

   $@_{temporal.periodicity}(@_{spatial.sum}(\Pi_{P_R(region=USA)}(TES_P)))$

3. Show me the interest in issue I when interest in personality P gained its peak?

   $\Pi_{P_t(time=tp)}(\Pi_{P_R(region=USA)}(TES_I))$

   where $tp = @_{temporal.max}(@_{spatial.sum}(\Pi_{P_R(region=USA)}(TES_P)))$

4. What is the similarity between the interest patterns for I and P on the above date?

   $\psi_{spatial.correlation}(\Pi_{P_t(time=tp)}(\Pi_{P_R(region=USA)}(TES_I)), \Pi_{region=USA \wedge time=tp}(TES_P))$

   where $tp = @_{temporal.max}(@_{spatial.sum}(\Pi_{P_R(region=USA)}(TES_P)))$

Figure 9.6: Sample result for running query 3 for political theme I='Healthcare', and politician P='Obama'

The results presented here are based on data collected for Personality 'Obama' and issue 'Healthcare' between Nov 5th 2009, and Mar 30, 2010. The peak of interest in Obama (Query 1) was found on 27th Jan, 2010, which corresponds with his 'State of the Union' address. Obviously, multiple events of interest (and local maximas) about 'Obama' occurred during the 4 months. The periodicity value (query 2) was found to be approximately 20 days. The E-mage for 'Healthcare' on the peak day (Query 3) is shown in Fig. 9.6. Lastly, the similarity value (Query 4) between interest in 'Obama' and 'Healthcare' for that day was found to be 0.6934 (approx. 69.3%).

9.2.3 Application: Seasonal characteristics analysis

In this experiment we consider average green color intensity[2] of all Flickr images uploaded from a location to be the representative value for the STTPoint data corresponding to that location. The experiment is aimed at analyzing climatic seasonal phenomena as they occur across different parts of US. Specifically we want to see which parts of US are more green than others, and what are the seasonal patterns or trajectory of such phenomena. This approach can be extended to understanding flora growth, or bird migration patterns, by using more sophisticated concept detectors for images in future. Here, we show queries for 'snow', based on e-mages created using meta-data (tag) based recognition.

1. Show me the variation in green color intensity as it varies over the year.
   $@_{Sum}(\Pi_{P_R(region=USA)}(TES_{green}))$
2. Where is the peak in the greenery across whole of USA?
   $@_{spatial.max}(\Pi_{P_R(region=USA)}(TES_{green}))$

[2]We normalize the data by using $value = G - average(R + G + B)$

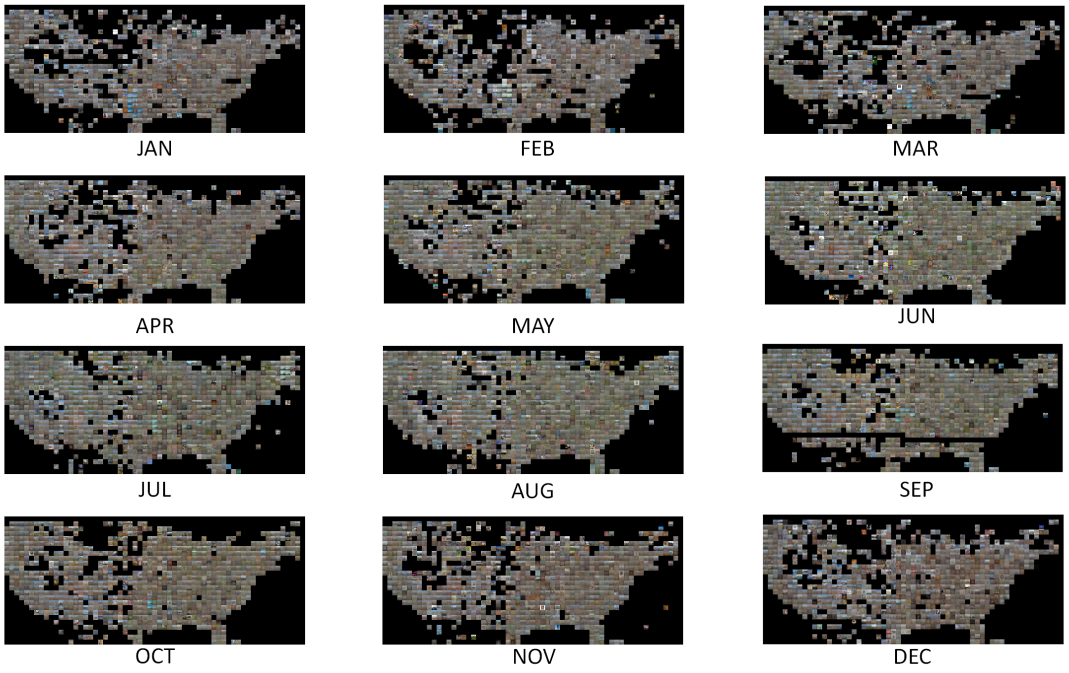

Figure 9.7: Spatio-(temporal) E-mages representing average of Flickr images posted from each location across months in 2010.

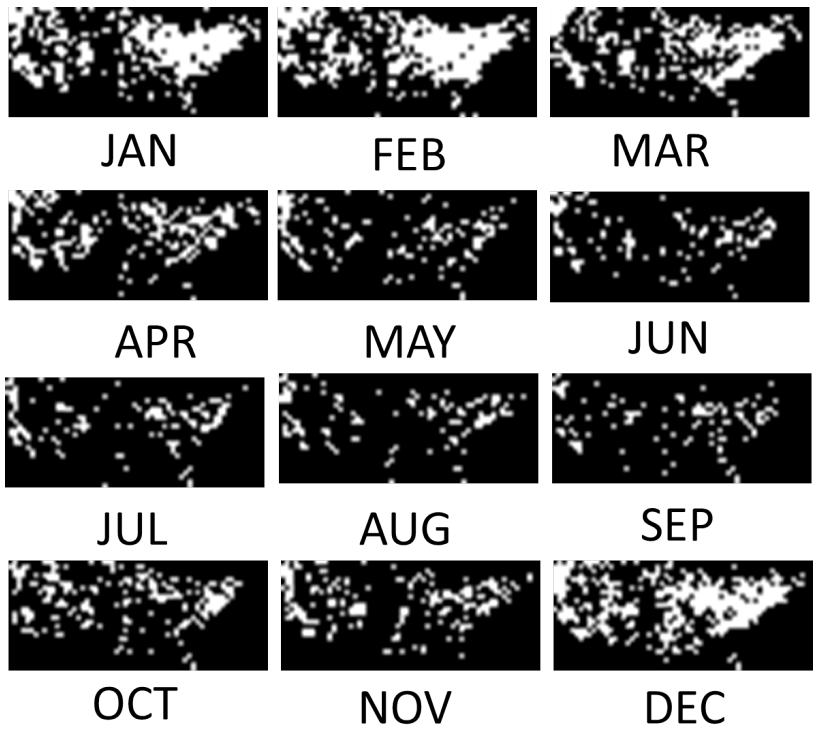

Figure 9.8: Snow themed E-mages through the year based on number of images tagged with 'snow' coming from each location

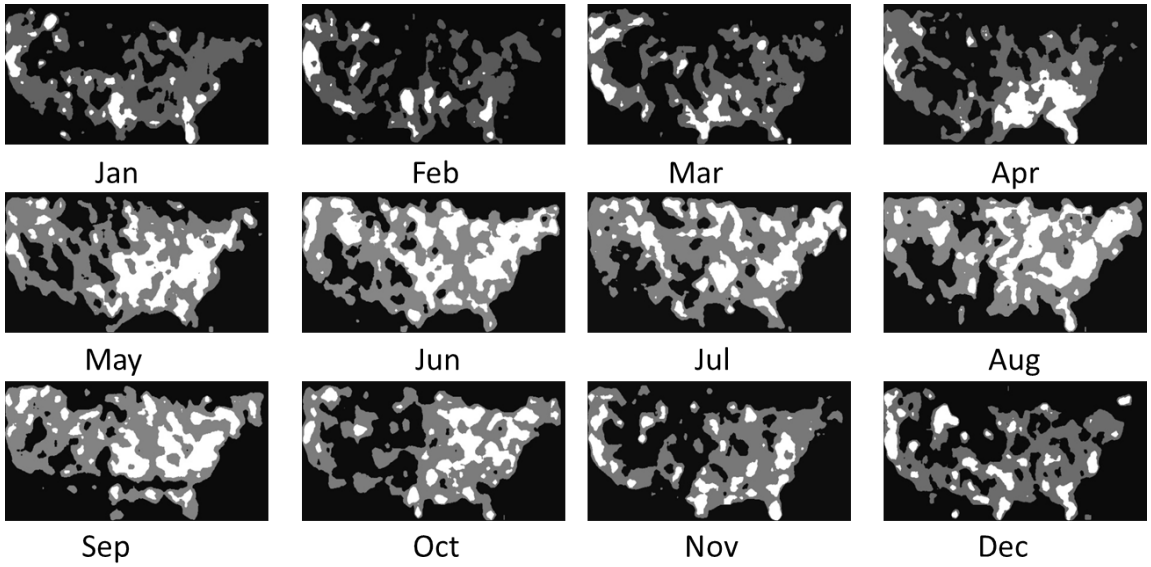

Figure 9.9: Movement of greenery zones (brightest= 'most green') across US through the year

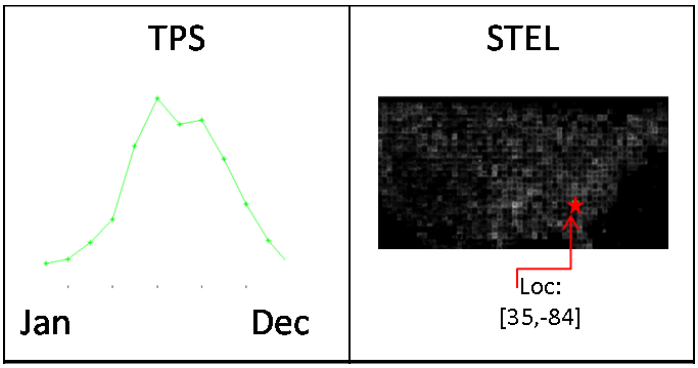

Figure 9.10: Answers for queries 1-2 for seasonal characteristics monitoring using Flickr data

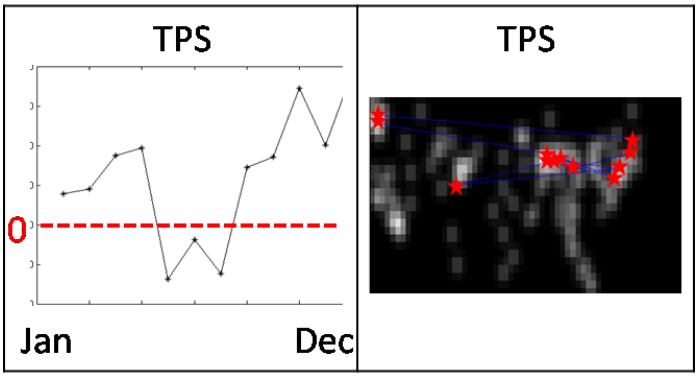

Figure 9.11: Answers for queries 3 and 4 for seasonal characteristics monitoring using Flickr data

3. Show me the segments based on greenery, as they vary over the year.

   $\gamma_{kmeans(k=3)}(\Pi_{P_R(region=USA)}(TES_{green}))$

4. Show me difference between red and green colors for the New England region as it varies over the year.

   $\oplus_{subtract}(\Pi_{P_R(region=[(40,-76),(44,-71)])}(TES_{red}),$

   $\Pi_{P_R(region=[(40,-76),(44,-71)])}(TES_{green}))$

5. Show me the trajectory of the epicenter of high-snow activity region throughout the year.

   $@_{spatial.epicenter}(\Pi_{P_v(val=3)}(\gamma_{kmeans(k=3)}(\Pi_{P_R(region=USA)}(TES_{snow}))))$

6. What is the degree of similarity between snow e-mages and the North to South linear decay pattern?

   $\psi_{spatial.correlation(pattern=NSLinearDecay)}(\Pi_{P_R(region=USA)}(TES_{snow}))$

These results are based on a Temporal E-mage Stream created from Flickr data from US, at a monthly granularity for a whole year. A total of 706,415 images were aggregated for answering queries 1-4, and the E-mages created by averaging images from all over US are shown in Figure 9.7. As shown in Fig. 9.10, the overall green color intensity (query 1) peaked during summer months. The area with most green pictures (query 2) was at [35,-84], which happens to be at the intersection of 3 national forests and 1 national park. The overall variation in zones (see Figure 9.9) of different greenery showed a reasonable trend, moving from the south-eastern US in April towards the north, covering most of the US in summer and then receding southwards again. The relative intensity of red and green colors also showed interesting trends. For example, in the New England region, *green* dominated *red* over the summer months, but *red* overtook *green* during the 'Fall' season.

The queries 5 and 6 were answered using meta-data from 79,628 images. The 'snow' tags data was normalized using the number of geocoded images uploaded from that region on any topic to generate E-mages as shown in Figure 9.8 The trajectory for the epicenter of the high snow is shown in Fig.9.11. Lastly, the similarity value between the aggregate snow activity and 2-D north-south linear decay pattern was found to be 64.6%.

Taken together, the three applications provide evidence that the E-mage data representation and operator can indeed be useful for generating situational insights.

9.3 Building multiple applications using the framework

We now focus our attention on building diverse real-time situation-aware applications by putting together all the three components of the framework viz. situation modeling, situation recognition, and visualization/alerts. The applications discussed in this section have been implemented using EventShop.

9.3.1 Flood evacuation in Thailand

We recently used the framework for suggesting safe locations to people who were trapped in the Thailand flood of Nov-Dec 2011. To do this we first modeled the Flood risk level, personalized it based on the concern shown, and configured rules for sending out the necessary alerts to different users. Multiple people received these tweets 'on-the-ground', and some tweets were re-tweeted. We discuss the models that were created and the implementation that was undertaken using EventShop.

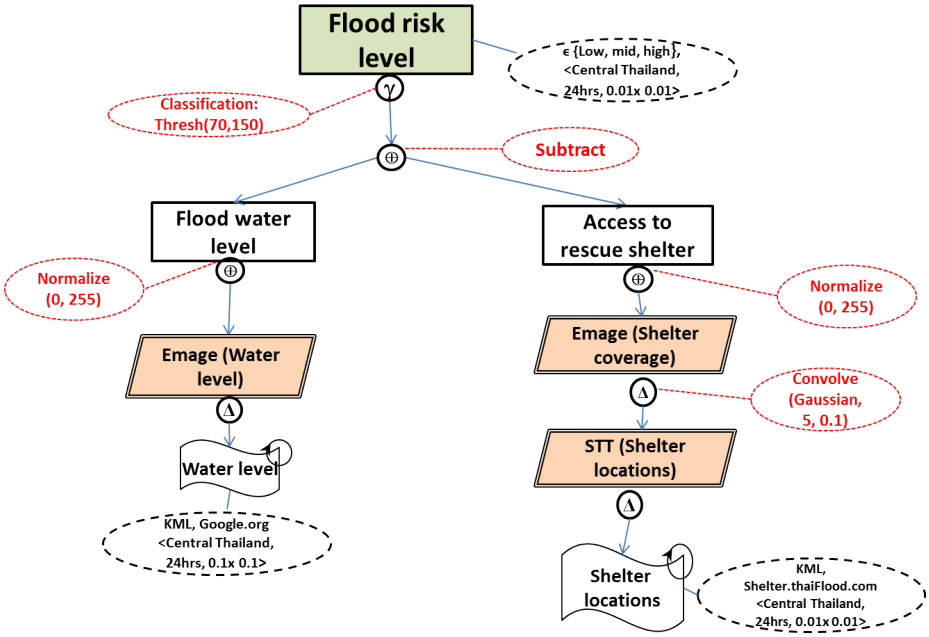

Figure 9.12: Situation Model for flood risk level

Created Situation Models

A graduate student from Thailand whose family was affected by the floods acted as our application domain expert in this study.

1) Macro situation:

A situation model was created using the approach described in Chapter 5. For brevity, we directly present the final created model. A shown in Figure 9.12, the macro-situation was defined based on a combination of the water level observed and the access to a nearby shelter. Each of the open shelters was assumed to have a Gaussian catchment area.

2) Defining personalized situation: Safety concern

The 'safety concern' level for each user was defined based on the combination of the macro-situation (flood safety level) with personal concern level. The personal concern

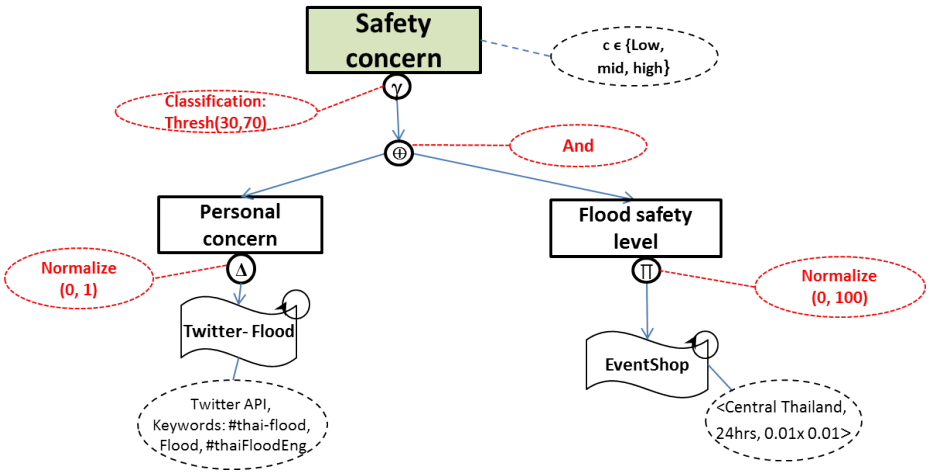

Figure 9.13: Situation Model for personal safety concern

level in this application was defined simply based on the incidence of predefined keywords in the user's Twitter-stream. These keywords indicated a user's concern about the situation, and their interest in seeking helpful information about nearby safe places.

3) Action Rules

We configured the following action rule.

IF u_i *is in (Safety concern = High), THEN connect(*u_i*, nearestLoc(*u_i*, Shelter))*

We implemented this conceptual model using the EventShop system as follows.

Evaluation and deployment using EventShop

1) Data Sources

The data sources employed in this application included the map of flood affected areas across Thailand, as well as shelter map, which were each updated every a few hours. The user concern was derived from Tweets coming from the considered area (central

(a) Flood Affected Areas (b) Shelter Map (c) Tweets on Thai Flood

Figure 9.14: Sample E-mages

Theme	URL	Type	Time Window	Synchronization Point	Spatial Bounding Box	Spatial Resolution
Flooding Area	http://www.thaiflood.com/floodmap	KML Stream	6 hours	0 ms	Southern Thailand	0.01 Lat x 0.01 Long
Shelter Map	http://shelter.thaiflood.com/webservice/request.kml	KML Stream	6 hours	0 ms	Southern Thailand	0.01 Lat x 0.01 Long
Thai Flood Tweet	Twitter Search API	Text Stream	6 hours	0 ms	Southern Thailand	0.01 Lat x 0.01 Long

Figure 9.15: Parameters for the data sources configured in the Flood Recommendation Application

Thailand) with keywords #ThaiFlood, #Flood, and #ThaiFloodEng [3]. Sample E-mages created from the actual data sources are shown in Figure 9.14.

2) Results

A sample of the macro-situation classification results is shown in Figure 9.16. As can be seen, significant parts of the country were under high risk (shown in red in the figure). The personalized action-taking rule was configured using the personalized alert unit in EventShop (also shown in Figure 9.16).

A snapshot of some of the actual tweets sent out by the system is shown in Figure 9.17. As can be seen, some of the tweets were re-tweeted by users, thus indicating a positive interest in receiving and spreading such information.

The Twitter account used was @SocLifeNetworks.

[3]This hashtag was used by Thai Flood tweets in English

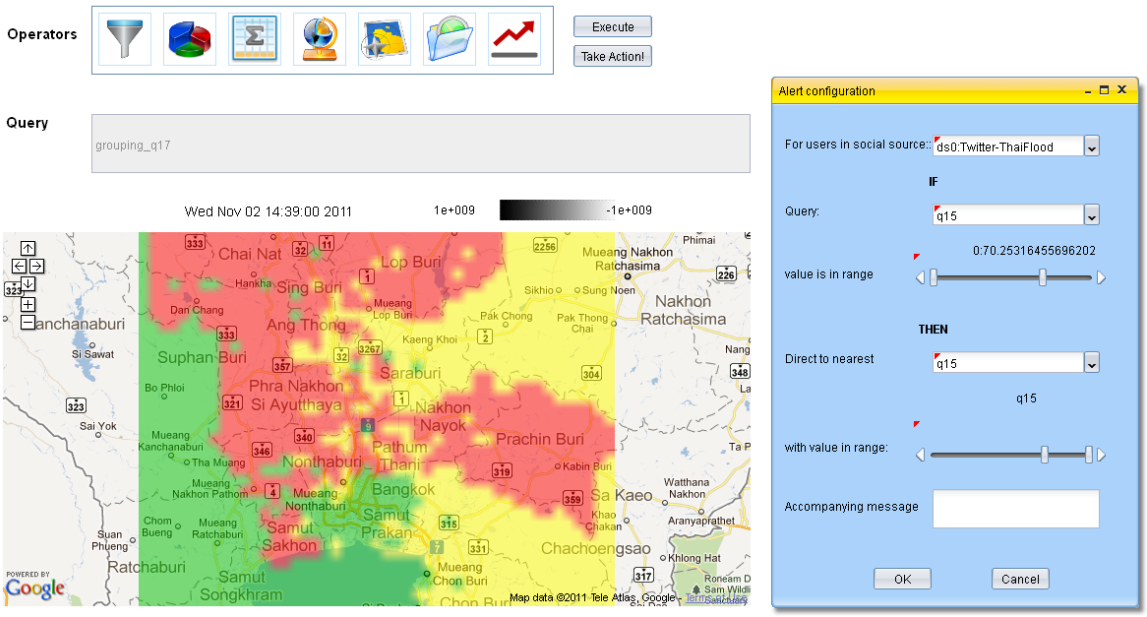

Figure 9.16: Resultant situation classification and Personalized Alert configuration

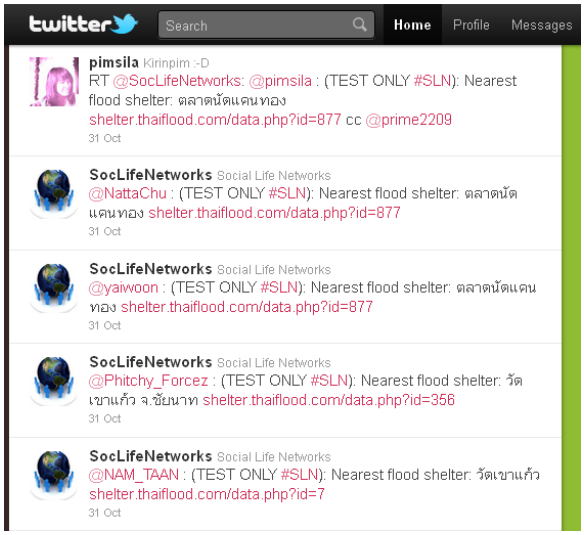

Figure 9.17: Sample tweets sent out to users

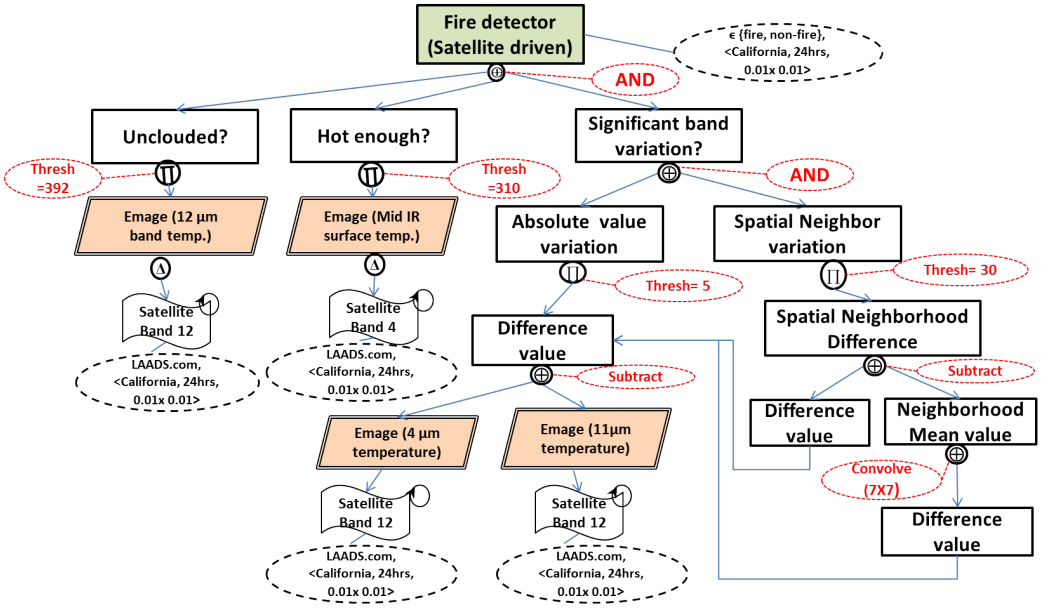

Figure 9.18: 'Wildfire' recognition model using satellite data

9.3.2 Wildfire recognition in California

Wildfires affect large portions of human ecology and often last days and weeks while spreading over large spatial boundaries. It is estimated that tropical fires around the world have destroyed about $15 \times 10^6 km^2$ of forests in the last decade [65]. Quantitative information about the spatial and temporal distribution of fires is important for forest protection and in the management of forest resources. Hence we decided to build a computational model for recognizing wildfires. For this we approached a domain expert (a research scientist in Earth Science department at our university) and requested her to volunteer for our case study.

Created models

Based on the process described in Chapter 5 and her expertise in the area of satellite based fire recognition, the domain expert worked with us to create a situation model as shown in Figure 9.3.2. This model is loosely based on the algorithm described

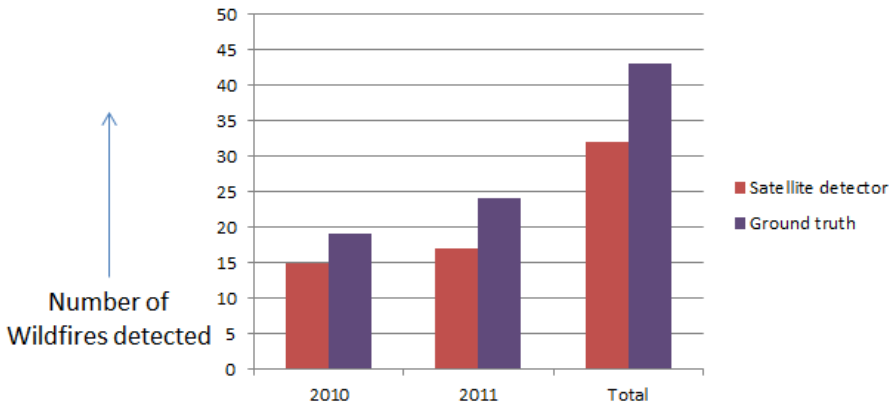

Figure 9.19: Recognition performance of satellite data based detector for Wildfires across California over last 2 years

in [59]. It focuses on using satellite data to recognize large wildfires. Specifically, it focuses on anomalies in inter-band variations between $4\mu m$ and $11\mu m$ wavelength radiations to recognize wildfires. This inter-band variation can only be observed in unclouded regions, which are identified by analyzing the $12\mu m$ band's radiation levels.

Satellite-data based detector: Comparison with ground truth

We configured the proposed wildfire model into EventShop and tried to detect the various wildfire situations across California based on archives of satellite data streams. The archive of satellite data was obtained from NASA's LAADS website *http://ladsweb.nascom.nasa.gov/data/*.

The created model could achieve 74% precision (refer to Figure 9.19) at detecting large fires ($> 1000m^2$) over last 2 years in California. The ground truth used for comparison was obtained from the website of the California Department of Forestry and Fire Protection (http://www.fire.ca.gov/).

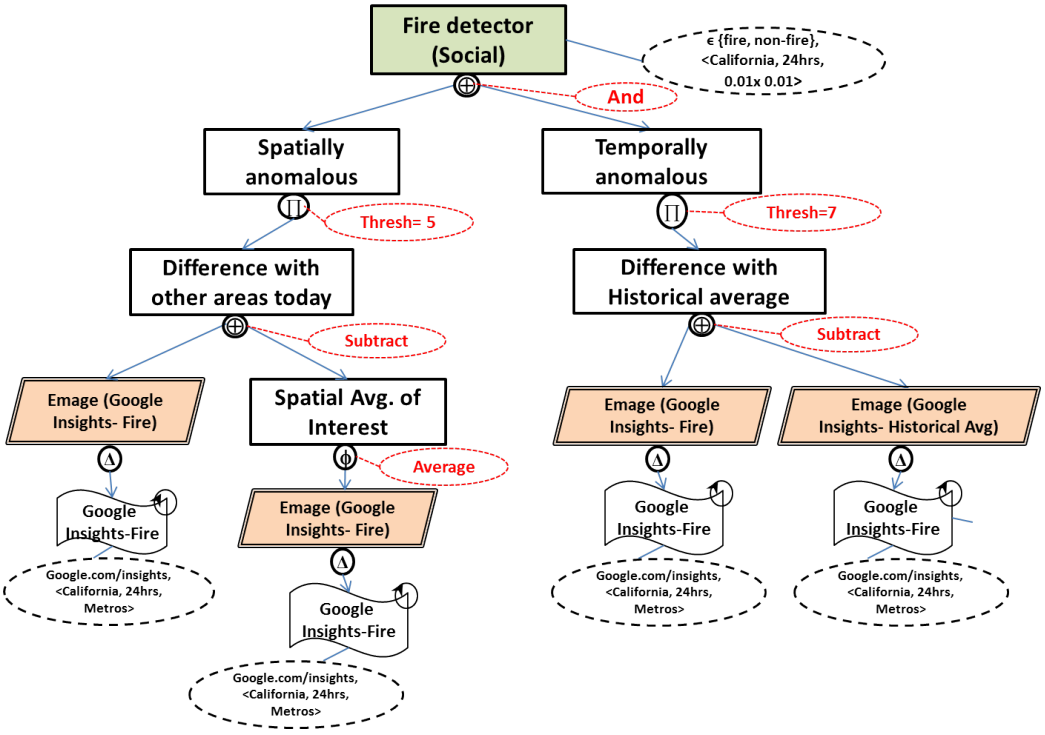

Figure 9.20: 'Wildfire' recognition model using social data

Re-iteration and refinement

We discussed these results with the expert and then built another model (see Figure 9.20) using purely social media (Fire related search queries made on Google from each location) data, available from *http://www.google.com/insights*. Note that the spatial granularity of this data is much coarser (data is available at 'metro area' level), but it complements the satellite based recognition especially in cases where a fire occurred in clouded regions, or was brief but affected large human populations.

We configured this model in EventShop and found that this model could recognize Fire situations in the correct time-frame with 70% accuracy. We consider this by itself to be an interesting finding, as it indicates that spatio-temporal data nuggets (millions of search query logs) can be combined to create the same effective information as was earlier limited to satellites or the proverbial 'God's view'[89].

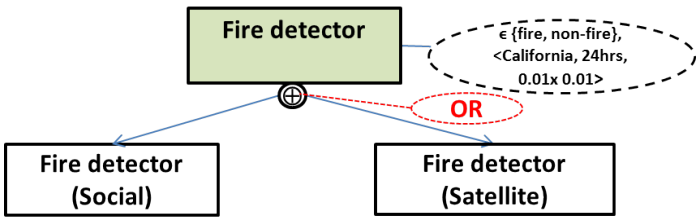

Figure 9.21: 'Wildfire' recognition model using satellite + social data

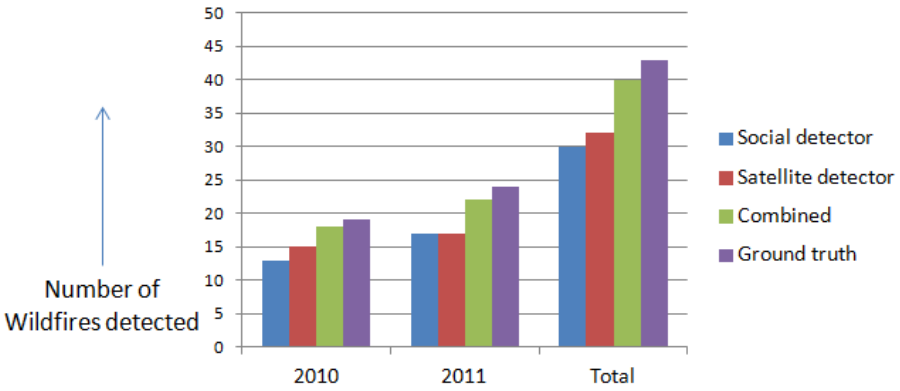

Figure 9.22: Recognition performance of satellite + social detector for Wildfires across California over last 2 years

Final performance: Comparison with ground truth

Lastly, we decided to combine the two recognition approaches and create a unified situation detector which simply combines the two detectors (see Figure 9.21).

The combined detector could recognize more than 90% of the large fires in California. Some of the sample results over different timeframes are shown in Figure 9.23, and a video capture of the filter configuration and results observed is available at *http://auge.ics.uci.edu/eventshop/videos/*.

9.3.3 Hurricane monitoring

In this experiment, we used simulated hurricane data to test EventShop's ability at performing spatio-temporal characterization and pattern matching queries as well

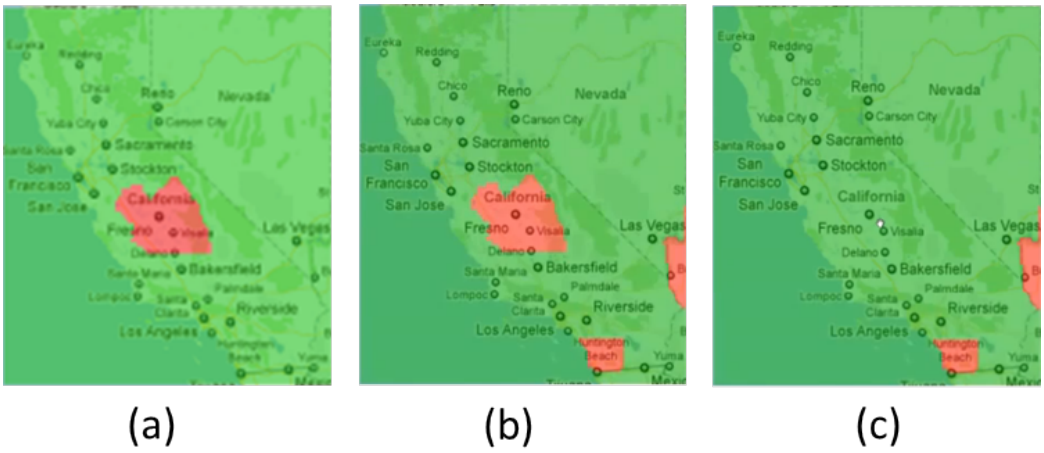

Figure 9.23: Wildfire recognition results over different timeframes

as its ability to continuously update query results in real time. We considered an Analyst level query aimed at analyzing the rate of growth in velocity of a hurricane as observed in geo-imagery.

Created models

The created situation model is shown in Figure 9.24. The basic problem is to find the location of the hurricane (via spatial pattern-matching), obtain its velocity, and compare the velocity pattern to a known temporal pattern.

Evaluation using EventShop

Data Sources

EventShop supports a data source simulator which generates STTPoints as specified by an initial set of *frame parameters*. We selected four locations as centers of four 2D-Gaussian distributions. We also specified amplitude and variance for each of the distributions. In addition, at each time window, we introduced a hurricane pattern at a randomly selected location in the generated E-mage. The parameters of the data

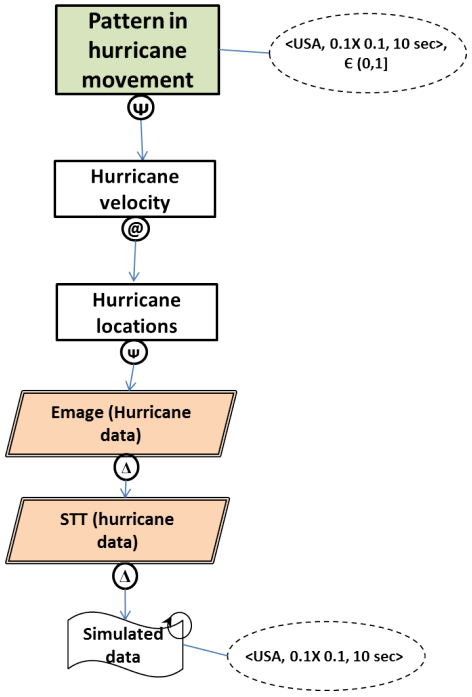

Figure 9.24: Model for movement patterns in hurricanes

Theme	URL	Type	Time Window	Synchronization Point	Spatial Bounding Box	Spatial Resolution
Simulated Hurricane Data	Local Memory Buffer	STT Stream	10 sec	0 ms	US	0.1 Lat x 0.1 Long

Figure 9.25: Parameters for simulated hurricane data source

sources are listed in Figure 9.25. A sample E-mage can be seen as in Figure 9.26.

The final frame parameters as well as the initial frame parameters used were $< 10s, 0, US, 0.1lat \times 0.1long >$

Results

Note that the situation model (shown in Figure 9.24) can be seen as a composite of two steps. The first step is the spatial pattern matching operator, which takes a hurricane pattern as an input and outputs the position and value of the best match. The E-mages in this experiment were generated every 10 seconds; hence, the query result was also updated accordingly every 10 seconds. The second step is the pattern observed in the velocity of the hurricane. The system compared the velocity pattern

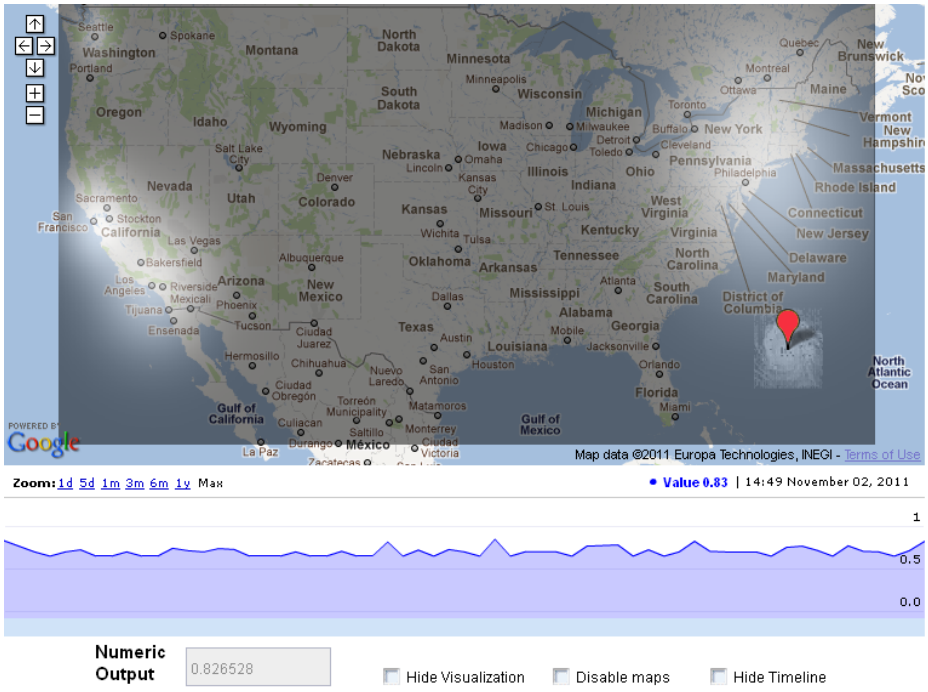

Figure 9.26: Hurricane Application: Spatial pattern matching

with an exponential (base 2, scale 1) temporal pattern.

For easier presentation we first show the output of the first step (i.e. applying the spatial pattern matching operator). The output has a geo coordinate as well as the similarity value, both of which are updated every 10 seconds. They are presented on the map, timeline and the output text box in EventShop. The location where the highest match was found in the active e-mage has been marked on the map. The similarity value (0.82) is shown in the numeric output text box, and the the variation in this similarity value over time can be seen in the series shown in the timeline.

Result of the second (final) part is a similarity value (between the hurricane speed observed over time, and a generated exponential pattern) which gets updated every 10 seconds. Figure 9.27 shows the results on a timeline. We do not try to draw any insights from the observed values, as they are based on simulated data, and simply verify that such applications can be supported by EventShop.

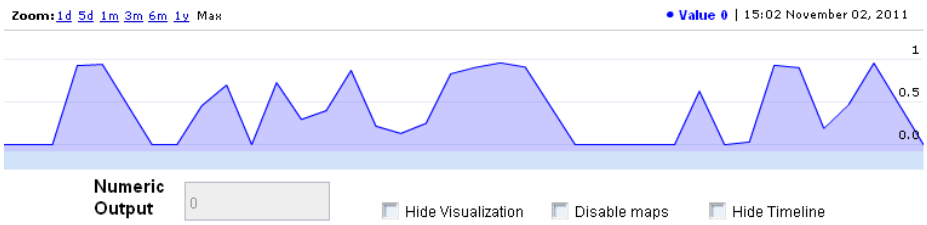

Figure 9.27: Hurricane Application: Temporal pattern matching

9.3.4 Flu epidemic monitoring and recommendation

Here we resume and extend our discussion on modeling and detecting epidemic outbreaks started in Chapter 5.

Created models

1) Defining macro situation: Flu Epidemic Outbreak Risk

We use the same model as discussed earlier in Chapter 5. As shown in Figure 9.28, it is characterized by growing unusual activity in Influneze-Like-Illness (ILI) incidents.

Evaluation using EventShop

We translated the created model into EventShop. A video capture of the process is available at http://auge.ics.uci.edu/eventshop/videos/. For the current purpose we used the average of tweets on ILI for the last month to be the 'historical' average level. We let the system run for two weeks (Apr/30/12-May/13/12) with real-time Twitter data feeds passing through the created filter, and (thankfully!) saw no severe Epidemic outbreak risks. Sample E-mages for the different data streams are shown in Figure 9.29, and a sample of the configured detector's result is shown in Figure 9.30.

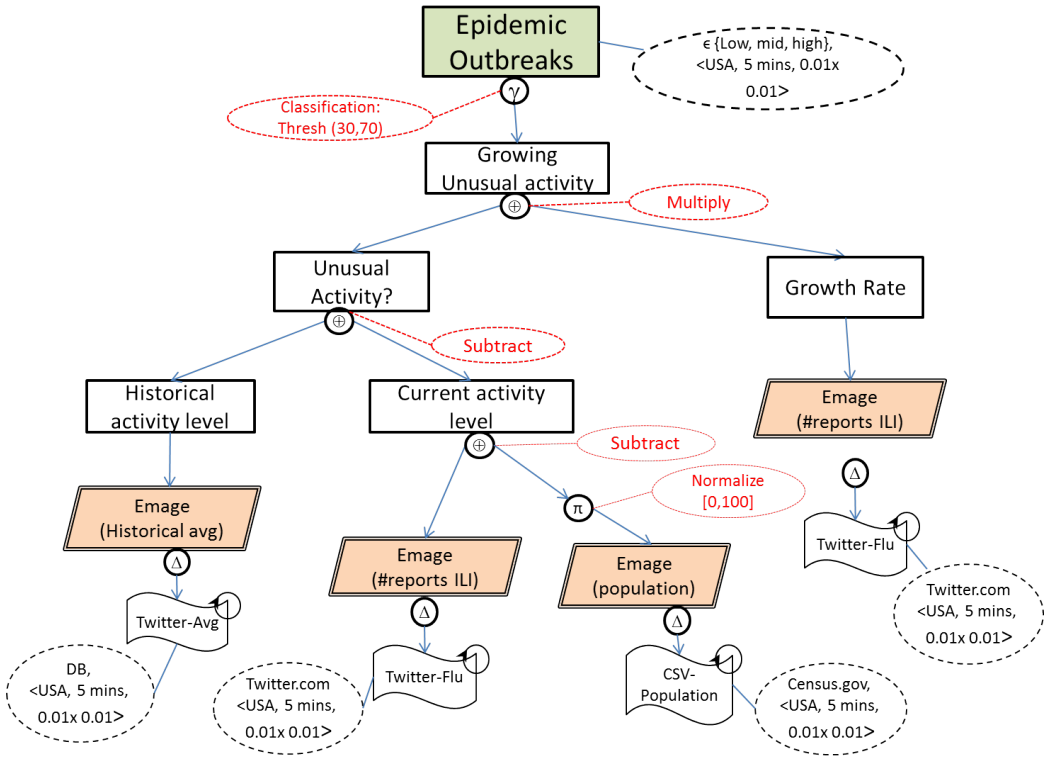

Figure 9.28: Situation Model for Epidemic Outbreaks

9.3.5 Asthma/Allergy recommendation system

This application builds upon our discussion in Chapter 8. The application detects the Allergy /risk level for different location in the US, and then it advises the highly-vulnerable people to stay indoors while prompting those in healthy environments to enjoy the outdoors (e.g. to go jogging at the nearest park).

Created model

Let's look at the 3 steps of personalized situation modeling:

1) Defining macro situation:

The macro-situation can be modeled as discussed in Chapter 8. The severity of risk in an environment to an asthma patient is related to the pollen count and air quality

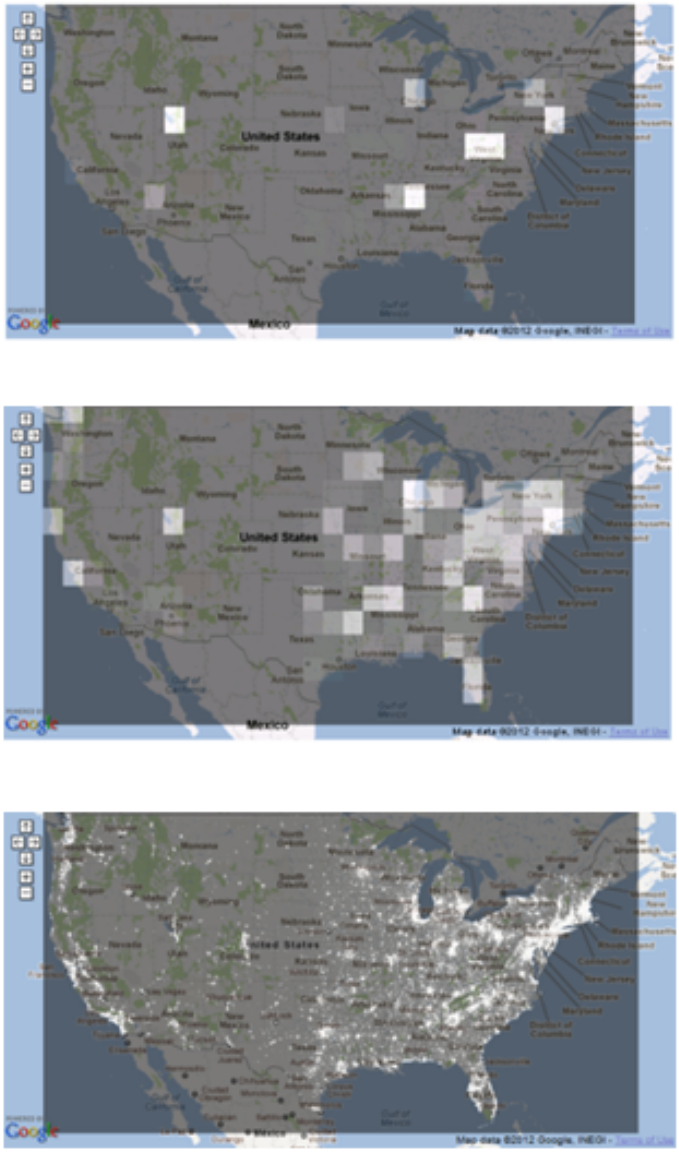

Figure 9.29: E-mage for (a) Reports on Flu (brighter indicates more reports), (b) Historical average, (c) Population

Figure 9.30: 'Epidemic outbreak risk level

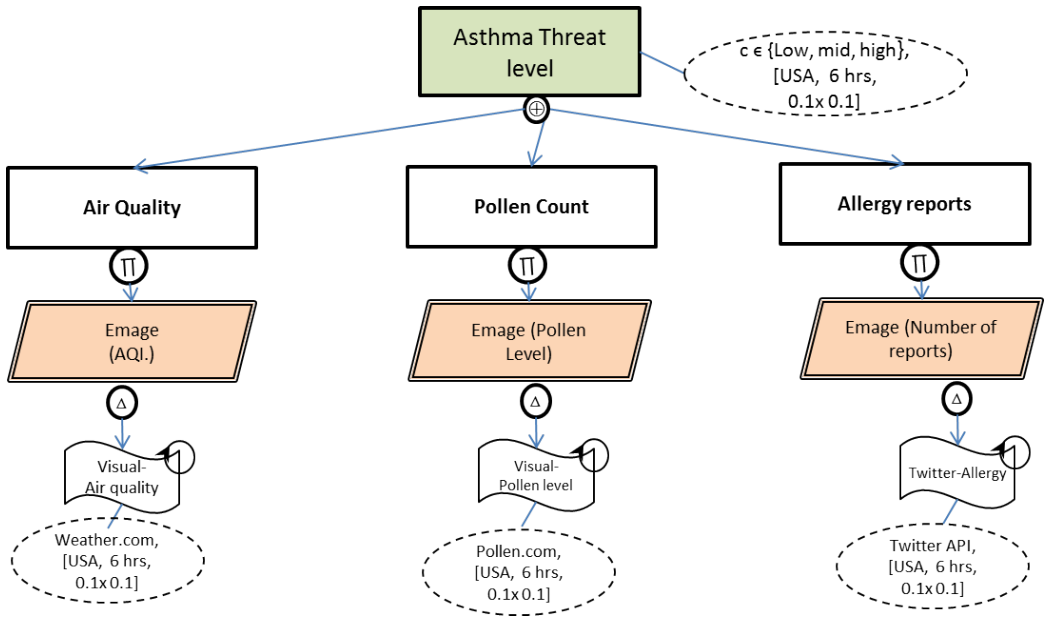

Figure 9.31: Situation Model for asthma threat level

in that area. Similarly a large number of reports on social media (human sensors) with allergy related terms indicate a high allergy incidence rate.

2) Defining personalized situation: Personal threat level

The personal threat level can be defined based on combining the environmental threat level with the personal exertion level. The exertion level can be derived from the sensor stream from a user's mobile device.

3) Situation-action rule:

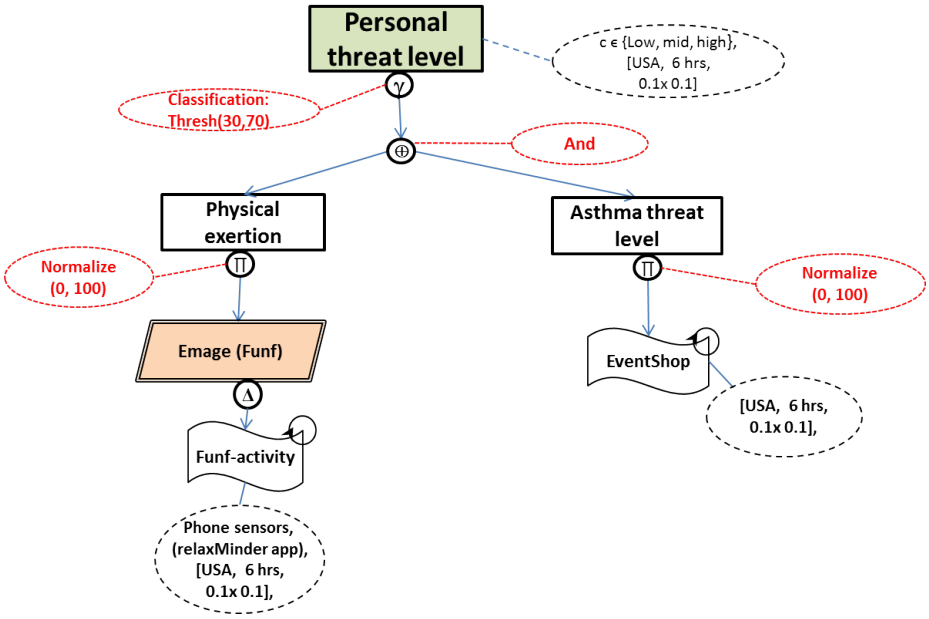

Figure 9.32: Situation Model for personal threat level

We configure two situation action rules:

IF u_i *is in (Personal Asthma Threat = low) THEN connect*(u_i, *nearestLoc*(u_i, *Park*)), *'Great day to go out jogging'.*

and

IF u_i *is in (Personal Asthma Threat = High) THEN connect*(u_i, *n/a), 'Avoid exertion! stay indoors'*

Results

We implemented these models in EventShop and configured the various parameters as shown in Figure 9.33.

The final frame parameters specified for this query are:

$< 1day, 0, US, 0.1lat \times 0.1long >$

E-mages coming from the three data sources with distinct initial frame parameters

Theme	URL	Type	Time Window	Synchronization Point	Spatial Bounding Box	Spatial Resolution
Pollen Count	http://pollen.com/images/usa_map.gif	Geo Image Stream	1 day	0 ms	US	0.1 Lat x 0.1 Long
Air Quality	http://www.epa.gov/airnow/today/forecast_aqi_20111101_usa.jpg	Geo Image Stream	1 day	0 ms	US	0.1 Lat x 0.1 Long
Asthma Tweet	Twitter Search API	Text Stream	6 hours	0 ms	US	0.1 Lat x 0.1 Long

Figure 9.33: Parameters for the data sources configured in the Allergy Risk Recommendation Application

Figure 9.34: Asthma Application: Sample E-mages

are mapped to E-mages following the same final frame parameters by the Resolution Mapper. Sample E-mages from the three data sources are shown in Figure 9.34, and a sample macro-situation result is shown in Figure 9.35.

The personalization was undertaken using the physical level readings from users' mobile phones. Specifically, we have developed and distributed an Android application (called RelaxMinder) using the 'funf' sensing framework [4] that reports each users activity level. It can be configured as a data source in EventShop with the parameters

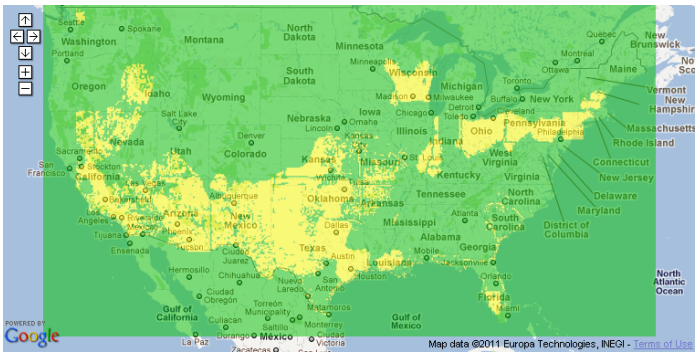

Figure 9.35: Sample result

Theme	URL	Type	Time Window	Synchronization Point	Spatial Bounding Box	Spatial Resolution
Physical Exertion	RelaxMinder App	Mobile Sensors	6 hours	0 ms	US	0.1 Lat x 0.1 Long

Figure 9.36: Parameters for the Personal activity level data source

Figure 9.37: Sample Tweets sent out from EventShop

as shown in Figure 9.36.

A combination of personal and macro values was used to send out personalized alerts. Some of the sample tweets sent out to users are shown in Figure 9.37. Note that different users are given different recommendations (e.g. 'Stop exerting', or 'Great day to be outdoors') based on their personalized situations.

We are currently working with a practicing doctor (an allergist) to refine the application and make it suitable for a field trial in the near future.

9.3.6 Discussion and looking back at the design goals

A summary of the different situation-based applications experimented with in this dissertation is presented in Figure 9.38. We highlight three observations on the ap-

plications and the use of the framework for creating them.

1. **Application diversity:**

   The applications show a wide variety in their target domain e.g. Health, Natural disasters, Seasonal patterns, Business Intelligence, and Political events. This indicates the broad applicability of the situation recognition problem and the framework's ability to handle them.

2. **Data diversity:**

   The data employed in these applications has varied from being offline to online, real to simulated, and covered a wide variety in terms of data sources (Twitter, Flickr, Google insights, Satellite data, Census, Geo-image streams, and Mobile phone sensors).

3. **Real-world validation:**

   Seven of the applications employed real world data. Situational insights in the first three applications were found to be relevant to real world events (e.g. launch of a new iPhone, Obama's health care announcement, and Fall season incidence). The Thailand flood application was deployed in practice, and sent out tweets to thousands of real users, some of whom re-tweeted the messages. Similarly, the wildfire recognition results were compared offline with 'ground-truth' and showed $> 90\%$ correlation.

We now also look back at the design goals set forth in Chapter 2 and evaluate the holistic design framework based on the experience at building the five applications.

1. **Expressive power**

   We found the abstractions designed to be expressive enough to cover a wide variety of situations in very varied applications. The level of sophistication varied

S.No	Application	Data Used	Application deployed?	Scale	Data modalities	Operators used	Implementation platform
1	Business intelligence	Real	No	Macro	Twitter, Census	F, A, C, Ch	Matlab
2	Politics	Real	No	Macro	Twitter, Election results	F, A, Ch, P	Matlab
3	Seasonal characteristics	Real	No	Macro	Flickr	F, A, C, Ch, P	Matlab
4	Thailand flood mitigation	Real	Yes	Macro, Personalized alerts	KML	F, A, C	EventShop
5	Wildfire detection in California	Real	Yes	Macro	Satellite data, Google insights	F, A, Ch	EventShop
6	Hurricane monitoring	Simulated	No	Macro	n/a	F, A, Ch, P	EventShop
7	Flu epidemic surveillance	Real	No	Macro	Twitter, Census	F, A, C	EventShop
8	Allergy/ Asthma recommendation	Real	In-progress	Macro, Personalized alerts	Twitter, Air Quality, Pollen Count	F, A, C	EventShop

Legend:
F = Filter,
A = Aggregate,
C = Classification,
Ch = Characterization,
P = Pattern Matching

Figure 9.38: Summary of different applications discussed in this dissertation

from basic aggregation of streams to sophisticated spatio-temporal matching. The applications also varied in terms of their domain, region considered (USA, California, and Thailand), Targeted audience (Analysts VS Lay-persons), and data streams employed.

2. **Lower the floor**

   (a) **Reduced time to build**

   Once the framework was in place, building additional applications took very little time (typically a couple of hours) - the majority of which were spent designing the requirements and modeling the situation, rather than building software. We consider this to be a very positive result. The actual time taken to configure each of the applications discussed into EventShop was less than 10 minutes (please refer to video captures at http://auge.ics.uci.edu/eventshop/videos/).

   (b) **Lower CS expertise required**

The only level of expertise required was an ability to deal with a graphical tool (EventShop) and configure its parameters. This is a huge improvement over the base case scenario where the interested domain experts would need to implement similar applications from scratch (e.g. write application code, configure databases, streams, websites, and implement various analysis operations)

3. **Raise the ceiling**

   We argue that the framework not only supported situation recognition but also raised the quality of the detectors defined.

   (a) **Better designed situation detectors.**

   The design support for situation modeling helped domain experts avoid the 'cold-start' mental block during the design process. It also lead to explicitly defined situation models, which made it much easier for the domain experts and the IT personnel to interact and agree on the building blocks and their descriptions. An ability to rapidly test, and refine the models also led to better quality detectors. For example in the wildfire recognition application, the recognition performance could be improved by supporting the use of diverse data streams and refining the model with very little additional cost.

   (b) **Provide personalization options**

   Traditional situation recognition and action-taking has focused on single large scale (e.g. over city, state, country) decision-making. The framework provided tools for personalized situation recognition and control. As seen in Thailand flood mitigation, Flu recommendation, and most prominently in the Asthma recommendation application, the framework could be configured to send out personalized alerts based on a combination of

macro-situation and personal parameters. To the best of our knowledge, this framework is the first systematic attempt at supporting this kind of affordance.

Chapter 10

Conclusions and Future Work

The growth in social media, multimodal mobile sensing, and location driven sensing have paved the way for faster, more reliable, and more transparent concept recognition than possible ever before. Situation recognition is the problem of deriving actionable insights from heterogeneous, real-time, big multimedia data to benefit human lives and resources in different applications. This dissertation has presented a framework for personalized situation recognition and control. Specifically, it defined an approach for conceptual situation modeling, real-time situation evaluation, and generating personalized alerts.

This dissertation has motivated and computationally grounded the problem of situation recognition (Chapter 2). It surveyed a variety of situation aware applications and defined the abstractions required for a generic framework to create situation based applications. The framework defines a common data-structure (STT) to unify heterogeneous data streams, to aggregate them (E-mage Streams), and then to analyze them (Situation Recognition Algebra). Under the hood, each of the analysis operations was transposed into image or video analysis operations, thus making sophisticated analysis

of spatio-temporal content possible via off-the-shelf media processing operators.

The design support for modeling different situations, personalizing them, implementing them, and rapidly refining them was provided via the Situation Models, Situation-Action Rules, and the EventShop system. The Situation Models (Chapter 5) allow the designers to logically partition their potentially vague and complex situation descriptors into explicit, computable components derivable from different sources. The personalization operators and Action rules templates (Chapter 8) support personalized action-taking at scale. The developed web-based system, EventShop (Chapter 7) operationalizes the framework and allows for easy validation and refinement of situation models for their deployment into diverse applications. The experiences with implementing multiple situation-aware applications (Chapter 9) demonstrated the expressive power of the framework as well as an ability to meet the design goals of 'raising the ceiling' and 'lowering the floor' [75], thus resulting in conceptually designed, faster, and easier-to-build situation-aware applications.

To the best of our knowledge, this framework is the first systematic attempt at combining heterogeneous real-time multimedia data into actionable situations. This is also a first attempt at creating a holistic toolkit which supports the creation of multiple situation-based applications spanning web-scale diversity in terms of functional domains, the designer's experience level, and the heterogeneity of the data employed.

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

- **Scalability:**

  Scalability is a major concern when building web-scale systems which incorporate realtime data from all possible sources. Multiple efforts in the distributed systems community (e.g. 'Hadoop', 'Storm' [15, 18]) are being developed and showing early signs of competence at handling large-scale data processing. Sim-

ilar growth in other areas of computing, and their adoption into this framework would be pivotal to making it available at a web scale.

- **User experience:**

  User experience remains a key issue for designing social eco-systems for Situation Recognition. Systems need to support users with varying cultural backgrounds, access to computing infrastructure, and expertise levels. We need to design ways in which thousands of users can easily collaborate to create different applications for personal and shared benefit.

- **Data discovery:**

  With the abundance of data, the real issue becomes how to find the right one. Techniques for identifying the right sources for the problem, and/or repurposing data for different problems, become important.

- **Application discovery:**

  Just like abundance of data, there will be an abundance of available applications and situation logics. Tools to easily find, experiment with, re-purpose, and combine such applications will become increasingly important in the future.

- **Personalization:**

  This work has taken the first steps towards large scale personalization of situations. Richer support for personalization, and its integration into EventShop, would be very useful in numerous applications.

- **Richer operation set:**

  With increasing adoption, there will be a demand for a richer array of tools and operations. The operators in EventShop are extensible and will be made open-source. More sophisticated operators developed by different parties will need to be integrated into EventShop to support more advanced applications.

This may also involve support for multiple data representations (e.g. grids + graphs) and the interplay between them.

- **Building an eco-system:**

  These tools can create societal impact only if they reach a large array of users for solving diverse problems. The integration of such tools into active community-based platforms like Ushahidi would leapfrog this process and build eco-systems which support combination of real-time data for multiple applications. Designing mechanisms which motivate users to become a part of such eco-systems will become increasingly important.

To summarize, this dissertation marks an early effort at supporting the analysis of real-time Spatio-temporal streams for understanding and responding to the evolving world. Once made accessible to the masses such tools could have transformative societal impact. Just like Wikipedia allows for integration of all the (static) knowledge in the world, this framework has a vision of integrating all the dynamic information of the world and operationalizing it into actionable situations. Together, different users can create various applications for personal and societal good and move towards realizing the Utopian vision of prescient understanding and response. With advancements in computational models and processing techniques, current systems can evolve into an effective nervous system for our society [78], one that will provide better civic services, a greener planet, and a safer, healthier population.

Bibliography

[1] D. 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*, 12(2):120–139, 2003.

[2] E. Adam. Fighter cockpits of the future. In *Digital Avionics Systems Conference, 1993. 12th DASC., AIAA/IEEE*, pages 318–323. IEEE, 1993.

[3] A. Adi and O. Etzion. Amit-the situation manager. *The VLDB JournalThe International Journal on Very Large Data Bases*, 13(2):177–203, 2004.

[4] 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*, 2011.

[5] I. F. Akyildiz, M. C. Vuran, and zgr B. Akan. On exploiting spatial and temporal correlation in wireless sensor networks. In *In Proceedings of WiOpt 2004: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 71–80, 2004.

[6] J. Anderson, R. Michalski, R. Michalski, J. Carbonell, and T. Mitchell. *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986.

[7] L. Anselin, I. Syabri, and Y. Kho. Geoda: An introduction to spatial data analysis. *Geographical Analysis*, 38(1):5–22, 2006.

[8] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear road: A stream data management benchmark. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 480–491. VLDB Endowment, 2004.

[9] N. Ashish, R. Eguchi, R. Hegde, C. Huyck, D. Kalashnikov, S. Mehrotra, P. Smyth, and N. Venkatasubramanian. Situational awareness technologies for disaster response. *Terrorism Informatics*, pages 517–544, 2008.

[10] Y. Bai, F. Wang, P. Liu, C. Zaniolo, and S. Liu. Rfid data processing with a data stream query language. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1184–1193. Ieee, 2007.

[11] N. Bansal and N. Koudas. Blogscope: a system for online analysis of high volume text streams. In *PVLDB*, pages 1410–1413, 2007.

[12] J. Barwise and J. Perry. The situation underground. *Stanford Working Papers in Semantics*, 1:1–55, 1980.

[13] J. Barwise and J. Perry. Situations and attitudes. *The Journal of Philosophy*, 78(11):668–691, 1981.

[14] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2011.

[15] D. Borthakur. The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 11:21, 2007.

[16] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.

[17] O. Brdiczka, P. Yuen, S. Zaidenberg, P. Reignier, and J. Crowley. Automatic acquisition of context models and its application to video surveillance. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 1175–1178. IEEE, 2006.

[18] J. Brockmeier. A twitter storm arrives: Storm project open sourced. *http://www.readwriteweb.com/cloud/2011/09/a-twitter-storm-arrives-storm.php*, 2011.

[19] N. Carrier, T. Deutsch, C. Gruber, M. Heid, and L. Jarrett. The business case for enterprise mashups. *IBM White Paper*, 2008.

[20] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S. Kim. Composite events for active databases: Semantics, contexts and detection. In *Proceedings of the international conference on very large data bases*, pages 606–606. INSTITUTE OF ELECTRICAL & ELECTRONICS ENGINEERS (IEEE), 1994.

[21] S. Chakravarthy and D. Mishra. Snoop: An expressive event specification language for active databases. *Data & Knowledge Engineering*, 14(1):1–26, 1994.

[22] M. Chatti, M. Jarke, M. Specht, and U. Schroeder. Model-driven mashup personal learning environments. *International Journal of Technology Enhanced Learning*, 3(1):21–39, 2011.

[23] J. Crowley. Context driven observation of human activity. *Ambient Intelligence*, pages 101–118, 2003.

[24] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 2011.

[25] F. Daniel, F. Casati, S. Soi, J. Fox, D. Zancarli, and M. Shan. Hosted universal integration on the web: The mashart platform. *Service-Oriented Computing*, pages 647–648, 2009.

[26] G. De Giacomo, Y. Lespérance, and H. Levesque. ¡ i¿ congolog¡/i¿, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1):109–169, 2000.

[27] A. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, 2000.

[28] N. Diakopoulos and D. Shamma. Characterizing debate performance via aggregated twitter sentiment. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1195–1198. ACM, 2010.

[29] D. Dietrich, W. Kastner, T. Maly, C. Roesener, G. Russ, and H. Schweinzer. Situation modeling. In *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, pages 93–102. IEEE, 2004.

[30] P. Dimandis. Abundance is our future. In *TED Conference*, 2012.

[31] C. Dominguez, M. Vidulich, E. Vogel, and G. McMillan. Situation awareness: Papers and annotated bibliography. armstrong laboratory, human system center, ref. Technical report, AL/CF-TR-1994-0085, 1994.

[32] B. Dostal. Enhancing situational understanding through employment of unmanned aerial vehicle. *Army Transformation Taking Shape: Interim Brigade Combat Team Newsletter*, 2007.

[33] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: Representation and algorithms. In *International Joint Conference on Artificial Intelligence*, volume 13, pages 166–166. LAWRENCE ERLBAUM ASSOCIATES LTD, 1993.

[34] R. Duda and P. Hart. *Pattern classification and scene analysis*. Wiley, 1996.

[35] N. Eagle, A. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, 2009.

[36] P. Elias, L. Roberts, et al. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[37] M. Endsley. Situation awareness global assessment technique (sagat). In *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*, volume 3, pages 789–795, May 1988.

[38] M. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64, 1995.

[39] J. Fagan. Mashing up multiple web feeds using yahoo! pipes. *Computers in Libraries*, 27(10):8, 2007.

[40] M. Gao. *EventShop: A scalable framework for analysis of spatio-temporal-thematic data streams*. PhD thesis, University of California, Irvine, 2012.

[41] M. Gao, V. Singh, and R. Jain. Eventshop:from heterogeneous web streams to personalized situation detection and control. In *Proceedings of the ACM international conference on Web Science*. ACM, 2012.

[42] N. Gehani, H. Jagadish, and O. Shmueli. Composite event specification in active databases: Model & implementation. In *Proceedings of the International Conference on Very Large Data Bases*, pages 327–327. Citeseer, 1992.

[43] L. Golab and M. Özsu. Data stream management issues–a survey. Technical report, Technical Report, Apr. 2003. db. uwaterloo. ca/˜ ddbms/publications/stream/streamsurvey. pdf, 2003.

[44] M. Gonzalez, C. Hidalgo, and A. Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[45] S. Greenhill, S. Venkatesh, A. Pearce, and T. Ly. Situation description language implementation. Technical report, DTIC Document, 2002.

[46] E. Griffin. *Foundations of Popfly: Rapid Mashup Development*. Springer, 2008.

[47] Ş. Gündüz and M. Özsu. A web page prediction model based on click-stream tree representation of user behavior. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–540. ACM, 2003.

[48] B. Gutenberg and C. Richter. Seismicity of the earth and associated phenomena. *Princeton, New Jersey*, 1954.

[49] Q. Hart and M. Gertz. Optimization of multiple continuous queries over streaming satellite data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 243–250. ACM, 2006.

[50] S. Haynes and R. Jain. Event detection and correspondence. *Optical Engineering*, 25:387–393, 1986.

[51] G. Higgins. System for distributing, processing and displaying financial information, Dec. 14 1993. US Patent 5,270,922.

[52] G. Hjaltason and H. Samet. Incremental distance join algorithms for spatial databases. In *ACM SIGMOD Record*, volume 27, pages 237–248. ACM, 1998.

[53] A. Jadhav, H. Purohit, P. Kapanipathi, P. Ananthram, A. Ranabahu, V. Nguyen, P. Mendes, A. Smith, M. Cooney, and A. Sheth. Twitris 2.0: Semantically empowered system for understanding perceptions from social data. *Proceedings of the Semantic Web Challenge 2010*, 2010.

[54] R. Jain, V. Singh, and G. M. Social life networks for the middle of the pyramid. In *International Workshop on Social Media Engagement*, 2011.

[55] R. Jain and D. Sonnen. Social life networks. *IT Professional*, 13(5):8–11, 2011.

[56] G. Jakobson, J. Buford, and L. Lewis. A framework of cognitive situation modeling and recognition. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1–7, Oct. 2006.

[57] E. Jeannot, C. Kelly, and D. Thompson. The development of situation awareness measures in atm systems. 2003.

[58] K. Johnston, J. Ver Hoef, K. Krivoruchko, and N. Lucas. *Using ArcGIS geostatistical analyst*, volume 300. Esri Redlands, CA, 2001.

[59] Y. Kaufman, C. Justice, L. Flynn, J. Kendall, E. Prins, L. Giglio, D. Ward, W. Menzel, and A. Setzer. Potential global fire monitoring from eos-modis. *Journal of Geophysical Research*, 103(D24):32215–32, 1998.

[60] S. Kingsbury. Wisdom for the Masses. *American Speech*, pages 358–360, 1987.

[61] K. Kloeckl, O. Senn, G. Di-Lolrenzo, and R. C. Live singapore!-an urban platform for real-time data to program the city. In *Computers in Urban Planning and Urban Management*, 2011.

[62] R. Kowalski and M. Sergot. A logic-based calculus of events. *New generation computing*, 4(1):67–95, 1986.

[63] H. Levesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1998.

[64] H. J. Levesque, R. Reiter, Y. Lesprance, F. Lin, R. B. Scherl, and R. B. Golog: A logic programming language for dynamic domains, 1994.

[65] Z. Li, S. Nadon, and J. Cihlar. Satellite-based detection of canadian boreal forest fires: Development and application of the algorithm. *International Journal of Remote Sensing*, 21(16):3057–3069, 2000.

[66] A. Loechel, G. Mihelcic, and S. Pickl. An open source approach for a military situational awareness system. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 1462–1471. IEEE, 2012.

[67] D. Luckham. *The power of events: an introduction to complex event processing in distributed enterprise systems*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[68] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

[69] A. Marcus, M. Bernstein, O. Badar, D. Karger, S. Madden, and R. Miller. Tweets as data: demonstration of tweeql and twitinfo. In *Proceedings of the 2011 international conference on Management of data*, pages 1259–1262. ACM, 2011.

[70] J. McCarthy, P. Hayes, and S. U. C. D. O. C. SCIENCE. *Some philosophical problems from the standpoint of artificial intelligence*. Stanford University, 1968.

[71] I. Merriam-Webster. *Merriam-Webster's collegiate dictionary*. Merriam-Webster, 2003.

[72] 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.

[73] N. Moray and T. Sheridan. Oil sont les neiges d'antan? *Dennis A. Vincenzi, Ph. D.*, page 1, 2004.

[74] N. Museux and J. Vanbockryck. Event based heterogeneous sensors fusion for public place surveillance. In *Information Fusion, 2007 10th International Conference on*, pages 1–8. IEEE, 2007.

[75] B. Myers, S. Hudson, and R. Pausch. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):3–28, 2000.

[76] A. Nazari Shirehjini. Situation modelling: a domain analysis and user study. In *Intelligent Environments, 2006. IE 06. 2nd IET International Conference on*, volume 2, pages 193–199. IET, 2006.

[77] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. *Computer Vision–ECCV 2006*, pages 490–503, 2006.

[78] A. Pentland. Society's nervous system: Building effective government, energy, and public health systems. *Computer-IEEE Computer Magazine*, 45(1):31, 2012.

[79] D. Pospelov. Situation-driven control: Theory and practice, 1986.

[80] D. A. Pospelov. *Situational Control: Theory and Practice(in Russian))*. Nauka, 1986.

[81] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, S. Patil, A. Flammini, and F. Menczer. Truthy: Mapping the spread of astroturf in microblog streams. In *Proceedings of the 20th international conference companion on World wide web*, pages 249–252. ACM, 2011.

[82] C. Ratti, S. Williams, D. Frenchman, and R. Pulselli. Mobile landscapes: using location data from cell phones for urban analysis. *ENVIRONMENT AND PLANNING B PLANNING AND DESIGN*, 33(5):727, 2006.

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

[84] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, 2001.

[85] G. Ritter and J. Wilson. *Handbook of computer vision algorithms in image algebra*. CRC, 2001.

[86] A. Rosenthal, U. Chakravarthy, B. Blaustein, and J. Blakely. Situation monitoring for active databases. In *Proceedings of the 15th international conference on Very large data bases*, pages 455–464. Morgan Kaufmann Publishers Inc., 1989.

[87] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[88] N. Sarter and D. Woods. Situation awareness: A critical but ill-defined phenomenon. *The International Journal of Aviation Psychology*, 1(1):45–57, 1991.

[89] B. Sheridan. A trillion points of data. *Newsweek*, 153, 2009.

[90] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[91] J. Shuai, P. Buck, P. Sockett, J. Aramini, and F. Pollari. A gis-driven integrated real-time surveillance pilot system for national west nile virus dead bird surveillance in canada. *International Journal of Health Geographics*, 5(1):17, 2006.

[92] V. Singh, M. Gao, and R. Jain. Situation detection and control using spatio-temporal analysis of microblogs. In *Proceedings of the 19th international conference on World wide web*, pages 1181–1182. ACM, 2010.

[93] V. Singh, M. Gao, and R. Jain. Social pixels: genesis and evaluation. In *Proceedings of the international conference on Multimedia*, pages 481–490. ACM, 2010.

[94] V. Singh, M. Gao, and R. Jain. Situation recognition: An evolving problem for heterogeneous dynamic big multimedia data. In *(to appear in) Proceedings of the 16th ACM international conference on Multimedia*. ACM, 2012.

[95] V. Singh and R. Jain. Situation based control for cyber-physical systems. In *IEEE workshop on Situation Management, MilCom,*, 2009.

[96] V. Singh and R. Jain. Structural analysis of the emerging event-web. In *Proceedings of the 19th international conference on World wide web*, pages 1183–1184. ACM, 2010.

[97] V. Singh, R. Jain, and M. Kankanhalli. Motivating contributors in social media networks. In *Proceedings of the first SIGMM workshop on Social media*, pages 11–18. ACM, 2009.

[98] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12):1349–1380, 2000.

[99] A. Smirnov, A. Kashevnik, T. Levashova, M. Pashkin, and N. Shilov. Situation modeling in decision support systems. In *Integration of Knowledge Intensive Multi-Agent Systems, 2007. KIMAS 2007. International Conference on*, pages 34–39. IEEE, 2007.

[100] K. Smith and P. Hancock. Situation awareness is adaptive, externally directed consciousness. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):137–148, 1995.

[101] P. Smith and R. Sarfaty. Creating a strategic plan for configuration management using computer aided software engineering (case) tools. Technical report, EG and G Idaho, Inc., Idaho Falls, ID (United States), 1993.

[102] B. Spitzberg, M. Tsou, L. An, D. Gupta, and J. Gawron. The map is not which territory?: Speculating on the geo-spatial diffusion of ideas in the arab spring of 2011. International Communication Association Conference, Phoenix, AZ, 2012.

[103] A. Steinberg, C. Bowman, and F. White. Revisions to the jdl data fusion model. Technical report, DTIC Document, 1999.

[104] K. Takata, J. Ma, B. Apduhan, R. Huang, and N. Shiratori. Lifelog image analysis based on activity situation models using contexts from wearable multi sensors. In *Multimedia and Ubiquitous Engineering, 2008. MUE 2008. International Conference on*, pages 160–163. IEEE, 2008.

[105] J. Ullman. *Principles of database systems*. Egully. com, 1983.

[106] C. Viedma. Mobile web mashups. 2010.

[107] F. Wang, S. Liu, P. Liu, and Y. Bai. Bridging physical and virtual worlds: complex event processing for rfid data streams. *Advances in Database Technology-EDBT 2006*, pages 588–607, 2006.

[108] Y. Wang. An fsm model for situation-aware mobile application software systems. In *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference*, pages 52–57, 2004.

[109] Y. Wang. An fsm model for situation-aware mobile application software systems. In *Proceedings of the 42nd annual Southeast regional conference*, pages 52–57. ACM, 2004.

[110] E. Whittaker. *A treatise on the analytical dynamics of particles and rigid bodies*. CUP Archive, 1937.

[111] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 407–418. ACM, 2006.

[112] Q. Yang, J. Snyder, and W. Tobler. *Map projection transformation: principles and applications*. CRC, 2000.

[113] S. Yau, S. Gupta, F. Karim, S. Ahamed, Y. Wang, and B. Wang. Smart classroom: Enhancing collaborative learning using pervasive computing technology. *II American Society of Engineering Education (ASEE)*, 2003.

[114] S. Yau and J. Liu. Hierarchical situation modeling and reasoning for pervasive computing. In *Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006. The Fourth IEEE Workshop on*, pages 6–pp. IEEE, 2006.