Source Profiling for Smart City Sensing

Yihong Zhang

School of Computer Science
The University of Adelaide

This dissertation is submitted for the degree of

*Doctor of Philosophy*

Supervisors: Prof. Michael Sheng and Dr. Claudia Szabo

October 2016
Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

I also give permission for the digital version of my thesis to be made available on the web, via the University’s digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Yihong Zhang

October 2016
To My Mother and Father

Who are in my flesh and soul
Acknowledgements

This thesis is the result of contributions from many people. I would like to take this opportunity to acknowledge their great effort.

I am deeply indebted to my supervisor, Dr. Claudia Szabo, who showed me the gate to a research career when I was still a master student and uncertain about my future. At the beginning of my PhD, she trained me and instilled in me the necessary skills for conducting scientific researches, through our numerous meetings, discussions, and arguments. I would not have acquired these skills without her enormous patience and persistence. When I started writing research papers, she also assisted me in working through numerous revisions for each of these papers, some of which fortunately get published and form a large part of this thesis.

I am also thankful to the support of my supervisor, Prof. Michael Sheng, whose wisdom and generosity have helped me through the more difficult times in my PhD. Prof. Michael Sheng always encouraged me to aim high and strive for excellence, setting himself as a good example. These encouragements in the end allowed me to avoid many problems and traps in a research career. Without his kind guidance, my progress would certainly be much more tedious and bitter.

I would also like extend my thanks to my senior colleagues Lina, Yongrui, Kewen, Ali and Scott, who have offered useful advises and shared their experiences to help me get through my PhD. And also to my colleague Susie, who helped me conduct some of the experiments presented in this thesis. And also to my colleagues Javier, Lachlan, and Nguyen, who proof-read many chapters in this thesis and provided interesting feedbacks.
I would also like to thank my friends Tony, David and Nick. The time I spent hanging out with them provide me the much-needed relief in a rigid process.

Finally I thank my mother and father for their patience, encouragement and support in this extraordinary long journey of three-year-and-a-half.

This thesis is supported by the Faculty of ECMS Divisional Scholarship, the University of Adelaide.
Abstract

Source Profiling for Smart City Sensing

by

Yihong Zhang

Doctor of Philosophy in Computer Science
The University of Adelaide, 2016

Recent years have seen the emergence of smart cities, which utilize various sensing data for applications such as pollution monitoring, infrastructure planning and traffic control. Current sensing projects tend to deploy a large number of low-cost and unreliable sensing sources, rather than a small number of high-quality sensing sources. It is therefore critical to provide data analysis in the face of unreliable sources.

This thesis focuses on two types of sensing sources that have been used in smart city sensing projects, namely, environmental sensors and human sensors. The environmental sensors are physical sensors that are made to monitor certain environmental features, such as temperature, humidity, and pollutant concentration. An environmental sensor can fail frequently and will start generating faulty data when there is chemical compound decay, battery exhaustion, or calibration problems. Human sensors, as recently proposed in a new area called social sensing, are online messaging platform users who post observations about their surrounding environments. The data generated by human sensors can be erroneous because the natural language used in their messages does not conform to a machine-readable
standard. Based on a survey of existing literature, this thesis presents source profiling-based solutions for three data analysis problems, data cleaning in environmental sensing, observation message classification in social sensing, and message location inference. Each of the solutions is validated with various real-world data and extensive experiments.

For data cleaning in environmental sensing, we propose two solutions, approaching from a frequentist perspective and a Bayesian perspective, respectively. The frequentist approach determines sensor reliability based on the frequency of reliable behavior in the past, and in each data collection iteration updates a reliability score, which can be used to weight down or remove the data from unreliable sources. The Bayesian approach models sensor reliability as a latent variable, and applies the Expectation Maximization framework to discover the latent sensor reliability and correct reading values for the environmental feature.

For observation message classification, we propose supervised and unsupervised solutions. We propose a supervised solution to distinguish messages according to three perspectives, namely, observation, affection, and speculation. We next propose a supervised solution based on user features such as trending activity, communication status, and writing styles. And finally, we propose an unsupervised solution based on lexical analysis and user profiling in four user attributes, namely, originality, interactivity, objectivity, and topic focus.

For location inference, we propose a solution based on name entity extraction and user message histories. The proposed solution extracts location names from text messages using a gazetteer, and after retrieving a number of past locations from the message history of a user, it applies outlier removal before inferring the current location. Incorporating observation classification and location inference, we propose an event detection system called Sense and Focus (SNAF), which detects real world events based on discussions exchanged on Twitter. A prototype implementation of the system has shown a number of detection results, 54% of which corresponding to real-world events, and in many case detected earlier than news reports, and with less than 1.5km location error.
# Table of contents

List of figures xv

List of tables xvii

Nomenclature xix

1 Introduction 1

1.1 Research Objectives ............................... 3
1.2 Contributions ........................................ 4
1.3 Publications Related to This Thesis .................... 6
1.4 Thesis Structure ...................................... 8

2 Smart City Sensing: Sensors and Twitter Users 11

2.1 Introduction ......................................... 12
2.2 Internet-of-Things Environmental Sensors ................. 14
2.3 Participatory Sensing on Twitter ........................ 18
2.4 Opportunistic Sensing on Twitter ........................ 20
2.5 Summary ............................................. 23

3 Environmental Sensor Profiling for Predicting Correct Reading Values 25

3.1 Overview ............................................. 26
3.2 Related Work ........................................ 29
# Table of contents

3.3 Frequentist Approach: Incremental Reliability Update ........................................ 32
   3.3.1 Faulty Data and Reliability ................................................................. 33
   3.3.2 Influence Mean ..................................................................................... 35
   3.3.3 Incremental Reliability Update .............................................................. 36
   3.3.4 IM-Reduce and IM-Remove ................................................................. 38
3.4 Bayesian Approach: Expectation Maximization ................................................. 39
   3.4.1 Background .......................................................................................... 39
   3.4.2 Likelihood Model ................................................................................... 42
   3.4.3 Expectation Maximization for Finding Reliable Sensors .......................... 43
   3.4.4 Final Algorithm ...................................................................................... 46
3.5 Experimental Analysis ....................................................................................... 47
   3.5.1 Simulated Air Pollution Sensing Data .................................................... 48
   3.5.2 Real Data with Synthetic Faults .............................................................. 50
   3.5.3 Melbourne Weather Data ....................................................................... 56
   3.5.4 OpenSense Ozone Data ........................................................................ 59
   3.5.5 Discussion .............................................................................................. 62
3.6 Summary ......................................................................................................... 62

4 User Profiling for Classifying Observation Tweets .............................................. 65
   4.1 Overview ..................................................................................................... 66
   4.2 Related Work ............................................................................................. 69
   4.3 Message Perspective Classification ........................................................... 73
      4.3.1 Message Perspectives .......................................................................... 74
      4.3.2 Identifying Lexical Features ............................................................... 76
      4.3.3 Identifying Textual Features ............................................................... 78
      4.3.4 Machine Learning Classifiers ............................................................ 78
   4.4 Supervised Observation Classification with User Features .......................... 80
Table of contents

4.4.1 Generating User Features ........................................ 80
4.4.2 Trending Activity Features ..................................... 81
4.4.3 Communication Status Features ................................. 82
4.4.4 Writing Style Features ........................................... 82
4.5 Unsupervised Observation Classification ......................... 84
  4.5.1 Observation Filtering ........................................... 84
  4.5.2 User Profiling for Personal Account Classification .......... 88
  4.5.3 Personal Account Classification with Profiles ................ 91
  4.5.4 Overall Algorithm ............................................. 95
4.6 Experimental Analysis ............................................. 95
  4.6.1 Message Perspective Classification .......................... 96
  4.6.2 Supervised Classification with User Features ............... 100
  4.6.3 Unsupervised Observation Classification ..................... 104
4.7 Summary .......................................................... 109

5 User Location Profiling for Localized Event Detection ............ 111
  5.1 Overview ........................................................ 112
  5.2 Related Work .................................................... 114
  5.3 SNAF: Location Inference and Event Detection .................. 116
    5.3.1 Location Extraction for A Single Tweet .................... 117
    5.3.2 Location Resolution Given Past Locations ............... 120
    5.3.3 Realtime Event Monitoring .................................. 125
  5.4 Experimental Analysis .......................................... 127
    5.4.1 Datasets .................................................... 128
    5.4.2 Measurements and Baseline Methods ....................... 129
    5.4.3 Location Accuracy Results ................................ 129
    5.4.4 Event Detection Results .................................. 131
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>A Prototype Realtime Event Monitoring System</td>
<td>133</td>
</tr>
<tr>
<td>5.6</td>
<td>Discussion</td>
<td>136</td>
</tr>
<tr>
<td>5.7</td>
<td>Summary</td>
<td>137</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>139</td>
</tr>
<tr>
<td>6.1</td>
<td>Thesis Summary</td>
<td>139</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Insights of Current Situation</td>
<td>139</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Developing Data Cleaning Techniques</td>
<td>140</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Applying Cleaned Data</td>
<td>142</td>
</tr>
<tr>
<td>6.2</td>
<td>Limitations</td>
<td>143</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Unreliable Sensor Behavior Assumption</td>
<td>143</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Consistent User Feature Assumption</td>
<td>143</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Multiple Sensing Source Requirement</td>
<td>144</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Access to Message History Requirement</td>
<td>145</td>
</tr>
<tr>
<td>6.3</td>
<td>Extending Results</td>
<td>145</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Data Integration with Unreliable Sources</td>
<td>145</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Online Rumor Detection</td>
<td>146</td>
</tr>
<tr>
<td>6.4</td>
<td>Future Works</td>
<td>147</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Sensor Fault Detection with Multiple Features</td>
<td>147</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Improving Social Sensing by Exploiting Friend Network</td>
<td>148</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Mash-up of Environmental Sensing and Social Sensing</td>
<td>148</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>151</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Curriculum Vitae</td>
<td>167</td>
</tr>
</tbody>
</table>
List of figures

3.1 Incremental Reliability Update .................................................. 32
3.2 Faulty sensor data patterns ......................................................... 33
3.3 Environmental readings within one hour ...................................... 41
3.4 Simulated Air Pollution Data ..................................................... 49
3.5 Simulated Air Pollution Noisy readings ....................................... 50
3.6 IM methods on Simulated Air Pollution Data ............................... 51
3.7 Intel Lab Data with Synthetic Faults .......................................... 52
3.8 IM methods on Intel Lab Data .................................................... 54
3.9 EM method on Intel Lab Data .................................................... 55
3.10 The Melbourne Weather Data Sensor Location ........................... 56
3.11 The Melbourne Weather Data .................................................. 57
3.12 IM methods on Melbourne Weather Dataset .............................. 58
3.13 EM method on Melbourne Weather Dataset .............................. 59
3.14 The OpenSense Ozone Data ..................................................... 60
3.15 IM methods on OpenSense Dataset ........................................... 61
3.16 EM method on OpenSense Dataset ........................................... 61

4.1 Supervised Message Perspective Classification ............................ 73
4.2 Unsupervised Observation Classification ..................................... 85
4.3 Perspective classification accuracy with lexical features ................. 98
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Perspective classification accuracy results with lexical and textual features</td>
<td>99</td>
</tr>
<tr>
<td>5.1</td>
<td>The overall architecture of SNAF: Sense and Focus</td>
<td>117</td>
</tr>
<tr>
<td>5.2</td>
<td>Mean location differences and percentages for tweets in different past months</td>
<td>121</td>
</tr>
<tr>
<td>5.3</td>
<td>Event detection based on connected components</td>
<td>126</td>
</tr>
<tr>
<td>5.4</td>
<td>Identified report locations in North America</td>
<td>130</td>
</tr>
<tr>
<td>5.5</td>
<td>The map view of the event detection system</td>
<td>134</td>
</tr>
<tr>
<td>5.6</td>
<td>The tweet view of the event detection system</td>
<td>135</td>
</tr>
</tbody>
</table>
List of tables

2.1 Traditional Sensor Network vs. IoT Sensing Project .......................... 16
2.2 Sensor-generated Twitter accounts and typical refresh rate ................. 20

3.1 Reading Prediction Mean Square Error ........................................... 62

4.1 Examples of tweets in a given perspective ....................................... 74
4.2 Effective lexical categories for different perspectives .......................... 77
4.3 Selected lexical categories and example words .................................. 77
4.4 Textual features for perspective classification ..................................... 78
4.5 Example feature vector ..................................................................... 79
4.6 User Feature Sets ............................................................................. 81
4.7 Business Account Identifier .............................................................. 83
4.8 Originality Test Rules ....................................................................... 83
4.9 Non-observation tweets filtered by POS tagging, for monitoring flight delay, shooting incidents, and rainbows ........................................... 86
4.10 Examples of specific-purpose accounts .............................................. 89
4.11 Perspective classification accuracy with lexical features ....................... 97
4.12 Perspective classification accuracy results with lexical and textual features . 97
4.13 Classification Precision .................................................................... 102
4.14 Classification F-value ....................................................................... 103
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.15</td>
<td>Filtering accuracy for hailstorm and car accident datasets</td>
<td>106</td>
</tr>
<tr>
<td>4.16</td>
<td>Filtering accuracy for the crisis dataset</td>
<td>108</td>
</tr>
<tr>
<td>5.1</td>
<td>Examples of non-placenames in the geo-coordinate dataset</td>
<td>118</td>
</tr>
<tr>
<td>5.2</td>
<td>Precision of location extraction on single tweets using refined gazetteer</td>
<td>119</td>
</tr>
<tr>
<td>5.3</td>
<td>Examples of placename in tweets not indicating user location</td>
<td>120</td>
</tr>
<tr>
<td>5.4</td>
<td>Precision and recall for Random Forest</td>
<td>128</td>
</tr>
<tr>
<td>5.5</td>
<td>Accuracy of location inference methods</td>
<td>131</td>
</tr>
<tr>
<td>5.6</td>
<td>Examples of detected events</td>
<td>132</td>
</tr>
</tbody>
</table>
Nomenclature

Roman Symbols

CDE Crime and Disaster Events
EM Expectation Maximization
GPS Global Positioning System
IM Influence Mean
KNN k-Nearest Neighborhood
LDA Linear Discriminant Analysis
LIWC Linguistic Inquiry and Word Count
LLSE Linear Least-Squares Estimation
MSE Mean Square Error
NGO Non-governmental organization
POI Point-of-interest
POS Part-of-speech
PSO Particle swarm optimization
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>Random Forrest</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio-frequency identification</td>
</tr>
<tr>
<td>SNAF</td>
<td>Sense and Focus Event Monitor System</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In recent years we have seen the emergence of smart cities [1–3], which incorporate the Internet-of-Things (IoT) and all aspects of communication networks, distributed and mobile computing, data management, and knowledge extraction. A smart city can be seen as a digital version of a city, and the foundation of such a smart city is various sensing data. An example is SmartSantander\(^1\). This Spanish port city of 178,000 inhabitants has been a testbed for smart city technologies since 2009, and offers 20,000 sensors spread across the city, including light sensors, temperature sensors, noise sensors, pollutant sensors, among others [1]. These sensors enable applications such as smart homes [4], smart traffic control [5], pollution monitoring [6], health support [3], and infrastructure planning that altogether form a smart city [2].

Two important sensing sources can be identified, namely, environmental sensors and human sensors [7, 8]. Environmental sensors are physical sensors that are made to monitor certain environmental features, such as temperature, humidity, and pollutant concentration. Their sole function is to provide reading values for the environmental features they are monitoring. Human sensors, a relatively new concept introduced only in recent years [8], are online messaging platform users who post observations about their surrounding environments.

\(^1\)http://www.smartsantander.eu/
While not as obvious as the reading values provided by environmental sensors, these observation messages can be used to generate information about certain objects or events of interest, such as earthquakes, road accidents, and public riots, which are also critical in smart city applications [10].

One common problem shared by environmental sensors and human sensors is that, from the data perspective, they are unreliable and tend to produce noisy and erroneous data [11, 12]. A physical sensor may produce an erroneous reading because of chemical compound decay, battery exhaustion, or a calibration problem [13]. These are very common issues for the commodity sensors used in many smart city sensing projects. A human sensor makes an erroneous reading usually not because the user makes a mistake in their expression, but because the natural language used does not conform to a machine-readable standard [14]. For example, the correct location of a report of a road accident may be unrecognizable because the users employed an informal or shortened location name.

For human sensors, another prominent problem is the sparsity of relevant messages [15]. For example, suppose that an application is collecting earthquake-related messages. A user may only post a few messages about ongoing earthquakes among his thousands of messages. Even if we filter the messages using keywords, the result message can still be highly noisy. For example, suppose that a crime monitoring system is collecting messages containing the keyword “shooting”, but the keyword “shooting” can mean the shooting of basketball, camera, or pistol in a message, and only in the last case the message is needed. More advance data analysis techniques are required to filter the messages for particular classification purposes. Existing approaches including machine learning methods and name entity extraction methods [16, 17].

We investigate source-profiling techniques to address these problems. Once we can determine whether a physical sensor is working or faulty, we can easily obtain better quality data by removing the faulty data from the faulty sources. Similarly, for human sensors, we
can obtain better data if we know who is more likely to post observations of objects or events of interest. For example, a basketball team manager is less likely to discuss the shooting of pistols when mentioning “shooting” in their messages. The primary hypothesis in this thesis is that by incorporating source information into data analysis, we can remove noises and errors and obtain better quality data. Therefore the goal of this work is twofold: first, to develop sensing data analysis techniques based on source profiling. For environmental sensors, we focus on identifying if the sensor is working or faulty based on the data it generated. For human sensors, we incorporate more source information into our techniques, including writing style, communication status, and past locations. Second, to validate if incorporating source information indeed improves data quality. This will be done using various synthetic as well as real-world datasets that we manually collected or have been made publicly available.

1.1 Research Objectives

Projects that deploy environmental sensors and human sensors serve the purpose of helping people understand their surrounding environments. However, noises and faults frequently generated by both types of sensors hinder this purpose, and make the information in question difficult or impossible to understand. The primary goal of the research presented in this thesis is to provide insights and solutions that can help with removing noises and improving information accuracy. Specifically, this research aims to reach the following goals:

- To provide insights into the current situation of IoT-inspired environmental sensing and human sensor-based social sensing. It is known that commodity sensors such as those deployed in most IoT environmental sensing projects are unreliable and tend to generate faulty data [18]. It is also known that the informal natural language used in composing short messages on social networks prevents accurate extraction of
relevant information from such messages [16]. This research aims to reveal the degree of the problem these issues can lead to, and the causes behind such issues.

- **To develop data cleaning techniques that improve the information accuracy given unreliable, noisy data.** This research will first investigate a wide range of existing data cleaning techniques, such as outlier removal, Bayesian inference, and machine-learning-based short message classification. Then it will advance novel techniques that can produce better data cleaning results than existing approaches. Techniques for environmental sensing data and social-sensing data may be developed separately because of the different nature of the data. Given that source information is a factor largely overlooked in existing literatures, this research will put an emphasis on source-based techniques. The effectiveness of proposed techniques will be validated using real-world data.

- **To show the potential application in which improved data can be used.** Once we clean the data and obtain information with higher accuracy, subsequent analysis of the data can also be improved. In environmental sensing projects, more accurate data can lead to better prediction of the correct environmental features. In social sensing projects, more accurate data can provide better detection of particular objects or events of interest. It is also in the goal of this thesis to show how cleaned data helps with achieving better results in subsequent data analysis.

### 1.2 Contributions

The contributions this thesis make to the domain of data mining and knowledge extraction are summarized as following.

- We provide an overview of current status of smart city sensing projects, a relatively new phenomenon, and survey a wide range of relevant literature. Although recent
years have seen the initiation of many smart city sensing projects, there is an absence of works that provide overviews for the current state-of-art, particularly covering both environmental sensing and social sensing. The overview provided in this thesis shows impacts, diversities, techniques, and drawbacks for current environmental sensing and social sensing projects, from a data-centric perspective. This overview provides an important foundation for data analysis in smart city sensing data.

• We propose two techniques for cleaning environmental sensing data. The first technique calculates sensor reliability from a frequentist perspective, which considers the reliability of a sensor as how often it produces correct readings. The second technique incorporates sensor reliability into a Bayesian model, and deploys an existing framework, Expectation Maximization, to infer the latent reliability values and correct readings. With real smart city environmental sensing data, we show that the proposed techniques produce better results when compared to existing approaches such as mean, median, and Linear Least-Squares Estimation (LLSE). Particularly for a temperature dataset where working and faulty sensors produce very different data, the proposed approaches can generate almost no error, while other approaches generating mean square errors over 30.

• We propose three techniques to filter observation messages from microblogging data. The first technique identifies each message as observation, affection, or speculation, using a supervised learning approach. The second technique, also using a supervised learning approach, incorporates user features such as communication status and writing styles for classifying observation messages. The third technique, using an unsupervised approach, classifies personal observations based on lexical analysis and user profiling. Experiments with real Twitter data show that each of these techniques improves observation filtering accuracies to some degree compared to existing approaches. Par-
particularly, the unsupervised approach can consistently improve classification accuracy by around 22%.

- We propose a technique to complete the missing location information in many microblog messages. This technique infers the location of a message by looking at users’ past messages and the place names mentioned in the messages. To overcome the noises in the location data, it deploys outlier removal techniques introduced in sensor network literatures. Experiments with real Twitter data show that the proposed technique can match a location with 87% messages, significantly improve the availability of location information, with better the inference accuracy than existing approaches.

- We introduce Sense and Focus (SNAF), a system that monitors object or events of interest based on Twitter data. SNAF incorporates three main components, the observation classification and location inference that are introduced earlier in the thesis, and an event detection system based on connected component graphs. With a prototype implemented using Java and PHP, the system is shown to be capable of detecting real-world events in realtime, in many instances earlier than news reports.

1.3 Publications Related to This Thesis

Over ten published or submitted papers were produced during my PhD study, and among them seven are direct results of the research presented in this thesis, and form the main body in many parts of this thesis. These publications are listed below, and the chapter on which each publication is based is shown in brackets.


6. (under review) **Yihong Zhang**, Claudia Szabo, Quan Z. Sheng. Improved Object and Event Monitoring on Twitter By User Profiling. Submitted to *The 17th International Conference on Web Information Systems Engineering*. (Chapter 4)

1.4 Thesis Structure

This thesis is organized into six chapters. Chapter 2 provides a detailed discussion of the background this thesis is based on. It shows the current situation of smart city sensing, which includes sensing projects using environmental sensors and human sensors. The chapter first shows actual environmental sensing and social sensing projects, the latter of which involves using human as sensors, then identifies existing techniques, issues, and solutions. From a data-centric perspective, it identifies a critical problem in both types of sensing data analysis, namely, cleaning noisy and erroneous data, and provides a survey of existing works related to this problem. For environmental sensing, the problem is represented in erroneous sensor readings. For social sensing, the problem is represented in two aspects: large portion of unrelated messages, and missing location information in most of the messages.

Chapters 3, 4, and 5 present solutions developed to address specific aspects of the identified problem. In each of these chapters, we first provide a review of related work, then we present the solution, followed by a discussion of experimental results.

In Chapter 3, we present our work in cleaning environmental sensing data. We approach the problem from two perspectives, namely, frequentist and Bayesian. In the frequentist approach, we define sensor reliability depending on how often the sensor provides correct reading value, and used the reliability scores to weight down or remove faulty data. In the Bayesian approach, we model sensor faults as a latent variable, and using an existing estimation framework, Expectation Maximization for discovering the latent faults and the correct reading values. In the experiments, we test these approaches on simulated environmental sensing data, as well as real data we collected from smart city sensing projects, and show cleaning accuracy.

In Chapter 4, we present our work in classifying observations from microblog messages. To filter the environmental observation messages from a large portion of unrelated messages, we introduce three classification solutions. The first two solutions are based on supervised
machine learning, and focus on message perspectives and user information, respectively. The third solution is an unsupervised solution based on lexical analysis and user profiling. In the experiments, we test these solutions on real Twitter data we collected and organized, as well as publicly available testing datasets, and show classification accuracy.

In Chapter 5, we present our work in message location inference and event detection. We propose a dictionary approach for extracting message locations based on the place names mentioned in users’ message histories, then we propose using outlier removal techniques to clean the generated location data. After obtaining the location, we propose a subsequent data analysis, namely, event detection, which is based on the accurate location inference in the previous data analysis step. In the experiments, we test location inference accuracy with real Twitter data we collected and show inference accuracy. We also test our event detection approach and show the detection results.

Chapter 6 concludes this thesis by providing final remarks and a discussion of future directions.
Chapter 2

Smart City Sensing: Sensors and Twitter Users

After establishing our research objectives, our second chapter will discuss the background on which this thesis is based, namely smart city sensing. Smart city is a phenomenon emerged in recent years that provides new research opportunities in areas of computer science including data management, knowledge discovery, and mobile systems. Sensing is a crucial component in smart cities, as data generated by various sensing technologies lies at the foundation of smart city infrastructures. This research focuses on two types of sensing sources, namely, environmental sensors and human sensors. Environmental sensors are physical sensors that measures environmental properties such as temperature, humidity, and pollutant concentration. In smart city sensing projects, the environmental sensors usually have Internet connectivity and mobility. The human sensors are microblog users who post observations of their environments. Following a recent research trend [9, 8], we model Twitter users as human sensors. We provide a survey of works related to these two type of sensing sources. Research issues, challenges, and existing solutions are discussed.
2.1 Introduction

In 2008, the number of small devices connected to the Internet had exceeded the number of the world’s population, and it is estimated that by 2020, there will be more than 50 billion such Internet-connected small devices [19]. These small devices, including sensors, actuators, and RFID-tagged objects, are collectively called the Internet-of-Things (IoT) [20, 21]. Over the last few years, works has lied foundation for managing and utilizing such a device network, including communication protocol [22], ontology [23], search engine [24–28], and data management [29–31].

The advance of IoT has enabled what is called a smart city vision. The building blocks of Internet-of-Things are sometimes referred to as smart objects [4]. By attaching RFID tags, for example, to daily objects such as clothes, keyholders, and coffee mugs, data and knowledge about these objects can be easily generated [32]. Development guidelines and tools have been given for users to participate in IoT with their own devices [33, 34, 18]. The Internet-connected electricity and water meters, also called smart meters, have been conserving energy consumption in numerous households and helping the development of infrastructures [35–37]. Smart city is a vision of a city governed by various data generated from smart objects that are present everywhere in the city. As Roberto Saracco, Chair of IEEE’s Future Directions Committee, said in 20141:

“You have two versions of cities, one is made of atoms. You and me, and cars, and all this sort of thing. We are atoms. Then you have another version which is a mirror city, which is made of bits. And the two are connected to one another. If you have sensors in the one, it can connect to the other.”

Sensing is a crucial component in smart cities. Previously emerged in so-called urban computing, sensing technology and sensing data are at the foundation of most smart city applications, such as traffic congestion analysis [38], energy consumption analysis [37], and

1http://sites.ieee.org/spotlight/a-vision-of-a-smart-happy-city/
pollution solutions [39]. For example, air pollution monitoring in cities such as Zurich, San Francisco and Nashville are carried out by sensors installed on vehicles that runs in the city [40–43]. This leads researches to propose sensing as a service model in smart cities supported by IoT and RFID technologies [2, 44]. In recent years, smartphones have emerged as an important sensing source in cities, given their wide ranges of sensing capacities, including accelerometer, digital compass, gyroscope, GPS, microphone, and camera [45, 46]. Frameworks that utilize smartphone sensing have been used in human behavior recognition, hot spot detection, and road bump monitoring [47, 48]. Some techniques have been proposed to recognize surrounding environments using sound signatures sensed by smartphones [49].

In our research of smart city sensing, we focus on two types of sensing sources: environmental sensors and human sensors. The first type are physical sensors that are made to monitor certain environmental features, such as temperature, humidity, and pollutant concentration. For example, pollutant concentration is usually detected by the reaction of certain chemicals placed in an electrochemical sensor [50]. In smart city applications, these sensors are connected to the Internet, and their readings are immediately available online [40]. In Section 2 of this chapter, we will provide an in-depth discussion of this type of environmental sensing.

The second type of sensors, perhaps the more interesting one, are human sensors. Given that humans have excellent capacity to sense and process observed information, the use of humans as sensors in modern cities has been a widely-studied subject [8]. A platform that has been helping to realize human sensing capacity in recent years is microblogging, which has user-friendly interfaces that allows human users to easily create and share messages, and even collaborate in projects [51]. Twitter, a highly-popular microblogging service, for example, generates more than 500 million short messages every day, and many of which are personal observations of environments [52]. These observations posted on microblogs have been used in earthquake monitor [53] and disease transmission prediction [54]. We have
seen a group of studies that has been dedicated to these kind of human sensing, sometimes called social sensing or community sensing [55, 38, 6].

There are two ways to generate human sensing data, namely, using human as the sensor operator, or using the human itself as the sensing source [56]. Both can deploy microblog as the data platform, the first usually by asking human operator to submit data intentionally [7, 57, 58], while in the second way data is collected from the user without their awareness [9, 59]. Depending on whether the user is aware of the data collection process, a social sensing project generally belongs to one of two distinct categories: participatory sensing or opportunistic sensing [60, 61]. In section 3 and 4 of this chapter, we will discuss participatory and opportunistic social sensing in detail. There is a third category of social sensing, sensing about human itself, or in other words, using human as the sensing target [62, 63], but is not the focus of this work.

There is an ongoing discussion whether a microblog like Twitter can be an adequate tool for reporting observations, and be deployed as a news media [52, 64]. Although Twitter has seen heavy usage in recent large human movements such as the Iran Crisis [65] and Arab Spring [66], qualities such credibility and consistency on such a platform have been questioned [64, 67, 68]. However, studies have revealed that online discussions have high correlations with offline phenomena [69], and discussions on Twitter and similar services can be very good indicators for predicting real-world events [70, 71].

2.2 Internet-of-Things Environmental Sensors

Urban environmental sensing provides us with a means to understand our surrounding environments. The monitoring of air pollution, ultraviolet radiation, noise, temperature, and humidity using sensors has important applications in safety, health, and disaster prevention [72, 73]. The emerging Internet-of-Things (IoT) is leading us towards a future where billions of devices are connected to the Internet, including sensors, actuators, and RFID tags, and
provide us a range of new sensing sources [29]. For example, Welbourne et al. attached RFID tags to various daily objects such as clothes and coffee mugs, and allowed these objects to become sensing sources for monitoring human behavior patterns [32]. Azizyan et al. proposed a system that utilizes smartphone sensing capacities such as sound and light, for automatic recognition of user surroundings [49]. Tasic et al. introduced a water meter that can sense water usage and provide feedback to users [36]. Wigan et al. discussed a number of issues involved in using data-generating smart meters, such as privacy, data ownership, and imbalance between data producer and consumers [37]. Clarke et al. proposed a framework that collects fitness sensing data from fitness gears and smartphones that a user may carry for smart city planning [3]. Tutorials such as the one proposed by Cvijikj et al. help users to build their own sensing devices and contribute to IoT sensing projects [33]. The IoT-style environmental sensing sources are of most interest to this work.

A typical IoT-style environmental sensor is shown by Devarakonda et al. [74]. They constructed and investigated a mobile air quality sensing box consisting of a sensor, a GPS receiver, a mobile data module, and a microcontroller, for monitoring street air pollution in NJ Turnpike. A similar work is done by Al-Ali et al., who constructed Internet-connected mobile pollutant sensors using a GPRS modem [75]. This type of sensors has already been deployed in several large-scale urban sensing projects. For example, the OpenSense project has air quality sensors installed on trams to monitor the air quality in Zurich [40], and the construction of the mobile sensor used is very similar to the sensor box introduced in [74]. The Common Sense project puts air quality sensors on street sweepers to monitor the air quality in San Francisco [41]. The city of Melbourne has a running project that makes several types of data regarding the city available online, including environmental data from small sensing sites\(^2\). Apart from project-lead sensors, there are also environmental sensors voluntarily contributed by individual users [7], such as the Air Quality Egg [76] and the

\(^2\)https://data.melbourne.vic.gov.au/about
Cicada Tracker\(^3\). For all these projects, sensing data are available immediately and freely online.

The Internet-connected sensor nodes in an IoT sensing project forms a sensor network similar to a traditional wireless sensor network, in that they both regularly generate sensing data that contain time, location, sensor id, and sensor reading. However, there are some critical differences between an IoT sensing project and a traditional sensor network, some of which are shown in Table 2.1.

Table 2.1 Traditional Sensor Network vs. IoT Sensing Project

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>IoT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity</td>
<td>Ad hoc network, data transmitted through sensor nodes until reached data center</td>
<td>Directly connected to the Internet through mobile data network</td>
</tr>
<tr>
<td>Data Frequency</td>
<td>Once an hour</td>
<td>Once half a minute</td>
</tr>
<tr>
<td>Data format</td>
<td>Aggregated data</td>
<td>Full data including individual sensor readings</td>
</tr>
<tr>
<td>Data availability</td>
<td>Administrator may selectively publish aggregated data on the Internet</td>
<td>All data immediately available.</td>
</tr>
</tbody>
</table>

Particularly, the immediate availability of individual sensors creates an opportunity for advanced data analysis techniques. From the data perspective, traditional sensor-network-based sensing projects usually rely on an administrator to collect, analyze, and publish sensing data [31, 77]. However, in IoT-inspired sensing projects, as each sensor connects directly to the Internet, data are usually publicly available to anyone, and data from each individual sensor can be examined. Current data analysis techniques proposed for traditional sensor networks mostly focus on data aggregation and energy consumption, and a common assumption for these techniques is that individual sensor readings will be lost during transmission or aggregation [78–80]. The new generation of IoT sensors will have less aggregation and energy consumption concerns, thus the data analysis will have a different focus. Furthermore,

\(^3\)http://project.wnyc.org/cicadas/
the availability of individual readings allows source-profiling-based data analysis, which is largely overlooked in existing sensor network literatures.

A common problem imposed on traditional sensor networks and IoT sensing projects is the unreliability of sensors. It is well-known in sensor network studies that commodity sensors are unreliable and tend to produce erroneous readings [18, 78]. The issues that cause sensor to behave unreliably include decay of sensor chemical compounds [81, 82], delayed reaction to changing environments [83], and unstable power supplies [11]. Depending on the manufacturing process, some sensors may need calibrations before becoming usable, [84, 85].

In existing sensor network literature, there are two types of solutions that deal with unreliable sensors and data, namely, sensor fault detection and outlier removal. Sensor fault detection methods focus on finding faulty sensors in the network. Ni et al. proposed a probabilistic faulty sensor selection method, based on the assumption that non-faulty sensor behave similarly [86]. Harkat et al. showed faulty sensors in an air quality monitoring network can be detected using nonlinear principal component analysis [87]. Xiao et al. proposed another faulty sensor detection method based on sensor correlation and voting [88]. Lo et al. proposed a distributed Bayesian approach to fault detection that scales well with the size of the network [89]. Outlier removal, on the other hand, focuses on removing faulty data that usually appear as the outliers in the data. For example, Zhang et al. propose an outlier detection method based on aggregation trees [90]. Basu et al. shows outliers in sensor data can be detected with time series data techniques [91]. Zhang et al. introduced another outlier detection method based on the temporal spatial correlation of sensor data [92]. Furthermore, a number of outlier detection techniques in temporal and spatial data are surveyed by Han et al. and Lu et al., respectively [93, 94].

In IoT sensing projects, particularly in projects where users voluntarily contributes sensors, and where the condition, availability, and location of a sensor can change anytime,
sensor reliabilities become even more uncertain [30, 95]. Fortunately, given that IoT sensors generally provide full readings from individual sensors, we can construct source-based techniques that are largely overlooked in existing literature. In Chapter 3, we will give a full discussion for our work on source-based techniques for sensor reliability inference and correct reading prediction.

2.3 Participatory Sensing on Twitter

Microblogging services allow many kinds of messages to be conveniently published and immediately made available online. A part of these messages contains environmental sensing information, particularly in urban areas, where microblog users have their daily activities. Given the huge popularity of microblogging services such as Twitter, the amount of sensing information we can get from them is significant. Therefore using a microblogging service as a sensing platform has attracted a wide range of research in recent years [9, 12, 96].

As mentioned earlier, social sensing projects can be divided into two categories, participatory or opportunistic. A microblogging service like Twitter is fundamentally a service for human users, and we have found that sensing information on it also falls into these two categories. The participatory sensing information mainly comes from automatically generated sensor accounts, and subscribed users will receive updates in real-time when new readings become available. The opportunistic sensing information on the other hand requires an information consumer to actively search for it. It does not require subscription, but to reach the desired information within the huge volume of messages, an information consumer will need highly sophisticated searching and filtering techniques.

We first discuss the participatory sensing information on Twitter. Some recent researches have proposed techniques and frameworks that integrate sensors with Twitter and other microblog services. For example, Baqer et al. proposed S-Sensors, a framework that connects sensor nodes to Twitter and use Twitter as a central platform for distributing and aggregating
sensing information [97]. They show that the options such as privacy control and operations such as following the unfollowing on Twitter is very suitable for distributing and receiving sensing information. Kranz et al. showed the connection of office objects such as coffee mugs and plants to the Internet, also using Twitter as a platform to subscribe to and receive information [98]. Clarke et al. proposed a framework to share fitness sensing information using Twitter as the sharing platform [3]. They showed that this kind of data sharing can be beneficial in several smart city applications such as the public health and urban planning. Zhang et al. proposed sharing and managing home monitoring systems using Weibo, a Twitter-like microblog service, and showed that integrating with the existing infrastructure of such services can significantly reduce the development and management cost of a sensor network [99].

In addition, user-friendly commercial microcontroller boards such as Raspberry Pi⁴ and Arduino⁵ provide tools and interfaces to connect to Twitter. There are also many online tutorials offered by technology enthusiasts that show sharing on Twitter various kinds of sensing data, ranging from earthquakes⁶ to the watering state of a plant⁷.

We investigated actual Twitter accounts and found a wide ranges of sensor-generated accounts that currently exist on Twitter. Table 2.2 shows actual sensor-generated accounts in various categories. Some of these accounts show local weather data such as temperature, humidity, and ultraviolet radiation. Some show air pollutant data such as PM2.5 and carbon dioxide concentration. Some other accounts show the state of certain hardware, such as the CPU temperature of a computer, or the power supply of a fridge. An examination of the message history shows that most of these accounts have rapid and consistent reading updates, similar to an IoT-style sensor node.

---

⁴http://www.reuk.co.uk/Publish-Temperature-Sensor-Readings-to-Twitter-Raspberry-Pi.htm
⁵http://apcmag.com/how-to-tweet-with-your-arduino.htm/
⁷https://www.botanicalls.com/archived_kits/twitter/
Table 2.2 Sensor-generated Twitter accounts and typical refresh rate

<table>
<thead>
<tr>
<th>Category</th>
<th>Accounts</th>
<th>Refresh Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather</td>
<td>@mayfieldhalls @Kilkenny_Met @fenpinzberg @YXEWeather @NadiFJ</td>
<td>once every five minutes</td>
</tr>
<tr>
<td>Air Pollutant</td>
<td>@RespiraChile @USEmbUBAir @xF7YP6X2 @SanDiego_AIR @BeijingAir</td>
<td>once an hour</td>
</tr>
<tr>
<td>UV Radiation</td>
<td>@fairviewpark @wf_yarmouth @LBIWeather @swellbzh @G1CAE</td>
<td>once an hour</td>
</tr>
<tr>
<td>Hardware State</td>
<td>@Pi22Yorks @Dukester94 @tedc_ts_uHome @MrDestructotron @CulturalCafeMon @DaniArd99</td>
<td>six times a day</td>
</tr>
</tbody>
</table>

In this work we put emphasis on the next type of sensing, namely, opportunistic sensing, rather than the participatory sensing discussed here. It is possible to collect the sensor-generated accounts and use them as a sensor network for various applications. However, there are several drawbacks using these accounts as a sensor network. First, the actual coverage is limited. Although it has been proposed by media and technology enthusiast, there are not many people actually connect sensor nodes to Twitter. By monitoring the keyword “temperature” on Twitter for a day, we were only able to capture several hundreds of temperature sensor accounts, which is very sparse at a global scale. Second, the incomplete information makes interpreting sensor meaning difficult. For example, many temperature sensor accounts we found do not provide any location information, while many hardware sensor accounts do not provide hardware specification. Third, the usefulness of many of these sensor accounts is limited. Although we can collect, for example, the CPU temperature of various machines around the world, it is difficult to extract information that will be interesting to the general public. Nevertheless, the data from participatory sensing can be a helpful supplement in a large scale data analysis, and may be considered in future work.

2.4 Opportunistic Sensing on Twitter

In opportunistic sensing, sensing data are collected without notifying the user who posts the data. Twitter allows public access to at maximum 1% of all its data traffics without fees. The
limit applies after optional keyword filtering. Usually if we monitor a certain keyword on Twitter, the filtered data are less than 1% of all Twitter data traffics. In other words, we can usually get all the tweets that contain a certain keyword posted in realtime. Given this access, we can turn Twitter into a sensing platform that monitors a certain event or object of interest that can be expressed by a few keywords.

In participatory sensing, the participants purposefully join a sensing project, and provide data in a format that can be easily understood and analyzed by data consumers. In opportunistic sensing, however, data providers are not expected to submit data in a formal, easy-to-analyze format. When reporting a certain event or object of interest, they usually report it in an informal, unstructured message using natural languages. Opportunistic sensing on Twitter for a particular event or object means we will opportunistically collect these informal and unstructured messages and extract information regarding the event or object from them. This immediately provide us with some challenges, including data analysis given noisy and incomplete data, and unreliable data sources.

Researches have been using opportunistically collected Twitter information for disaster location [10], object tracking [53], and event detection [96]. However, accurately classifying object or event-related messages is a challenging task. Sakaki et al. [53] developed a system that tracks the movement of earthquakes and typhoons based on personal reports detected on Twitter. They compared a number of features for building the event-related message classifier, with the best-performing feature set achieving a precision of 63.64%. Li et al. [10] studied the use of tweets for detecting crime and disaster events (CDE) as they were reported on Twitter. They trained a classifier based on the words present in identified CDE tweets, and achieved a precision of 30% and a recall of 85%.

Various works have proposed techniques for classifying tweets. Some are based on supervised machine learning techniques [100, 101], while others are based on unsupervised techniques such as clustering and sentiment analysis [102, 96, 17]. These works focus on
various applications including rumor detection [103–105], election prediction [106, 107], and business analytics [16, 108]. However, it has been shown that solutions and models effective in one application often will not be as effective in another application. For example, Castillo et al. investigated detecting news and rumors on Twitter in one work, and found that the effective solutions for two investigated applications are different [100].

One approach that may improve the classification of observations is through user profiling. Twitter user modelling and profiling have been widely studied in various works [109, 110]. Some of these works focus on measuring user influences [111–113], while some others focus on user social roles [114–116]. A number of works studied user models in situations such as information diffusion [117, 118] and rumor propagation [119, 120]. For the purpose of our study, we aim to find the kind of users who are more likely to give environmental observations. In Chapter 4, we will present our work on user-based tweet classification for the purpose of event and object monitoring, using both supervised and unsupervised approaches.

To convert a tweet message to a valid sensing value for an event or object, in addition to the report message itself, we need two more pieces of information, namely, time and location. As opposed to time information, which is readily available for all tweets, location information is scarce. Current works for obtaining the geo-location information of tweets rely heavily on the tweets’ GPS data [9, 121]. In one of our experiments, we sampled one thousand random tweets, but found that only 0.9% had GPS data. This implies that if only GPS-enabled tweets are considered, most of the observations posted on Twitter will not be noticed. Thus the where in the tweet has attracted a number of researches [122–125]. To increase the availability of location information, tweet messages and other publicly accessible information on Twitter need to be treated as location sources [14, 10, 126, 127].

Extraction of location information from tweets has been studied by several works [10, 14, 128–132]. Many of these works focus on the city-level location [10, 130, 131]. However,
finer-grained location information is often vital for local events, such as shooting incidents and vehicle crashes. Previous studies show that, exact location extraction from tweet texts often result in large errors [128, 132]. The issues include the lack of a comprehensive gazetteer, the use of informal names, and mentioning of a placename other than the place where the user is located. Currently it is difficult to obtain a comprehensive gazetteer covering street level place-of-interest at a global scale, while also including informal names such as “the egg”, “great bridge”, and “red rocks”. It becomes more complicated when users mention placenames that are not where they are standing or sitting, for example, when discussing holiday destinations. In Chapter 5, we will present our work on techniques for message-based tweet location inference.

2.5 Summary

Emerging smart cities generates a huge amount of sensing data through billions of connected devices the Internet-of-Things. Two important sensing sources are environmental sensors and human sensors. With regard to environmental sensors, we discussed the differences between traditional sensor networks and IoT sensing projects, and identified research challenges and existing approaches. With regard to human sensors, we show how Twitter users are modelled as sensors in a recent research trend, and discussed the issues and existing approaches. A common issue for both types of sensing sources is that they both generate noisy, unreliable sensing data. A common approach that existing works related to two types of sources largely overlook is a source-based approach. Basing on this situation, we researched source-based data analysis approaches for these two types of sources, which will be presented in following chapters.
Chapter 3

Environmental Sensor Profiling for Predicting Correct Reading Values

In the previous chapter, we introduce the current state of smart city sensing and related issues. In this chapter, we will tackle the first problem, which is data cleaning in environmental sensing data. Environmental sensing has become a significant way for people to understand their living environments. However, commodity sensors that are used in most sensing projects are known to be unreliable and prone to produce faulty data [78]. It is an important and challenging problem to predict correct reading values from low quality sensing data. It has been shown that existing prediction approaches are unable to provide satisfactory prediction accuracy [74]. The Internet-of-Things (IoT) inspired sensors provide frequent and detailed sensor readings that allow profiling of individual sensor reliability, which has been largely overlooked in previous literatures. Based on individual sensor reliability profiling, we approach the correct reading value prediction problem from two perspectives: frequentist and Bayesian. We propose and test extensively three prediction algorithms. The experimental results show that the proposed approaches were able to produce predictions much more accurate than the existing approaches.


3.1 Overview

In environmental sensing, sensors are deployed in physical environments to monitor environmental attributes such as temperature, humidity, water pressure, and pollutant concentration. With the emergence of the Internet of Things (IoT), which connects billions of small devices such as sensors and RFID tags to the Internet, environmental sensing is becoming a significant means towards understanding and transforming the environment [29]. In some environmental sensing projects, multitudes of sensors are deployed by institutional or governmental agencies in urban or remote areas. For example, the Common Sense project puts air quality sensors on street sweepers to monitor the air quality in San Francisco [41]. The OpenSense project has air quality sensors on trams to monitor the air quality in Zurich [40]. The city of Melbourne has a running project that makes several types of data regarding the city available online, including environmental data from small sensing sites\(^1\). In some other projects, existing sensors such as the sensors urban residents carry with their smartphones are leveraged for environmental sensing, which is sometimes termed people-centric sensing [62].

Depending on whether the data collection is opportunistic or participatory, users in some cases do not need to be aware of the sensing and data collecting process, as discussed in Chapter 2 [61]. A smartphone remote sensing platform has been proposed to detect the road bumps the in urban area using accelerometer sensors installed on smartphones carried by the drivers [47]. The Citizen Sense project\(^2\) in one experiment puts air quality sensors into shoes, which allows the wearers voluntarily contribute air quality data while walking around the street without active actions. Participatory people-centric sensing projects ask residents to report sensing data using their own devices, usually with incentives [60, 7]. In an experimental project, users are asked to capture and report litters in a hiking area [12].

---

\(^1\)https://data.melbourne.vic.gov.au/about
\(^2\)http://www.citizensense.net/
from sensors on smart phones, dedicated environmental sensors have also been contributed by users, such as the Air Quality Egg[^3] and the Cicada Tracker[^4].

In most environmental sensing projects, inexpensive commodity sensors are deployed instead of high quality industrial sensors to minimize costs. Commodity sensors, however, are widely known to be unreliable and prone to producing noisy and erroneous data, and raw readings from these sensors are often deemed unusable \[18, 78\]. Data cleaning is therefore an important issue in environmental sensing, especially when critical realtime decisions need to be made based on the collected data, such as street-level pollutant warning \[74\]. Recent works have proposed solutions to predict the correct readings from noisy sensor data \[78, 133, 134\]. A common approach to automatically predict correct readings is by aggregating spatially correlated readings, using either mean \[78, 134\] or median \[133\]. However, such approaches have not achieved satisfying accuracy in situations where the data are too noisy. A project that recorded pollution readings using sensors in proper and difficult conditions and preprocessed the data using mean found that the correlation coefficient between readings from two types of sensors only reached 0.59 in linear regression tests \[74\].

Intuitively, knowing individual sensor reliability can improve prediction accuracy by, for example, giving unreliable sensors less weight when aggregating readings, or remove the sensor from data completely. We adopt a data-centric approach to sensor reliabilities, and identify potential sensor malfunctioning through faulty data. Sensor malfunctioning can be classified in two main categories, namely, systematic and random \[135\]. Systematic sensor malfunctioning is usually caused by the crude calibration and the deterioration of sensing material. The faulty data caused by systematic malfunctioning typically can be fixed by a single change in the calibration parameter, as proposed by several works \[135, 85\]. We instead focus on the random malfunctioning, which can be caused by unpredictable issues such as sensor damages in unexpected weather. Random malfunctioning can be permanent.

[^4]: [http://project.wnyc.org/cicadas/](http://project.wnyc.org/cicadas/)
or temporary. For example, when a taxi carrying a sensor drives into a tunnel, the sensor may not function properly due to the difficult condition in the tunnel, but can soon recover as the taxi exits the tunnel. Random malfunctioning usually causes faulty data with unpredictable patterns, which are difficult to anticipate.

Traditionally in sensor network studies, data from individual sensors are not considered, and only aggregated readings from a multitude of sensors are preserved [136]. The main reason is to reduce data traffic in the ad hoc network to prolong the life of the battery the sensor nodes carry. Without data from individual sensors, it is difficult to profile the conditions of individual sensors using data-centric approach. However, the Internet-of-Things-inspired sensor nodes that emerged recently have the capacity of connecting to the Internet directly, as well as extended battery life [74]. Many environmental sensing projects using IoT sensors publish individual sensor readings on the Internet, such as the OpenSense project\(^5\). With this approach, instead of centralized or ad-hoc data collection as in traditional sensor networks, any user who has an Internet connection can access all data from all sensors in the sensor network. In contrast to the traditional aggregated data, in which sensor readings are temporally aggregated by hours or days, the data offered in this manner often include raw sensor readings collected once or twice per minute. This allows profiling individual sensors on an hourly basis.

In engineering, reliability is defined as \emph{the probability that a device will perform its intended function during a specified period of time under stated conditions} [137]. We approach sensor reliability profiling from two perspectives, namely, frequentist and Bayesian. A frequentist perspective would see sensor reliability based on the frequency of reliable behavior in the past. Our frequentist approach determines sensor reliability based on the accumulated reliable or unreliable behavior, using a reward or penalty function, and makes prediction of correct readings using a reliability-weighted mean function. We call our frequentist prediction method \textit{Influence Mean} (IM), as it aggregates sensor \textit{influences} on the

\(^{5}\text{http://data.opensense.ethz.ch/}\)
prediction rather than raw sensor readings. Similar to a question that has been studied in data integration research [138], to investigate whether reducing weight or removing the source completely produces better prediction results, we develop two algorithms, called IM-Reduce, and IM-Remove. In IM-Reduce, sensors with lower reliabilities will be assigned less weight in aggregation, but all sensors are considered. In contrast, IM-Remove will remove sensors whose reliabilities are lower than a threshold from the aggregation.

A Bayesian perspective would see sensor reliability as an unknown quality that becomes clearer with each update of information. More importantly, modeling sensor reliability from a Bayesian perspective allows us to integrate the problem into existing Bayesian prediction tools. Our Bayesian approach involves modeling sensor reliabilities and predictions in a Bayesian framework, and using an Expectation-Maximization (EM) algorithm that iteratively finds the environmental model and the latent states of sensor faults. EM algorithms have proven effectiveness in solving data integration problems [139, 140]. However, a similar technique for environmental sensing data has not been found in existing literature.

We test both approaches extensively using synthetic and real datasets. For the synthetic dataset, we follow a technique proposed by Sharma et al. [13], which first takes a proportion of confirmed clean data, then injects faulty data patterns. For the real dataset, we collected two smart city environmental sensing datasets, and tested our approaches on them without any modification.

3.2 Related Work

In empirical studies, sensor faults and faulty data have been observed frequently. Buonadonna et al. [18] learned from real sensor network deployment that sensors often fail in an unexpected way, and that the calibration and maintenance of sensors are difficult. Jeffery et al. [78] examined an environmental sensing dataset in which a quarter of the sensors in the network failed but continued to provide faulty readings. It is commonly agreed that
Commodity sensors produce low-quality data, and very often the raw sensor readings are unusable. Hermans et al. [136] discussed several sensor data quality measurements including accuracy, consistency, timeliness, and completeness. They employed a set of heuristics to test initial evaluations. Lim et al. [141] studied trustworthiness as a measurement of sensor data quality.

A data-centric approach to detect sensor faults has been studied in several works. Ni et al. [11] investigated different types of sensor malfunctioning (e.g., battery exhaustion and hardware malfunction) and associated their faulty data patterns. They discover patterns such as spike, stuck-at and high variance, but have not proposed systematic detection methods. Sharma et al. [13] identified three faulty data patterns in a number of real datasets, and proposed techniques for their detection. In the evaluation of their techniques, they injected faulty data patterns into known clean data, and used the original clean data as the ground truth. We adopted this data synthesis method when designing our experiments. They also proposed a sensor fault detection method based on linear least squares estimator (LLSE) to which we will compare our methods in the experiments. A type of sensor faults called systematic malfunctioning can be fixed by simple changes in sensor calibration parameters, as proposed by Bychkovskiy et al. [135] and Hasenfratz et al. [85]. We instead deal with random malfunctioning, which cannot be predicted and prevented beforehand. Such faults can occur when a sensor is damaged by unexpected weather, or when a mobile sensor travels through a tunnel.

A large number of works focus on data cleaning in wireless sensor networks [78, 142, 134]. Jeffery et al. [78] proposed a procedure for cleaning sensor data that includes a smoothing phase and a merge phase. In the merge phase, data from spatially correlated sensors are merged using the mean. Zhuang et al. [134] proposed to use moving average and regression for cleaning single sensor data. Given that environmental properties such as temperature or pollutant concentration are temporally and spatially coherent, a more effective
way to clean data is to identify and remove outliers in the data. Branch et al. [143] proposed a distance-based ranking approach for calculating global outliers in the data. Burdakis et al. [144] proposed a geometric approach for identifying outliers in sensor data. Keally et al. [145] proposed clustering-based outlier detection using a Hidden Markov Model. Subramaniam et al. [79] proposed a distance-based outlier detection method using a kernel density estimator. Sheng et al. [146] proposed a histogram approach for identifying outliers in sensor data using multi-level aggregation. Deligiannakis et al. [147] proposed to identify data anomalies based on the multi-level structure of sensor networks. Franke et al. [148] proposed a spatial outlier detection approach based on polygon construction. Giatrakos et al. [149] proposed to calculate approximate outliers in the network using similarity test. A key objective in sensor network researches is to reduce energy consumption of sensor nodes, and many of the above works proposed data cleaning techniques focused on reducing communication between sensor nodes [147, 143, 77, 149]. These techniques are based on the assumption that sensor readings are aggregated when transmitted, and individual sensor readings are not available or difficult to obtain. The IoT inspired environmental sensors, however, are assumed to be connected to the Internet directly, like those used by Devarakonda et al. in [74]. Such direct Internet connection capacity allows individual sensor readings to be accessed and preserved, which creates an opportunity for studying individual sensor behaviors. For example, Hasenfratz et al. [85] proposed an on-the-fly sensor calibration method that, in a network of Internet-connected sensors, changes the calibration parameters of individual sensors based on their respective readings.

Data integration with regard to individual source reliability is an important research topic in information retrieval. In the web environment, it is not unusual for multiple data sources to have different views on a same fact [150]. Galland et al. [151] discussed a number of techniques for truth discovery in accurate web data, including voting, estimations using two and three parameters, and cosine similarity-based clustering. Zhao et al. [152] proposed
a Bayesian approach to discover truthful information from conflicting source. Blanco et al. [153] proposed probabilistic models to clean data from inaccurate sources. Wang et al. [12] proposed a maximum likelihood estimation approach that can be applied to binomial sensing data and web data. Yin et al. [150] proposed another Bayesian network approach for cleaning inaccurate web data. In the sensor network literature, however, due to the absence of data from individual sources, similar techniques are scarce.

3.3 Frequentist Approach: Incremental Reliability Update

Without prior knowledge of sensor hardware information, we calculate individual sensor reliabilities by comparing individual sensor readings with predicted correct readings, and incrementally updating sensor reliabilities in each data collection iteration. The obtained sensor reliabilities are then used to weight aggregations in the next iteration. The overall process of our method is shown in Fig. 3.1. This process repeats in iterations as the data from the sensor network are continuously received, and in each iteration, predicted correct readings are output to data consumer. In this section, we first discuss generic faulty data patterns in real sensor datasets and the definition of sensor reliability. We then introduce our proposed data cleaning method, focusing on influence mean (IM) cleaning and the incremental reliability update model.

![Fig. 3.1 Incremental Reliability Update](image-url)
3.3 Frequentist Approach: Incremental Reliability Update

3.3.1 Faulty Data and Reliability

Sensors can produce noisy and erroneous data when operating in less than ideal working conditions. Fig. 3.2 shows three patterns of faulty data that may be caused by sensor malfunctioning, which are taken from real-world datasets. Fig. 3.2a is the ozone reading from the OpenSense ozone data. Fig. 3.2b and 3.2c are temperature readings from the Intel Lab data. We will describe these two datasets in detail in the experiment section.

(a) High Volatility  
(b) Single Spike  
(c) Intense Spikes

Fig. 3.2 Faulty sensor data patterns
According to the work by Ni et al. [11], high volatility, characterized by a sudden rise of variance in the data, can be caused by hardware failures or a weakening in battery supply. Single spikes, occasional unusually high or low readings occurring in a series of otherwise normal readings, can be caused by battery failure. Intense single spikes that occur with high frequency may indicate hardware malfunction.

We focus on calculating a representative reliability value for each individual sensor that can be used to improve prediction accuracy. In engineering, reliability is defined as the probability that a device will perform its intended function during a specified period of time under stated conditions’ [137]. The intended function of an environmental sensor is to generate correct readings according to the environmental feature that it is monitoring. Consequently, when a sensor produces a reading, sensor reliability indicates the probability that it is the same as the presumed correct value.

It is very rare that a sensor can produce a reading value that is exactly the same as the correct value. To define a reading as consistent with the correct reading value when the difference between the reading value \( r \) and the correct value \( v \) is within acceptable range, we use a tolerance threshold \( \delta \) and calculate the consistency of a reading \( r \) as the following:

\[
cons(r) = \begin{cases} 
1, & |r - v| \leq \delta \\
0, & \text{otherwise}
\end{cases}
\] (3.1)

We calculate the reliability of a sensor as the percentage of the readings made by the sensor that are consistent with the prediction, from the total number of readings the sensor has made during an observation period:

\[
srlb = \frac{1}{n} \sum_{i=1}^{n} cons(r_i)
\] (3.2)
where \( \{r_1, r_2, \ldots, r_n\} \) are the readings made by the sensor. When applying the method to continuous streams, the observation period is usually a moving time window with a fixed length.

The reliability of a sensor is taken as the reliability of its readings, by the definition discussed above. Suppose the set of the sensors are \( \{s_1, s_2, \ldots, s_l\} \). Let \( \{srlb_1, srlb_2, \ldots, srlb_l\} \) be each sensor’s reliability. We define the reliability of a reading as the reliability of the sensor that produced it:

\[
rlb(r) = srlb_i, \quad \text{if } r \text{ was produced by } s_i
\] (3.3)

Our IM cleaning method, shown in Fig. 3.1, consists of two main modules, namely i) the influence mean prediction which takes into account individual sensor reliabilities for predicting correct readings, and ii) the reliability update module. We provide two versions of IM cleaning algorithm, IM-Reduce and IM-Remove, which we will present later. Following the data-centric approach, we do not assume any prior hardware information that can be used to infer the reliability of individual sensors. Usually we set a same value as the initial reliability for all sensors in the network. The continuous operation of the cleaning method consists of iterations. In each iteration, new sensing data are collected, predictions of correct readings are made, and the reliabilities are updated by comparing individual readings to the prediction.

### 3.3.2 Influence Mean

In an environmental sensing application, the true reading value can be predicted as the mean of the readings made by a group of spatially correlated sensors:

\[
P_{\text{MEAN}}(R) = \frac{1}{k} \sum_{r \in R} r
\] (3.4)
where \( R = \{ r_1, r_2, \ldots, r_k \} \) is the set of \( k \) readings produced by the spatially correlated sensors.

If readings from individual sensors are preserved, then each reading will be associated with a sensor. Assume we have a dataset of reading \( R = r_1, r_2, \ldots, r_k \), in which each reading \( r \) is associated with a sensor of known reliability. By using Equation (3.3), we obtain \( \{ rlb(r_1), rlb(r_2), \ldots, rlb(r_k) \} \) as the reliability of each reading. The reliability of a reading indicates the probability of the reading being the true reading. We define the Influence Mean as:

\[
P_{IM}(R) = \frac{\sum_{r \in R} r \times rlb(r)}{\sum_{r \in R} rlb(r)}
\]  

(3.5)

Consider the prediction as a voting process. With simple mean, sensors have equal weights and each sensor has one vote. When a large portion of sensors in the network are faulty and report wrong readings, the prediction becomes very inaccurate. By giving reliable sensors higher weight in the voting process, we shift the prediction closer to the correct value, and effectively, more reliable sensors have stronger influence on the prediction results. We call the prediction defined by Equation (3.5) Influence Mean, in the sense that it does not aggregate specific readings, but the influences of the sensors, which are determined by their reliabilities. For example, assume that a dataset consists of a reading of 15 reported by a sensor of reliability 1, and a reading of 10 reported by a sensor of reliability of 0.5. The correct reading will be predicted as \( \frac{15 \times 1 + 10 \times 0.5}{1 + 0.5} = 13.3 \), out of which 10 was contributed by the first sensor, and 3.3 was contributed by the second sensor. In this case, the contribution of the second sensor to the prediction is a third of the contribution of the first sensor, while with simple mean, it will be two thirds.

### 3.3.3 Incremental Reliability Update

Using the prediction as the correct reading \( v \), we can use the definition of reliability in Equation 3.2 to update the reliability of the sensor. We use an incremental approach so that
3.3 Frequentist Approach: Incremental Reliability Update

no previous records need to be considered, and only the data captured in one iteration is processed. As mentioned earlier, given the unpredictable sensor faults, using incremental update to adjust individual sensor reliabilities in each time frame is more suitable than using a pre-defined model to calculate the reliabilities based on entire past data.

The incremental reliability update formula is derived as following. Suppose that after making \( n \) readings, the reliability calculated for a sensor using Equation (3.2) is \( srlb \). If the sensor has made another reading since then, the new reliability \( srlb' \) can be calculated as:

\[
\begin{align*}
  srlb' &= \frac{1}{n+1} \sum_{i=1}^{n+1} \text{cons}(r_i) \\
  &= \frac{1}{n+1} \left( \sum_{i=1}^{n} \text{cons}(r_i) + \text{cons}(r_{n+1}) \right) \\
  &= \frac{1}{n+1} \sum_{i=1}^{n} \text{cons}(r_i) + \frac{1}{n+1} \text{cons}(r_{n+1}) \\
  &= \frac{1}{n} \sum_{i=1}^{n} \text{cons}(r_i) \times \frac{n}{n+1} + \frac{1}{n+1} \text{cons}(r_{n+1}) \\
  &= srlb \times \frac{n}{n+1} + \frac{1}{n+1} \text{cons}(r_{n+1}) 
\end{align*}
\]

Equation (3.6) provides a formula for updating the reliability of a sensor given the current reliability and a new reading. Substituting Equation (3.1) into the formula gives a reward or penalty function, which allows the sensor to gain or lose some reliability based on its new reading:

\[
srlb' = \begin{cases} 
  srlb + \frac{1 - srlb}{n+1}, & \text{if } \text{cons}(r_{n+1}) = 1 \\
  srlb - \frac{srlb}{n+1}, & \text{if } \text{cons}(r_{n+1}) = 0 
\end{cases}
\]

(3.7)

Usually we choose an observation time window in which a fixed number of past readings from a sensor is considered. As such, the amount of change using above update function is solely determined by the current reliability of the sensor. As an example, suppose that a sensor currently has the reliability of 0.8, and goes on to make one consistent reading, and
one inconsistent reading. The number of readings it makes in the observation time window is always 10. Now after the first reading, its reliability becomes \(0.8 + \frac{(1 - 0.8)}{10} = 0.82\). After the second reading, the reliability becomes \(0.82 - \frac{0.82}{10} = 0.738\).

### 3.3.4 IM-Reduce and IM-Remove

We propose two algorithms, IM-Reduce and IM-Remove. In a data collection iteration, in which sensing data from a sensor network for a period of time are collected in the data center, IM-Reduce use the formula shown in Equation (3.5) and (3.7) to calculate the prediction and update individual sensor reliability. Algorithm 1 shows the IM-Reduce procedure. The prediction \(v\) is calculated using all data collected in the iteration (line 3), and on which the reliability update is depending (line 4).

**Algorithm 1 IM-Reduce**

1: assign 1 to all sensor reliabilities \(srlb_i \in SRLB\)
2: for each \(D\) collected in a data collection iteration do
3: \(v \leftarrow P_{IM}(D)\)
4: update each \(srlb_i\) using individual sensor readings, \(v\) and Equation 3.7
5: end for

IM-Remove is very similar to IM-Reduce, only that in each iteration, data from sensors whose reliabilities are lower than a threshold are excluded. It is important to note that, even though unreliable sensors are excluded from prediction calculation, their reliability updates are run normally. This is to allow these sensors with temporary malfunctions to rejoin the prediction calculation when their reliability recovers. Line 4 to 6 in Algorithm 2 shows the procedure for selecting reliable sensors in IM-Remove, after which there are prediction and reliability update steps similar to IM-Reduce (line 7, 8).

When applied to continuous data streams, the process waits until the end of the predefined time window while the data are being collected. At the end of each time window, IM cleaning is applied to \(D\), the dataset collected in the iteration. In Algorithm 2, a parameter
3.4 Bayesian Approach: Expectation Maximization

Expectation Maximization (EM) is a technique for finding statistic models given that there are latent variables associated with observation data. It is particularly suitable when the model and the latent variables are unknown and independent to each other. In environmental sensing projects, we usually do not have prior knowledge of the environmental model and individual sensor working conditions. Our approach involves building a likelihood function that takes into account both the environmental model and sensor faulty states, and we find that it is straightforward to devise the problem into an EM algorithm. We assume our analysis target is numerical readings with Gaussian noises. In this section we will present our approach in detail.

3.4.1 Background

Expectation Maximization (EM) is a well-known iterative technique in statistics for finding models with latent variables [154]. To devise an EM algorithm for a particular problem, it is

\[ \tau \text{ needs to be set as the threshold that divides reliable and unreliable sensors, which can be based on domain knowledge. } \]

Step 4 to 6 in Algorithm 2 are used to remove data from unreliable sensors. The reliability update is the same for Algorithm 1 and 2.

Algorithm 2 IM-Remove

1: $\tau \leftarrow$ reliability threshold of reliable sensors
2: assign 1 to all sensor reliabilities $srlb_i \in SRLB$
3: for each $D$ collected in a data collection iteration do
4: $N \leftarrow \{1,2,...,n\}$ where $n$ is the number of sensors
5: generate $K \subseteq N$ where for each $i \in K, srlb_i \geq \tau$
6: generate $D' \subseteq D$ where all sensor id present in $D'$ also in $K$
7: $v \leftarrow P_{IM}(D')$
8: update each $srlb_i$ using individual sensor readings, $v$ and Equation 3.7
9: end for
essential to build the problem that allows EM to be performed. An EM algorithm requires
the definition of two components, the statistic model $\theta$ from which the data is generated, and
the latent variable $Z$ that is independent from the statistic model. These two components will
provide a likelihood function that the algorithm is set to maximize:

$$L(\theta; X, Z) = p(X, Z|\theta)$$

A devised EM algorithm will alter between an $E-step$, which finds the latent variable
assumed a fixed model, and an $M-step$, which finds the model parameters that maximize
the expectation of the log likelihood of the model, assumed a fixed latent variable:

$$\arg\max_{\theta} E(\log L(\theta; X, Z))$$

The maximization is usually done using derivatives. Theoretically, the EM algorithm
always finds a model that has a higher likelihood expectation than the initially-set model,
however, it may converge to a local maximum, and miss the global maximum.

In our EM algorithm, the statistic model is the monitored environmental feature and
the latent variable is the faulty state of the sensors. We observed that in a small timeframe,
such as one hour, environmental features usually follow a linear model. Most environmental
features do not change drastically, and while for longer periods of time there might be more
complex models, in a small timeframe such as one hour, only some gradual changes can
be observed. Fig. 3.3 shows some examples of environmental features within one-hour
timeframes. On the top are the temperature readings of 4th March, 2004, provided by the
Intel Lab Data$^6$. On the bottom are the ozone concentration readings in 1st June, 2015,
provided by the OpenSense Project.

$^6$http://db.csail.mit.edu/labdata/labdata.html
Fig. 3.3 Environmental readings within one hour

Fig. 3.3 shows typical environmental features that follow a recognizable linear model within one-hour period, given some variance. If a sensor produces a reading once or twice per minute, then normally these 60 or 120 readings within an hour should follow a linear model. In our approach, we assume the environmental feature follows a linear model in a
timframe of length $w$, and the non-faulty readings collected over this period follow this model with normally distributed noises.

As studied by Ni et al. [11], the common sensor faults that cause various faulty data patterns include battery exhaustion, connection problem, and calibration error. Most sensor fault types are independent of the monitored environmental feature. We assume there is a latent state for each reading indicating whether the reading is faulty or non-faulty, considering that a reading produced by a faulty sensor is faulty.

### 3.4.2 Likelihood Model

Assume that in a timeframe of length $w$ the underlying environmental feature is linearly correlated to time. Let us collect a set of readings $X$ over a period $w$.

$$X = \{(x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n)\}$$

where $x_i$ is the reading value, and $t_i$ is the time of reading, and $n$ is the number of readings collected over $w$.

Let us first assume all sensors are working, and the readings have normally-distributed noises. Depending on the monitored environmental feature, the readings follow a linear model with Gaussian noises:

$$\theta = \{\alpha, \beta, \sigma^2\}$$

where $\alpha$ is the intercept, $\beta$ is the slope, and $\sigma^2$ is the standard deviation of Gaussian noises.

As defined in previous literatures [155], the likelihood of this model given a reading and its time $\{x_i, t_i\}$ is

$$L(\theta; x_i, t_i) = p(x_i, t_i | \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i - \alpha - \beta t_i)^2}{2\sigma^2}}$$
Now consider the case that there are faulty readings from malfunctioning sensors. Let us use a selection array \( Z = \{z_1, z_2, ..., z_n\} \) where

\[
    z_i = \begin{cases} 
    0 & \text{if } x_i \text{ is faulty} \\
    1 & \text{otherwise}
    \end{cases}
\]

The likelihood of a model is usually determined based on not a single observation, but a set of observations. Here we are determining the likelihood of the linear model given a set of readings as the joint likelihood given each reading. To ensure that a faulty reading does not affect the likelihood, we define the likelihood of the model as 1 regardless of the reading value, if the reading is faulty.

\[
    L(\theta; x_i, t_i, z_i) = \begin{cases} 
    L(\theta; x_i, t_i) & \text{if } z_i = 1 \\
    1 & \text{otherwise}
    \end{cases}
\]

which is equivalent to:

\[
    L(\theta; x_i, t_i, z_i) = L(\theta; x_i, t_i)^{z_i}
\]

Finally, the likelihood of the model given a set of \( n \) readings:

\[
    L(\theta; X, Z) = \prod_{i=1}^{n} L(\theta; x_i, t_i, z_i) \\
    = \prod_{i=1}^{n} L(\theta; x_i, t_i)^{z_i}
\]

### 3.4.3 Expectation Maximization for Finding Reliable Sensors

We assume that individual sensor readings are preserved, and in the collected data, each sensor reading is associated with a sensor id. If there are \( k \) sensors in the dataset, we have \( S = \{s_1, s_2, ..., s_n\} \), where each \( s_i \leq k \) is the sensor id for the reading \((x_i, t_i)\). Also we assume
that there is a background knowledge of the probability a reading can be faulty in the sensor network in general, denoted as $p(z = 1)$.

**E-step**

In the Expectation step, we are going to find the latent variable $Z$ using Bayesian’s rule. Let us first consider a single reading $r = (x, t)$. Given the model, the probability that the reading is non-faulty can be calculated as:

$$p(z = 1 | r) = \frac{p(r | z = 1) p(z = 1)}{p(r)}$$

and

$$p(r) = p(r | z = 0) p(z = 0) + p(r | z = 1) p(z = 1)$$

so

$$p(z = 1 | r) = \frac{p(r | z = 1) p(z = 1)}{p(r | z = 0) p(z = 0) + p(r | z = 1) p(z = 1)}$$

Given the model, $p(r | z = 0)$ and $p(r | z = 1)$ are essentially the likelihood function defined in Equation 3.8. Substituting in the likelihood equation will give us

$$p(z = 1 | r) = \frac{L(\theta; x, t) p(z = 1)}{1 - p(z = 1) + L(\theta; x, t) p(z = 1)}$$  (3.9)

We then calculate each $z_i = p(z_i = 1 | (x_i, t_i))$. After all $z_i$ are calculated, we use the following method to re-assign 0s and 1s to $Z$ as following: 1) group readings with the same sensor id according to $S$; 2) calculate the mean $z_s$ for all $z_i$ in each group; 3) define a confidence threshold $\delta$ so that if $z_s > \delta$, assign 1 to all $z_i$ in the group, otherwise assign 0. We choose $\delta$ as value of the $n \cdot p(z = 1)$-th element in a descendingly sorted version of $Z$. 
M-step

With the updated $Z$, find $\theta$ to maximize the expectation of the log likelihood:

$$Q = E(\log L(\theta; X, Z))$$

$$= \sum_{i=1}^{n} \log L(\theta; x_i, t_i, z_i)$$

$$= \sum_{i=1}^{n} \log \left[ \left( \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{(x_i - \alpha - \beta t_i)^2}{2\sigma^2}} \right)^{z_i} \right]$$

$$= \sum_{i=1}^{n} \left[ z_i \cdot \left( -\frac{1}{2} \log 2\pi \sigma^2 - \frac{(x_i - \alpha - \beta t_i)^2}{2\sigma^2} \right) \right]$$

Given any $\sigma^2$, $Q$ is a decreasing function of

$$g(\alpha, \beta) = \sum_{i=1}^{n} \left[ z_i \cdot (x_i - \alpha - \beta t_i)^2 \right]$$

We can find the minimal of $g(\alpha, \beta)$ through derivatives. Let us first set $\frac{\partial g}{\partial \alpha} = 0$:

$$\sum_{i=1}^{n} \left[ -2z_i \cdot (x_i - \alpha - t_i \beta) \right] = 0$$

$$\sum_{i=1}^{n} \left[ z_i \cdot (x_i - \alpha - t_i \beta) \right] = 0$$

$$\alpha = \frac{\sum_{i=1}^{n} (z_i \cdot x_i)}{\sum_{i=1}^{n} z_i} - \beta \frac{\sum_{i=1}^{n} (z_i \cdot t_i)}{\sum_{i=1}^{n} z_i}$$

Let us call $\bar{x} = \frac{\sum_{i=1}^{n} (z_i \cdot x_i)}{\sum_{i=1}^{n} z_i}$ and $\bar{t} = \frac{\sum_{i=1}^{n} (z_i \cdot t_i)}{\sum_{i=1}^{n} z_i}$ the effective means of $x$ and $t$, as they are mean values that only count when $z_i = 1$. And the above formula can be written as

$$\alpha = \bar{x} - \beta \bar{t}$$  \hspace{1cm} (3.10)
To find $\beta$, set $\frac{\partial g}{\partial \beta} = 0$:
\[
\sum_{i=1}^{n} [-2z_i \cdot t_i (x_i - \alpha - t_i \beta)] = 0
\]
\[
\sum_{i=1}^{n} [z_i \cdot t_i (x_i - \alpha - t_i \beta)] = 0
\]

Substituting $\alpha$ will give
\[
\sum_{i=1}^{n} [z_i \cdot t_i (x_i - (\bar{x} - \beta \bar{t}) - t_i \beta)] = 0
\]
\[
\sum_{i=1}^{n} [z_i \cdot t_i (x_i - \bar{x}) - z_i \cdot \beta t_i (t_i - \bar{t})] = 0
\]
\[
\beta = \frac{\sum_{i=1}^{n} z_i \cdot t_i (x_i - \bar{x})}{\sum_{i=1}^{n} z_i \cdot t_i (t_i - \bar{t})} \quad (3.11)
\]

Maximizing $Q$ as a function of $\sigma^2$ given $\alpha$ and $\beta$ will lead to:
\[
\sigma^2 = \frac{\sum_{i=1}^{n} z_i \cdot (x_i - \alpha - \beta t_i)^2}{\sum_{i=1}^{n} z_i} \quad (3.12)
\]

### 3.4.4 Final Algorithm

The EM algorithm runs in iterations, each iteration updates a new set of $Z$ and $\theta$. The algorithm terminates when $Z$ or $\theta$ converge. Algorithm 3 calculates a new set of $Z$ given $\theta$. First it calculates the reliability of each reading (line 2), then using a threshold relative to the expected number of working readings (line 5) to determine the faulty state of sensors, and reassign reliability to each reading as 0 and 1 (line 7-12). Algorithm 4 shows initialization, terminating condition and steps in each iteration of the algorithm. After initialization (line
3.5 Experimental Analysis

We validate our proposed methods on various datasets, including simulated data, real dataset with synthetic faults, and original datasets taken from real-world sensing projects. In each
experiment, we run our cleaning methods alongside with baseline methods such as mean, median and Linear Least Square Error (LLSE), and compare the results.

3.5.1 Simulated Air Pollution Sensing Data

Our first experiment focuses on monitoring air pollution in urban areas using mobile sensors. Such a scenario has been run in several projects such as OpenSense, which puts air quality sensors on trams in Zurich [40], and Common Sense, which puts air quality sensors on street sweepers in San Francisco [41]. The dataset in such projects usually contains sensor readings, as well as time and location of the sensor readings. In addition, each reading is associated with a sensor, and sensors change their locations frequently.

We first simulated a pollution map. The pollution map consists of $100 \times 100$ location points, and the corresponding pollution information at each point, as shown in Fig. 3.4a. The size of a dot on the map indicates the pollution level: the larger the dot, the higher the pollution level at the corresponding location. The maximum pollution level is 1, and the minimum pollution level is 0. We then simulated 20 mobile sensors. In each data collection iteration, a sensor made 50 readings at random locations on the map, and a total of 1,000 readings were made, similar to the readings shown in Fig. 3.4b. These are clean readings, as they are exactly the same as the ground truth pollution level at their report locations.

We used the faulty data injection method introduced in [13] to simulate sensor malfunctioning, which first establishes a set of clean data and then adds faulty data. We injected high volatility faults, similar to those shown in Fig. 3.2a, which are commonly found in air quality sensors.

In this experiment, we tested the effectiveness of the IM algorithms defined in Algorithm 1 and 2. First we set an initial reliability of 1 for all sensors. In each iteration, we generated a set of noisy readings similar to the one shown in Fig. 3.5. To divide the readings into spatially correlated groups, we divided the map into $100 \times 10 \times 10$ blocks, and the readings
whose coordinates fall within the same block were grouped. In each iteration, one prediction from each of the tested algorithms was made for each block. The ground truth pollution level of each block is calculated as the mean of pollution levels of all location points in the block. We also recorded the predictions of Mean and Median methods in each iteration. The Mean prediction is defined in Equation (3.4). The Median is defined as $P_{MED}(R) = median(R)$, where $R$ is the set of readings in one block. For IM-Remove, we chose a high value of 0.9 for $\tau$ to stress the distinction between reliable and unreliable sensors.

We measured the precision and the mean square error for the predictions made by four methods in each iteration, namely, Mean, Median, IM-Reduce, and IM-Remove, as shown in Fig. 3.6.

We noticed that the performance of the Mean and Median methods remained stable over iterations. The performance of IM-Reduce and IM-Remove, however, quickly improved in the first 10 iterations, before it became stable. The reason is that the reliability update process was picking up appropriate individual reliabilities, thus allowing the reliability-weighted method to become more accurate. After 10 iterations, IM algorithms steadily outperformed
the other two methods. In the 50 iteration run, the average precisions for Mean, Median, IM-Reduce and IM-Remove were 0.6, 0.78, 0.83 and 0.85, respectively, while the average mean square error were 0.017, 0.011, 0.007 and 0.007, respectively. In the long run, the IM algorithms can potentially have five to ten percent higher precision than the Mean and Median methods.

3.5.2 Real Data with Synthetic Faults

We tested our approach on a real dataset provided by the Intel Berkeley Research Lab, called Intel Lab data\textsuperscript{7}. The data contains environmental readings, such as humidity and temperature, reported by 52 Mica2Dot sensors. The sensors were installed in an indoor area, and had the same reporting frequency of once per 31 seconds. In our experiment, we chose a portion of temperature data in the Intel Lab data made by nine nodes with ID 1, 2, 3, 4, 6, 7, 8, 9, 10. These nine sensors were installed next to each other in a continuous open area, and can be considered as spatially correlated. Sensor node 5 is in the same area as these sensors, but for some reasons it has not produced readings, so we did not consider it in this study. We chose

\textsuperscript{7}http://db.csail.mit.edu/labdata/labdata.html
Fig. 3.6 IM methods on Simulated Air Pollution Data
a study period of roughly 75 hours, between March 2 and March 5, 2004. There were 9,000 report epochs in this period. We visually confirmed that the readings produced by the nine sensors for these epochs are clean, as shown in Fig. 3.7a.

To generate the noises, we injected faulty data into the dataset. We injected intense spikes, which can be found in other parts of the Intel Lab data. To imitate sensor condition changes over time, we injected the faults in three stages. First we grouped the sensors into three
groups: the first group contained sensors 1, 4, 8, the second group contained sensors 2, 6, 9, and the third group contained sensors 3, 7, 10. We made the second group of sensors fail in stage one and two, and the third group of sensors fail in stage two and three. So in the first fault stage, which lasted from epoch 2,250 to 4,500, readings from the sensors in the second group were injected with faulty data, as shown in Fig. 3.7b. In the second fault stage, which lasted from epoch 4,500 to 6,750, readings from the sensors in the second and third groups were injected with faulty data, as shown in Fig. 3.7c. In the third stage with remaining epochs, only readings from the sensors in the third group were injected with faulty data, as shown in Fig. 3.7d. When being injected with faulty data, each reading had a probability of 0.5 to be replaced by a spike sensor value (60 in our case).

Similar to our experiments with the simulated dataset, we ran IM cleaning with the generated noisy data, as well as the Mean and Median prediction. Since these nine sensors were considered as a single spatially correlated group, only one prediction was generated in each iteration. We recorded the predictions made by four algorithms in each iteration. For the $\tau$ threshold in IM-Remove, we chose 0.9, because we expect the sensors to function very reliably unless they are in a fault stage.

We aggregate the results by hour, and measured the mean square error for each of the algorithms in each hour. Fig. 3.8 shows predictions of the four algorithms and the ground truth over the 75-hour period, and the mean square error of the four algorithms. The beginning of the first fault stage is roughly at the 18th hour, for the second fault stage is the 37th hour, and for the third fault stage is the 56th hour.

For the EM method, we compared it with two baseline method, namely Mean and LLSE. LLSE, introduced by Sharma et al. [13], is a data cleaning approach that removes faulty data based on comparison between sensors. For each reading $x_1$, LLSE approach first produces an estimation of the reading $\hat{x}_1$ based on a comparison reading $x_2$ produced by another sensor,
Environmental Sensor Profiling for Predicting Correct Reading Values

Fig. 3.8 IM methods on Intel Lab Data

using the formula:

\[ \hat{x}_1(x_2) = m_{x_1} + \frac{\lambda_{x_1,x_2}}{\lambda_{x_2}} (x_2 - m_{x_2}) \]

where \( m_{x_1} \) and \( m_{x_2} \) are the mean readings of the sensor and the compared sensor. \( \lambda_{x_1,x_2} \) is the reading covariance between the sensor and the compared sensor. \( \lambda_{x_2} \) is the variance of the readings of the compared sensor. Then for each reading, the estimated error \( |t_1 - \hat{t}_1| \) is calculated. Finally, a confidence limit \( p \) is set, so that so that only upper \( p\% \) data is selected as non-faulty data, according to their estimated errors. The prediction of LLSE approach for the iteration is thus the mean value of the selected data. We ran the EM method and compared baseline methods on the dataset. Setting one hour as the timeframe, the readings and squared error over the iterations are shown in Fig. 3.9.

According to the results, the error the Mean approach produced increase as the portion of faulty data increase. Once the first fault stage began, Mean and IM-Reduce began to produce large errors. In the third fault stage, the performance of IM-Reduce improves from the second stage, and becomes similar to what it is in the first stage. This adjustment shows that our reliability update process not only detects sensor faults, but also captures sensors’
recovery from the faults. Median method worked very well to avoid the extreme values of the faulty data in the first and third fault stage, but not in the second fault stage, where more than half of the sensors are malfunctioning. Only IM-Remove and EM produced accurate predictions in the second fault stage. During the entire period, IM-Remove had produced predictions very close to the ground truth, expect for a small period at the beginning of the first and second fault stage, where reliabilities were being adjusted to new malfunctioning conditions. EM achieved a similar performance, producing almost no error during the entire experiment. The LLSE method was effective when the number of faulty data is small, but in the second fault stage, where the majority of the sensors were faulty, LLSE picked wrong data as the non-faulty data and thus produced large errors. The overall MSE for IM-Reduce in the experiment was 0.313, and for EM it was 0.15, while all other methods had MSE over 30.
3.5.3 Melbourne Weather Data

For the final experiments, we tested our approach on two real, unmodified datasets. The first dataset, called Melbourne Weather Data, is provided by a smart city project based in Melbourne, Australia\(^8\). The project placed a number of environmental sensors throughout Melbourne city, and has been collecting and publishing environmental data such as temperature, humidity and light levels, since December, 2014. We downloaded a portion of the data, which contains temperature readings between 20th February and 12th March, 2015, over a period of 400 hours. The readings were produced by nine sensors installed on fixed locations, which are shown in Fig. 3.10.

![Fig. 3.10 The Melbourne Weather Data Sensor Location](image)

As can be seen from the map, the sensors can be divided in two groups, according to their distances to each other. The first group, we call Group A, includes sensors 501, 502, 505, 507 and 508. The second group, we call Group B, include sensors 506, 509, 510 and 511. The sensors produce a reading once every ten minutes, and we aggregate them by hour. Fig. 3.11 shows readings from two groups of sensors.

3.5 Experimental Analysis

Since the distances between the two groups of sensors are within four kilometers, we consider all sensors are spatially correlated, with respect to the weather in Melbourne city. Examining the data reveals that the all sensors readings are very similar. The main parts the sensors differ are around the peak readings. Also readings from Group A have portions of missing data. Though it clearly indicates certain sensor malfunctioning, the missing data do not affect the calculation of our cleaning method, as a sensor that did not provide data in a data collection iteration would be simply ignored for that iteration. Dealing with missing data in our approach will be a future study topic. The available data from Group A are very similar to the data from Group B of the same time.

To establish the ground truth, we obtained weather data from WeatherSpark\textsuperscript{9}, a weather website that can provide historical data of weather stations in many cities, including Melbourne. We obtained the hourly temperature recorded by a weather station in Melbourne, over the 400-hour period in the experiment data.

\textsuperscript{9}https://weatherspark.com/about

Fig. 3.11 The Melbourne Weather Data
Then we ran the experiment in the same way as the previous experiments. We considered data in each hour as one iteration. Since the readings from all sensors are very similar, our approach was not able to distinguish the different sensor reliabilities. It was during the peak periods when sensor readings have relatively large difference, that our approach was able to produce better predictions. Fig. 3.12 shows the predictions and MSE of IM algorithms between 181st hour and 200th hour, one of the peak periods. As can be seen, around the peak, our approach produced better prediction with smaller error than the Mean and Median methods. We chose a $\tau$ value of 0.6 for IM-Remove, because all sensors in the data are relatively reliable.

![Fig. 3.12 IM methods on Melbourne Weather Dataset](image)

Setting one hour as the timeframe for one iteration, the readings and the squared errors obtained by the EM method and compared methods are shown in Fig. 3.13.

In this experiment, all sensor readings follow closely the ground truth with minimum noises, so all approaches generated small errors. The EM method was able to produce more accurate predictions, particularly during reading peaks, and generate an overall MSE of 3.13, compared to 4.83 generated by Mean and 4.67 generated by IM-Remove.
3.5 Experimental Analysis

The second dataset, called OpenSense Ozone data, is an air pollution dataset provided by the OpenSense Project\textsuperscript{10}. The OpenSense project installed air pollution sensors on trams running in Zurich, Switzerland, and the sensor readings from different locations of the city are published immediately on the website as live data. For instance, the locations of readings on 10th June, 2015, can be seen in Fig. 3.14a, where the size of the marker indicates the reading level. We collected from the website two weeks of ozone concentration data, between 10th and 23rd June, 2015. The dataset contains readings from a number of sensors, each reading contains a sensor id, a timestamp in milliseconds, a latitude and longitude pair, and the ozone concentration reading. The basic reading frequency for a sensor is once every 30 seconds, however, all sensors have missing portions in the dataset.

\textsuperscript{10}http://www.opensense.ethz.ch/trac/
Environmental Sensor Profiling for Predicting Correct Reading Values

The OpenSense project also provides data from a weather workstation as the ground truth data. The weather workstation is running in the tram station of Schimmelstrasse, which produces a ozone concentration reading and other weather data once every minute. Since the trams run across different locations of the city that can be several kilometers away from the tram station, and air pollutant such as ozone can vary drastically in different areas of the city, we first limit the testing dataset to the readings near the tram station. From the collected data, we selected a subset of data whose distances to the tram station are within 1km according to the coordinates, and use it as our testing dataset. Fig. 3.14b shows selected readings on 10th June, 2015.

Running IM-Reduce, IM-Remove and the compared methods produce results shown in Fig. 3.15. We chose a $\tau$ value of 0.4 for IM-Remove, because we expect the sensor quality to be poor.

Setting one hour as the timeframe for one iteration, running the EM method and compared methods on the dataset generated results shown in Fig. 3.16.
In this experiment, the testing data overall deviates significantly from the ground truth. For instance, the maximum reading in ground truth data is 73.22, but the maximum reading in the testing data is only 52.87. Because of the overall low quality of the testing data, all approaches produced large errors. However, as can be seen from the figure, the prediction by
EM approach was closer to the ground truth than the other approaches. The errors produced by EM were also smaller than the errors of the other approaches throughout the experiment. IM-Remove was also able to produce more accuracy predictions than Mean and IM-Reduce. The overall MSE for EM method in this dataset was 307, for IM-Remove it was 427, for Mean and IM-Reduce they were 476 and 461, respectively.

### 3.5.5 Discussion

Table. 3.1 compares overall MSE of three approaches across three experiments with real datasets. We can see that, while our EM approach achieved better results in all three experiments, it performed particularly well in the first experiment with synthetic faults. This is because in the first experiment, working sensors and faulty sensors differ significantly in their readings, and correctly picking the working sensors would have critical impact on the result. In the second experiment, the differences between the working sensors and faulty sensors were not significant, thus the impact was smaller for choosing the correct working sensors. We can also see that IM-Remove achieved better accuracy than IM-Reduce in all experiments, indicating that removing unreliable sources generally provide more accurate predictions than just reducing their weight in aggregation.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mean</th>
<th>Median</th>
<th>IM-Reduce</th>
<th>IM-Remove</th>
<th>LLSE</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Lab Data</td>
<td>62.93</td>
<td>41.29</td>
<td>39.49</td>
<td>0.31</td>
<td>37.90</td>
<td>0.15</td>
</tr>
<tr>
<td>Melbourne Weather</td>
<td>4.83</td>
<td>4.76</td>
<td>4.69</td>
<td>4.67</td>
<td>3.51</td>
<td>3.13</td>
</tr>
<tr>
<td>OpenSense Ozone</td>
<td>476</td>
<td>473</td>
<td>461</td>
<td>427</td>
<td>477</td>
<td>307</td>
</tr>
</tbody>
</table>

### 3.6 Summary

Environmental sensing has become a significant way for people to understand their living environments. The Internet-of-Things (IoT) inspired sensors provide frequent and detailed sensor
readings that allows profiling of individual sensor reliability, which has been largely overlooked in previous literature. Based on individual sensor reliability profiling, we approach the correct reading value prediction problem from frequentist and Bayesian perspectives. We propose and compare three prediction algorithms, namely, IM-Reduce, IM-Remove, and EM. We test the algorithms extensively using synthetic and real datasets. The experimental results show that, the proposed approaches were able to produce predictions much more accurate than the existing approaches. In the case where all sensors were normally working and produced similar readings, our approaches were able to improve the accuracy by about 10%. In the case where reliable sensor and faulty sensors differed significantly, our approaches were particularly effective, with the IM-Remove algorithm generated a MSE of 0.31, and the EM generated a MSE of 0.15, while for all baseline approaches it was over 35. We also show that our approaches can adapt to sensor reliability changes, and can successfully capture sensor faults and the recovery from faulty state.
Chapter 4

User Profiling for Classifying Observation Tweets

In this chapter, we will move from environmental sensing to social sensing. We will tackle the first problem of observation classification. Microblogging services such as Twitter allows users to conveniently publish show messages about their thought, interests, and observations. Observations about certain events or objects of interest, when collected, can be seen as some sensing values provided by human sensors. In this Chapter, we describe our work on classifying observation messages in microblogging services. The main challenge for observation classification is the difficulty to accurately classify these messages, and current works have not achieved adequate accuracy. We approach the problem from three novel viewpoints. First we consider the perspectives a messages can be composed of, namely, observation, affection, and speculation. Then we incorporate users features into supervised learning classification methods, which are features largely overlooked by existing works. Then we propose an unsupervised approach that combines lexical message analysis and user profiling for identifying personal observations. We run separate experiments to test all three approaches with real Twitter data, and show that each approach is able to consistently improve classification accuracy.
4.1 Overview

Microblogging services, which have gained high popularity in recent years, offer a convenient way for individuals to create and share information online. An example is Twitter, which allows its users to create and publish short messages with a maximum of 140 characters, called *tweets*. Currently, around 284 million active Twitter users generate 500 million tweets every day, and around 80% of Twitter users use their mobile phones to create tweets\(^1\). There is no cost associated with publishing tweets, and the messages entered by a user are immediately available for anyone to read online. The use of mobile platforms for tweeting implies that users can easily report observed events and objects in their physical vicinity. Direct observations of newsworthy events are an invaluable source of news, even if the reporter is not a professional journalist. This has seen Twitter being regarded as an emerging news source by recent work \(52, 156\), and users who post event-related messages on Twitter are sometimes termed *citizen reporters* \(157, 64\).

Twitter has been analyzed as a reporting tool for the 2009 Iran Election Crisis \(65\), the 2011 Arab Spring movement \(66\), and recently the 2014 Ukraine Crisis\(^2\), during which numerous users have made firsthand observations about ongoing events. When the protesters gathered around city hall in Kiev in early February, and when the clashes erupted between the police and protestor on 18th February, a large number of firsthand observations were published on Twitter by people from the crowd, describing what was happening. Later events involving military actions around the boarder of Ukraine also generated numerous tweets made by people who were experiencing them firsthand, expressing their observations, emotional reactions, and speculations. These messages provided invaluable information at a time when no credible news source had published related stories\(^3\).

\(^{1}\)https://about.twitter.com/company
4.1 Overview

Personal observations on Twitter and their implicitly-associated time and location data can also be an invaluable information source for monitoring objects and events. Beyond news reporting, existing work [14, 53] has found that the objects reported by Twitter users can lead to useful information, with critical applications such as natural disaster location and crime monitoring. For example, tweets about earthquakes and typhoons have been collected and used for tracking the movements of these natural phenomenon in Japan [53]; tweets about flood, hurricane, riots, and shootings have also been used for disaster and crime monitoring [14, 10]. Urban areas around the world, particularly in the United States, Europe, and Japan, see dense distributions of Twitter users [158], which can be seen as a sensor network that has a large and dense monitoring coverage in these areas.

Observation messages posted by Twitter users, however, are different from typical sensor reading values. They are usually composed in informal natural languages that do not have a pre-defined format. Moreover, a user who posts observations about environments may also post a wide range of other messages, including personal interests, discussions of news, and business-related conversations, which cannot be counted as observations [159]. And given an object or event expressed by a keyword, the messages containing such a keyword often do not refer to the object or event observed by the user. For instance, we collected messages containing the keywords “hailstorm” and “car accident”, and the ratio of the messages that contain an actual observation among all messages containing the keyword can be as low as 37%. Distinguishing observation messages from other tweets thus requires certain means of classification. Currently, the most common approach to classify tweets is using supervised machine learning techniques [100, 160, 53, 16], while some works also propose unsupervised solutions [17, 96].

A supervised machine learning approach involves devising a number of features that can be extracted from the tweet data, labelling a number of training examples, and applying machine learning techniques to generate a classifier. The accuracy of the classifier heavily
depends on the chosen features, and the quality of training examples. One interesting insight shown in existing literature is that one particular effective feature provides better classification results than a combination of multiple features. The works in [100, 160, 53] all show that, given a number of feature sets, the classification accuracy is higher when using an individual feature set, than when using a combination of them. For example, in [53], an individual feature set achieved a precision of 68.57%, while the combination of three feature sets only achieved 65.91%. Therefore finding a particularly effective feature in supervised methods is very crucial.

We first approach the classification problem using supervised methods. One type of features currently overlooked is the set of user-related features. Twitter provides public access to user profiles and tweet histories, which can be used to generate various user features. While commonly used in credibility analysis and rumor detection [100, 160], user features have not been tested for observation classifications. We focus on user-based feature design, and propose two solutions. The first solution distinguishes three user perspectives that may be used when composing a message, namely observation, affection, or speculation. We then turn our focus to a particular perspective, namely, the observation. The second solution focuses on filtering observation messages using statistical user-based features. In Section 4.2 and 4.4 we will present these two solutions, respectively. In Section 4.3 and 4.5, we will present experiments for testing the feasibility and accuracy of these two solutions using real Twitter data.

We then approach the problem using an unsupervised method. Supervised machine learning methods, which require the manual preparation of training samples, have several drawbacks. First, the manual annotation of text message is a labor-intensive and expensive task. Second, annotated examples usually are not compatible when analysis is done for different purposes, and research projects for short texts analysis usually do not share annotations. This implies that a supervised learning method used, for example, to infer information from
a dataset collected during the Ukraine crisis, might not be used in the analysis of dataset referring to Syrian refugees. Third, the accuracy of the supervised methods depends on the quality of the manual annotations, which is difficult to guarantee. *Unsupervised methods* have the potential to address these issues.

Unsupervised methods are commonly used in data analysis in domains such as sensor networks [79, 90]. In the domain of microblogs and short texts, however, due to the complexity and uncertainty of human user data, work on unsupervised method is still at an early stage [102, 161]. The informal and unstructured text messages used on microblogs creates uncertainty for any kind of classification models, and solutions and models effective in one application often will not be as effective in another application. For example, the work in [162] has shown the effect of user roles in Twitter rumor classification varies significantly for different rumor instances. A challenge for a specific application using unsupervised methods therefore would be to find a particular model that is effective for that application.

To improve the accuracy of our unsupervised approach and to address several of the challenges discussed above, such as high dimensionality and uncertainty, we advance a novel approach that employs lexical analysis and user profiling. The lexical analysis module filters *observation messages* based on two attributes, part-of-speech (POS) tag and message objectivity. The user profiling separates *personal accounts* from other types of accounts based on four analyzed attributes, namely, *objectivity, interactivity, originality, and topic focus*. We will present this approach in detail in Section 4.6. In Section 4.7 we will present the classification accuracy of this approach tested using real Twitter data.

### 4.2 Related Work

Twitter as a public media and news source has been studied in several works. Wu et al. [159] investigated the demographics of influential Twitter users, whom they grouped into media, organization, celebrity, and blogger. Their study concludes that bloggers are popular
personal accounts that produce the most influential tweets. Vis [156] studied Twitter as a reporting tool during the 2011 UK riots. Vis identified several actor types describing users who were involved in event reporting, including journalists, activists, mainstream media, police, and fake accounts. Metaxas et al. [64] studied tweets reporting criminal activities in Mexico in 2010 and 2011. They used peaks in tweet posting for detecting events, while users involved are grouped into different actor roles. These studies are based on the roles of reporting users, which we consider inappropriate, because an ordinary user does not need to have a consistent role when tweeting about an event. In contrast, while not proposing an automated classification method, Maddock et al. [162] studied misinformation, speculation, question, and other categories of information reported on Twitter after an important event, manually labelling the category of each tweet. Using rumors circulated after the Boston Marathon Bombing in 2013, they show how each type of message evolves in the propagation of rumors, and that users in certain roles behave differently in different rumor instances. Kwon et al. [160] investigated supervised methods for identifying rumors on Twitter using a number of prominent features, including propagation peaks, friendship network graph, and linguistic properties based on LIWC (Linguistic Inquiry and Word Count). They found that selecting the right features is critical to classifier accuracy. Sriram et al. [16] similarly studied supervised tweet classification for five types of tweets: news, opinions, deals, events, and private messages. Although various machine learning techniques are compared, they also found that selecting the right features is the most influential factor for classification accuracy.

Despite the fact that messages on Twitter are very often informal and incomplete [157], researches have used Twitter information for election prediction [163], disaster location [10], object tracking [9], and event detection [96]. Tumasjan et al. [163] found that the sentiments users express on Twitter correspond closely to actual election results, based on Germany federal election in 2009. Sang et al. [106] conducted a content analysis and also found that sentiments expressed on Twitter correspond closely to election results, based
4.2 Related Work

on Dutch Senate election in 2011. Their sentiment analysis is based on LWIC categories [164]. Unankard et al. [96] deployed sentiment analysis and clustering techniques to predict Australian federal election in 2013 based on Twitter messages, and their prediction results matches real election results closely. Lingad et al. [14] studied locations mentioned in disaster-related messages in order to identify the position of natural disasters and affected areas. Their method establishes a link between the location mentioned in tweet messages and the real locations. Lin et al. [125] proposed an event-based point-of-interest (POI) detection system based on Twitter message. They construct clusters of messages based on location profiles and detects spatiotemporal hot spots. Kwon et al. [160] identifies several signatures that can be used to distinguish false rumors among stories circulating on Twitter. By manually identifying the truthfulness of the stories, they found that the machine learning-based automatic classification method can determine rumors with relatively high accuracy. These works showed a range of possibilities that the content of reports on Twitter can be used, and more importantly, they show that discussions on Twitter reflect very well real-world phenomenon.

Identifying a particular type of messages in Twitter data requires certain classification technique, and given that tweet messages are usually informal, unorganized texts, it is difficult to apply traditional model-based natural language processing techniques. Currently, the most used method for tweet content analysis is machine-learning-based classification. Such a method involves devising a set of features that can be extracted from tweet messages, and using the extracted feature strings as the input to certain machine learning models. For example, the earthquake monitoring system proposed by Sakaki et al. [53] classified tweets into positive and negative reports, based on a set of relatively simple features, including the word count and the position of the query word. Li et al. [10] used location words, URLs and reply signs in tweet messages and an automatic classifier model in their event detection system. Castillo et al. [100] studied information credibility on Twitter. To identify
newsworthy messages and their credibilities, they included text, topic, and propagation features in automatic classifiers. Sriram et al. [16] provided a solution to tweet classification for the purpose of understanding user intention. They classified five types of tweets: news, opinions, deals, events, and private messages, based on features such as the presence of time-event phrase, opinion words, and dollar signs. Gupta [10] developed an online system for identifying tweet credibilities, based on a SVM classifier and a number of features including character and word count, and the linguistic category of each word.

However, accurately classifying object or event-related messages is a challenging task. Sakaki et al. [9], who developed the system that tracks the movement of earthquakes and typhoons based on reports detected on Twitter, compared a number of features for building the observation message classifier, and the best-performing feature set achieving a precision of 63.64%. Li et al. [10] trained a classifier based on the words present in identified CDE tweets, and achieved a precision of 30% and a recall of 85%. To improve the classification accuracy, some studies employ external information sources. For example, Unankard et al. [161] developed an event detection system, based on location correlations between the users and the event. They use an online Gazetteer service for location identification. Lucia et al. [17] proposed a classification technique that expands short messages based on name-entity identification and external knowledge bases. The quality of classifiers that rely on external sources, however, depends on the quality of the external source employed. Moreover, the methods are not applicable in cases where the external source is not available.

Unsupervised methods have the advantage of needing less manual effort. Current unsupervised methods for microblog analysis are still at an early stage of research. Carroll et al. [102] developed an unsupervised method for determining the objectivity of in Chinese microblog texts. They defined objectivity as sentiment neutrality, but the application is limited to brand and company reputation analysis. Unankard et al. [96] developed a framework for predicting election results using Twitter messages, in which message and user sentiments are calculated

4http://www.geonames.org
based on positive and negative word counts. Since observation messages are not strongly related to message sentiments, filtering observations, however, requires technique beyond sentiment analysis.

4.3 Message Perspective Classification

Given a large number of Twitter posts, it is difficult for anyone to read through the tweets manually, identify these perspectives, and further analyze the data. In our experiments, we collected tweets related to the 2014 Ukraine Crisis over a period of 56 days, and the dataset contains over 950,000 tweets. An automated classification method that helps identify the perspectives of the tweets is crucial. In this part of work, we aim to find a supervised learning model that can accurately classify event-related tweets according to three perspectives they are posted in, namely, *Immediate Observation*, *Affection*, and *Speculation*. To identify the most suitable learning techniques, we select a number of lexical and textual features that can be obtained from tweet messages, and test them on various model building techniques, including SVM, Naive Bayes, and Random Forest. An overview of our approach is shown in Fig. 4.1

4.3.1 Message Perspectives

Analyzing an event as it is reported on Twitter by a variety of users can reveal a range of information, from the location of the event, the number of people present and the actions they are taking, to the predicted outcome of the event [10, 9, 163]. However, the large variety of tweet messages can be distracting and distorting for the analysis. For example, Maddock et al. showed that, a controversial story may invoke hedges, questions, speculations, misinformation, corrections and many other unrelated messages on Twitter [162]. Therefore it is invaluable if the perspectives of the tweet messages can be first provided. We distinguish
Fig. 4.1 Supervised Message Perspective Classification

between three perspectives that a Twitter user might employ when composing messages, namely, Immediate Observation, Affection, and Speculation. Table 4.1 shows an example of each of the perspectives.

*Immediate Observations* are observations made immediately at the scene of the event. Through Immediate Observations, we can gain direct observation of events that are not easily accessible. In addition, since the messages are associated with time and location data, we can also infer the time and location of the event from the information indirectly provided by the user. For example, when we collect a tweet post in Times Square on Tuesday morning, saying “the air is fresh”, which is associated with a timestamp and GPS data, we obtain the air quality information in Times Square on Tuesday morning, even if the message itself does not provide this information.

*Affection tweets* are messages about how an event impacts users emotionally. Sometimes it is important to know the public sentiments toward an event for government decisions and policymaking, particularly when the event is controversial and the public sentiments are diverse. Twitter users express opinion about events in many ways, and some users may express opinions about future events or events they are not directly involving in. Aggregating
sentiments with these opinions may make the results noisy and inaccurate. By limiting the scope to the affections of the users who are directly involving in and impacted by the event using automated classification, we can obtain a much more accurate understanding of how the event actually affected users.

*Speculation tweets* are users’ attempt to guess or predict the nature and future of uncertain events. A key characteristic of speculations is that the information they provided has not been confirmed as true or otherwise. When an important event happens and many facts are unclear, it is guaranteed that it will invoke wide speculations. While messages themselves have little value for understanding the event, the speculations that contain false information may invoke public panic when widely circulated. Therefore tracking speculations may help one to identify which false stories are gaining popularity and be alarmed.
4.3.2 Identifying Lexical Features

Each tweet message consists of up to 140 characters. Since our lexical analysis focuses on words, we break each tweet message up by spaces and punctuations into a series of words. For example, the tweet “Yesterday, 32 monuments of Lenin were toppled around Ukraine” is tokenized into “Yesterday”, “32”, “monuments”, “of”, “Lenin”, “were”, “toppled”, “around”, and “Ukraine”. These words are the information unites in our approach, which basically involves counting words in different categories. As an example, in this tweet message, there is one word related to time, “Yesterday”, and one word related to space, “around”.

We use the categories defined in *Linguistic Inquiry and Word Count* (LIWC) for lexical analysis. LIWC provides a dictionary for different categories of English words, such as functional words, common verbs, and emotional words, and is one of the most used tools in natural language processing [164]. The dictionary contains four main categories and 64 sub-categories, and the number of entries exceeds 10,000. The entries in different categories may overlap.

We use L1 to L64 as the code name the 64 LIWC categories. Not all categories of words are helpful for distinguishing tweets of a certain perspective from other tweets. For example, emotional words are clear signatures of the affection perspective, but they are less meaningful for identifying observation and speculation tweets. Therefore we first tested the presence of each of the 64 sub-categories defined in LIWC 2007 using the following method. Given a set of tweets $T$ whose perspectives we already knew, for each perspective, we calculated the effect of each lexical category $e_l$ as $e_l = |p_{l,1} - p_{l,0}|/|T|$, where $p_{l,1}$ is the count of words in the lexical category $l$ in all tweets in that perspective, $p_{l,0}$ is the count of words in the lexical category $l$ in all tweets not in that perspective, and $|T|$ is the total number of tweets. Once we have the effect of each lexical category $\{e_1, e_2, e_3, ..., e_{64}\}$, we take the 10 most effective lexical categories for each perspective. Table 4.2 shows the categories we obtained. Table
4.3 Message Perspective Classification

4.3 shows the category names defined in LIWC 2007 and some example dictionary entries of these categories.

Table 4.2 Effective lexical categories for different perspectives

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Category Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate Observation</td>
<td>L1, L2, L4, L9, L22, L23, L27, L29, L33, L51</td>
</tr>
<tr>
<td>Affection</td>
<td>L1, L2, L10, L11, L13, L22, L23, L27, L28, L51</td>
</tr>
<tr>
<td>Speculation</td>
<td>L1, L10, L11, L13, L16, L22, L28, L31, L33, L53</td>
</tr>
</tbody>
</table>

Table 4.3 Selected lexical categories and example words

<table>
<thead>
<tr>
<th>Code</th>
<th>Lexical Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>Function Words</td>
<td>about, back, can, despite, each</td>
</tr>
<tr>
<td>L2</td>
<td>Personal Pron</td>
<td>he, she, I, we, ours</td>
</tr>
<tr>
<td>L4</td>
<td>1st Plural</td>
<td>lets, our, ourselves, us, we</td>
</tr>
<tr>
<td>L10</td>
<td>Common Verbs</td>
<td>accept, become, call, depends, eaten</td>
</tr>
<tr>
<td>L11</td>
<td>Auxiliary Verbs</td>
<td>are, can, do, gonna, had</td>
</tr>
<tr>
<td>L13</td>
<td>Present Tense</td>
<td>ask, asks, become, becomes</td>
</tr>
<tr>
<td>L16</td>
<td>Prepositions</td>
<td>about, before, down, except, for</td>
</tr>
<tr>
<td>L22</td>
<td>Total Pron</td>
<td>anybody, he, I, ourselves, that</td>
</tr>
<tr>
<td>L23</td>
<td>Social Process</td>
<td>admit, baby, captain, deal, email</td>
</tr>
<tr>
<td>L27</td>
<td>Affective Processes</td>
<td>abuse, bad, care, danger, eager</td>
</tr>
<tr>
<td>L28</td>
<td>Positive Emotion</td>
<td>accept, beauty, careful, daring, easy</td>
</tr>
<tr>
<td>L29</td>
<td>Negative Emotion</td>
<td>abuse, bad, careless, danger, enemy</td>
</tr>
<tr>
<td>L31</td>
<td>Anger</td>
<td>abuse, bastard, cheat, destroy, enrage</td>
</tr>
<tr>
<td>L33</td>
<td>Cognitive Processes</td>
<td>absolute, ban, cause, deduce, effect</td>
</tr>
<tr>
<td>L51</td>
<td>Relativity</td>
<td>above, before, carry, dance, early</td>
</tr>
<tr>
<td>L53</td>
<td>Space</td>
<td>air, bend, capacity, deep, east</td>
</tr>
</tbody>
</table>
4.3.3 Identifying Textual Features

In addition to lexical categories, we choose five textual features that are expressive and may reveal the perspective of a tweet. These features include word count, emphasis count, exclaim mark count, question mark count, and tag presence. Table 4.4 gives brief explanations for these features. The textual features are analyzed for all three perspectives.

Table 4.4 Textual features for perspective classification

<table>
<thead>
<tr>
<th>Code</th>
<th>Textual Features</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Word Count</td>
<td>the number of words in the tweet message</td>
</tr>
<tr>
<td>T2</td>
<td>Emphasis</td>
<td>the number of words in capital letters, e.g., GREAT</td>
</tr>
<tr>
<td>T3</td>
<td>Exclaim Mark</td>
<td>the number of exclaim marks</td>
</tr>
<tr>
<td>T4</td>
<td>Question Mark</td>
<td>the number of question marks</td>
</tr>
<tr>
<td>T5</td>
<td>Hashtag Presence</td>
<td>whether a hashtag is present, 1=present, 0=otherwise</td>
</tr>
</tbody>
</table>

4.3.4 Machine Learning Classifiers

Automatic learning models have been widely used in various classification problems. A supervised learning model is generated from a set of labelled data, called training set, and then applied to unlabelled data for predicting their labels, or classes. Various techniques have been established for building the supervised learning model, such as Support Vector Machine (SVM), Naive Bayes, Random Forrest, and Linear Discriminant Analysis (LDA), and have been frequently used in existing works for tweet classification [100, 9].

A SVM classifier finds a hyperplane that separates data in different classes with a maximum margin [165]. It usually maps data into a higher-dimension space to improve efficiency, using different kernel functions, such as linear and polynomial. A Naive Bayes classifier is trained using Bayes Theorem, and is based on the assumption that features in the dataset are strongly independent [166]. A Random Forest classifier is built from multiple
decision trees, which are sets of rules that separate data in different classes generated from the training data [167]. A LDA classifier finds a linear combination of features that separates data in different classes [168]. In our experiments, we trained classifiers using all four techniques and compared their classification accuracies. For the SVM classifier, we tested both the linear and polynomial kernels. We deployed existing implementations provided as R packages, including the e1071 package\(^5\) for SVM, Naive Bayes and Random Forest, and the MASS package\(^6\) for LDA.

For each perspective, we supply a set of tweets as labelled either positive or negative. Positive for a perspective means that the tweet is composed in that perspective, and negative means the tweet is not composed in that perspective. A separate classifier is built for each perspective using the respective training data. The tweet messages are transformed into features using the method mentioned in previous sections, before processed by the classifiers. The training data for each perspective consists of 15 features, the word count in ten most effective lexical categories for that perspective, and five textual features shared by all perspectives.

As an example, Table 4.5 shows the feature string for the tweet “Shell hits school in #Donetsk, eastern #Ukraine, killing two children amid renewed violence”. Since we are interested in whether this message is an immediate observation, we select the Immediate Observation feature set shown in Table 4.2. The textual feature set shown in Table 4.4 is also used.

---

\(^5\)https://cran.r-project.org/web/packages/e1071/

\(^6\)https://cran.r-project.org/web/packages/MASS/
4.4 Supervised Observation Classification with User Features

We now turn to classification of a particular perspective, the observation. The observation messages can be seen as sensing data from human sensors with implicitly associated time and location information. In the previous part of work, we show that how lexical and textual features can be used to effectively classify observation tweets. In this part of work, we will test user features, which are largely overlooked in existing message classification literatures. Since Twitter offer the access to a wide range of user information, we can generate many user features, such as follower count, registration age, used in other works [100]. However, as existing works find that a small number of particularly effective features provide better results than the combination of a large number of features, instead of generating and testing every possible feature, we focus on finding small sets of particularly effective features. We design three sets of features that reflect users’ tendency to post observation messages, namely *trending activity* features, *communication status* features, and *writing role* features. We test the features with various machine learning models, including SVM, LDA, Naive Bayes, and Random Forest. Using real tweet data, our experiments show that employing a user feature set consistently improves the classification accuracy over the existing approaches.

4.4.1 Generating User Features

We aim to find a set of effective user features that can significantly improve the accuracy of observation classification. Twitter provides various user data through its search API\(^7\), including user name, follower and following count, and a number of past tweets, which can be used to generate a wide range of features about the user. With regard to observation classification, we focus on the features that reveal whether a user is more likely to post an

\(^7\)https://dev.twitter.com/rest/public/search
observation message. Particularly, we focus on the trending activity, the communication status, and the writing role of the user. Table 4.6 shows feature sets and a brief explanation of each feature we selected.

Table 4.6 User Feature Sets

<table>
<thead>
<tr>
<th>Feature Set A: trending activity</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>hashtag rate</td>
<td>ratio of messages containing “#” in the user’s past messages</td>
</tr>
<tr>
<td>retweet rate</td>
<td>ratio of messages containing “RT” in the user’s past messages</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature Set B: communication status</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>follower count</td>
<td>number of followers as returned by Twitter API</td>
</tr>
<tr>
<td>mention rate</td>
<td>ratio of messages containing “@” in the user’s past messages</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature Set C: writing role</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>whether the user description contains role identifier such as news, company</td>
</tr>
<tr>
<td>topic focus</td>
<td>the degree the user’s messages focus on a small number of topics</td>
</tr>
<tr>
<td>original rate</td>
<td>ratio of original messages in the user’s past messages</td>
</tr>
</tbody>
</table>

4.4.2 Trending Activity Features

Twitter allows messages to identify themselves as belonging to a popular topic, through the use of hashtag, a topic keyword proceeded by “#”, in the messages. For example, most messages related to the US 2016 Presidential Election has the hashtag “#Election2016”. A user on Twitter may contribute to a trending topic by including the hashtag in their messages. A user may also retweet a popular message by repeating the message and adding “RT”. Hashtags and retweets contribute to the emergence of trending topics, which popular topics that have wide spreads on Twitter. We consider that, if a user employs Twitter more for participating in trending topics, they are less likely to post messages about their immediate environment. Therefore we include the hashtag rate and the retweet rate as the trending activity feature set.
4.4.3 Communication Status Features

A user on Twitter may communicate with other users by following and being followed. If a user follows another user, the messages of the followed user will be displayed on the Twitter page of the following user. Following another user, therefore, does not establish communication, as the following user only passively receives the message of the followed user. Any message a user posts, however, does establish a communication between the user and their followers, as it is known to the user that the message will be seen by their followers. Therefore we include the number of followers in the communication status feature set. Twitter also allows explicit communication between users via the mention sign “@”, followed by the username. If a user is mentioned in a message, they will see the message regardless of follower relationships. So we include mention rate as one of the communication status features. Communication status can be helpful for identifying observation messages because a user who focus too much or too little on communication may not be interesting in provide observations of their personal environments.

4.4.4 Writing Style Features

In addition to personal users, Twitter also supports business accounts, such as news outlets and company accounts [159]. These accounts usually only post business-related messages, and will rarely post observations about immediate environments. However, Twitter generally does not provide an identifier that distinguishes business and personal users, and such distinction only becomes clear when the account description and posted messages are manually examined. We propose three features that may help identify business accounts. The first feature checks if the user description contains identifiers such as “news”, “media”, “company”, or “ngo” (non-governmental organization). Table 4.7 lists the identifier we check.
Table 4.7 Business Account Identifier

<table>
<thead>
<tr>
<th>News</th>
<th>news media blog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>company organization organisation corporation brand product</td>
</tr>
<tr>
<td>NGO</td>
<td>ngo charity cause</td>
</tr>
</tbody>
</table>

We measure the *topic focus* based on the user’s past messages, as business accounts are more likely to focus on a narrow range of topics. To calculate a user’s topic focus, we count the frequency of each topic word for all topic words appearing in past messages. For simplicity, we consider a topic word as a word that starts with a capital letter; the first word in a sentence is ignored. Once we have a descendingly-sorted list of topic word occurrences \( \{nt_1, nt_2, \ldots, nt_k\} \), the topic focus of a user is calculated based on the fraction of the first quarter of the most frequent topic words:

\[
ufocus = \frac{\sum_{i=1}^{n/4} nt_i}{\sum_{j=1}^{n} nt_j}
\]

Although using first half or third may achieve similar effect, we find that using first quarter of topic words generally represent well the focused topics of the user. Then we consider the *originality* of the user, based on the ratio of original messages in the user’s past messages. To determine if a message is an original message, we apply the rules describe in Table 4.8. A message that satisfies any of the rules in the table is considered non-original.

Table 4.8 Originality Test Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>retweet</td>
<td>contains the word RT</td>
</tr>
<tr>
<td>quotation</td>
<td>contains quotation marks</td>
</tr>
<tr>
<td>speech</td>
<td>mention or capitalized word before colon</td>
</tr>
<tr>
<td>news title</td>
<td>all words capitalized before link</td>
</tr>
<tr>
<td>repeat</td>
<td>contains “says”, “claims”, “via”, or “according to”</td>
</tr>
<tr>
<td>news mention</td>
<td>mention contains “news”, “radio”, or “breaking”</td>
</tr>
<tr>
<td>news agent</td>
<td>mention contains news agent name such as “ABC” or “CNN”</td>
</tr>
</tbody>
</table>


4.5 Unsupervised Observation Classification

In addition to supervised observation classification methods, we also developed an unsupervised observation classification method. Our method filters observations of objects and events from personal accounts, by performing the following steps. First, we identify observations from collected tweets for a specific keyword. Second, using also the same collected tweets, we distinguish personal accounts from other types of accounts. A personal account is a Twitter account employed for personal use, and is assumed to be free from business or propaganda interests. Our insight is that tweets from personal accounts often contain realtime and localized observations of objects and events. Finally, from the observation tweets identified in the first step, we retain only those made from personal accounts.

An overview of our method is shown in Fig. 4.2. To identify observation tweets, we run lexical analysis on tweet texts based on the par-of-speech (POS) tagging, objectivity analysis, and originality test. To identify personal accounts, we first analyze four attributes for each user, namely, objectivity, interactivity, originality, and topic focus. Then we use a clustering algorithm for classifying personal accounts based on the attribute values. In this section, we will describe our method in detail.

4.5.1 Observation Filtering

After using the Twitter Filter API\(^8\) to obtain tweets that contain the object or event keyword such as “rainbow” or “car accident”, our lexical analysis method focuses on extracting observation tweets. Not all tweets containing the keywords of interest are observations of objects and events, since in some cases the keywords can have another semantic, context-based meaning, and the objects and events can be mentioned in general comments instead of specific observations, e.g., “I dislike car accidents”. We address this issue by utilizing three techniques, namely, par-of-speech (POS) tagging, objectivity analysis, and originality test.

\(^8\)https://dev.twitter.com/streaming/public
POS tagging allows filtering of messages in which the object or event keyword is not used as a subject of observation. Objectivity analysis allows filtering of uncertain messages, such as questions and general comments. Originality test filters out those messages that are not originally created by the user, such as retweets or quotations.

**Filtering Based on Part-Of-Speech Tagging**

Our insight is that the objects and events mentioned in an observation are most likely to be nouns and gerunds, such as in “I just saw a rainbow”, or “A shooting outside my home”. On the other hand, keywords not used as nouns and gerunds often indicate that the tweet is not a
Table 4.9 Non-observation tweets filtered by POS tagging, for monitoring flight delay, shooting incidents, and rainbows

<table>
<thead>
<tr>
<th>Tweet Text</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep praying for the typhoon to magically delay my flight a day</td>
<td>VB</td>
</tr>
<tr>
<td>Can we pretend that airplanes, in the night sky, are like shooting stars?</td>
<td>JJ</td>
</tr>
<tr>
<td>This guy got on a rainbow colored LV belt</td>
<td>JJ</td>
</tr>
</tbody>
</table>

VB=base form verb, JJ=adjective

specific observation. Some examples of non-observation tweets are shown in Table 4.9, with the role of the keyword determined by POS tagging.

POS tagging is a technique that matches words in a text with their part-of-speech categories, such as modal, noun, verb, and adverb [169]. We use a filtering rule on top of POS tagging to effectively remove a portion of tweets that are obviously not observations. After performing POS tagging for a tweet, we accept it if the POS tag for the keyword is NN (Noun, singular or mass), NNP (proper noun, singular), and VBG (verb, gerund or present particle). The tweet is rejected if the keyword has other POS tags.

Filtering Based on Objectivity Analysis

Our insight is that a specific observation of an object or event usually is written in a more objective tone than a general tweet. Generally, the objectivity of a message is affected by sentimental words and uncertain words, such as “great”, “bad”, “maybe”, “anyone”. Sentimentality and uncertainty as factors for determining message objectivity has already been proposed in existing works [170, 102]. In our approach, we calculate tweet objectivity based on both sentimentality and uncertainty, using the following formula:

\[
objectivity(t) = 1 - [senti_p(t) + 0.5 \times senti_n(t)] \\
\times (1 - \sqrt{uncertainty(t)})
\]
where $senti_p$ is the positive sentiment and $senti_n$ is the negative sentiment. We have found that negative sentiments have a large presence in observation messages, so following this insight here, we weight down the effect of negative sentiments on reducing the objectivity in the formula. Furthermore, since uncertainty plays an important role in determining the objectivity of a message, as discovered in [170], we increase the effect of uncertainty by scaling it to a larger value.

For sentiment analysis, we employ previously proven effective methods, which employ a positive/negative words dictionary and the slang sentiment dictionary [96]. The positive and negative sentiments of a tweet text $t$ are measured as:

$$senti_p(t) = \frac{count_p(t)}{count_w(t)}$$

$$senti_n(t) = \frac{count_n(t)}{count_w(t)}$$

where $count_p(t)$ and $count_n(t)$ are the word count for positive and negative words in $t$, and $count_w(t)$ is the word count of $t$.

For uncertainty analysis, we use a dictionary of uncertain words based on the LIWC category of hesitation words [164]. To measure the uncertainty of tweet text $t$, we consider the number of uncertain words in the text, and whether the text is a question. Based on our perception, a question generally possess the same level of uncertainty as a sentence in which half of the words are uncertain words.

$$uncertainty(t) = \begin{cases} 
0.5, & \text{if } t \text{ ends with a question mark} \\
\frac{count_u(t)}{count_w(t)}, & \text{otherwise}
\end{cases}$$

where $count_u(t)$ is the word count for uncertain words in $t$. 

Originality Test

Our analysis of various datasets show that sometimes personal users may repeat some messages created by other users, which do not count as their own observations. The repeated messages not only produce redundancy, but also generate noises for analysis. Thus it is crucial to determine message originality. We deploy the same set of rules previously used for user originality test, shown in Table 4.8. A message satisfies any of the rules in the table is considered non-original, and will be filtered out.

Some repeated messages are obvious to identify, such as retweets, which have “RT” at the beginning of the messages. Other forms of repeated messages can be more difficult to spot, such as indirect quotes, which often but not necessarily contain the word “says” or “claims”. Given the various ways a message may be repeated, the rules listed in Table 4.8 do not cover all non-original messages. In our experiments, nevertheless, we find these rules filter out most of the repeated messages.

Lexical Analysis Algorithm

Algorithm 5 describes our lexical analysis method. The input is a keyword \( w \), and a set \( T \) of tweet texts containing the keyword. The output is a set of predictions of whether each tweet text \( t \in T \) is an observation, \( O \). In line 7, we use a parameter \( \theta \) to control the level of objectivity a tweet requires to meet to be considered an observation. The default value for \( \theta \) is the first quartile of overall objectivity in the tweet set. POS, objectivity and originality tests are done in line 4-12. The prediction will be positive if and only if the tweet passes all three tests (line 13).

4.5.2 User Profiling for Personal Account Classification

Previous works have shown that news generated from personal observations on Twitter can be much faster than traditional media, and the implicitly-associated location data can be used
for localizing the object or the event [53]. However, there are many Twitter accounts that are not for personal use, and do not have the same time and location association for their observation messages, and while they add noises to the data collected, it is usually difficult to distinguish them from personal accounts. The main issue is that all accounts on Twitter uses the same format to store data, and usually there is no effective way to judge the type of account other than looking at the content of the account posts directly. These accounts include news, business, activist and advertisement accounts. We call these latter types of accounts \textit{specific-purpose accounts}, and show some well-known examples in Table 4.10.

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
\textbf{News} & @cnnbrk @wsj @foxnews @huffingtonpost @bbcworld @politico \\
\hline
\textbf{Business} & @AdamDenison @GMblogs @MarriottIntl @chicagobulls @Marvel \\
\hline
\textbf{Activist} & @Greenpeace @femmajority @OU_Unheard @freedomtomarry \\
\hline
\end{tabular}
\end{table}

Our study of personal and specific-purpose accounts leads to the following observations:
- News accounts tweet about various topics in a strictly objective tone. Their tweets usually contain links to Web articles. Depending on the specialty, a media account can cover a wide range of topics.

- Business accounts contain conversations, observations, and product promotions, but the range of topic is limited to the specific business.

- Activist and advertisement accounts rarely use objective tone, and their range of topics is also limited.

A personal account, however, does not have such clear-cut characteristics as specific-purpose accounts, and usually contains a mix of information sharing, conversation with other users, and original content that covers various topics. We propose that:

**Conjecture 4.5.1** A personal account has moderate levels in objectivity, interactivity, originality, and topic focus.

We use various statistics generated from Twitter data to calculate the levels of objectivity, interactivity, originality, and topic focus for Twitter users. To profile a user, first we collect a set of past tweets made by the user, $H$. Then we select the original tweets in $H$ based on the rules described in Table 4.8, as $OH = \{oh_1, oh_2, ..., oh_l\}$, where $|OH| = l$.

The objectivity of a user is calculated based on the objectivity of each tweet in $OH$:

$$u_{objectivity} = \frac{\sum_{i=1}^{l} objectivity(oh_i)}{l}$$

The interactivity of a user is calculated based on the number of tweets containing mention mark “@” in $H$:

$$u_{interactivity} = \frac{\text{count}@ (H)}{|H|}$$
The originality of a user is calculated based on the fraction of original tweets in $H$.

$$u_{originality} = \frac{l}{|H|}$$

Calculating a user’s topic focus has been introduced in the previous section. Given a descendingly-sorted list of topic word occurrences $\{nt_1, nt_2, ..., nt_n\}$, the topic focus of a user is calculated based on the fraction of the first quarter of the most frequent topic words:

$$u_{focus} = \frac{\sum_{i=1}^{n/4} nt_i}{\sum_{j=1}^{n} nt_j}$$

A user is thus profiled by the quadruple:

$$u = \{u_{objectivity}, u_{interactivity}, u_{originality}, u_{focus}\}$$

### 4.5.3 Personal Account Classification with Profiles

We propose an algorithm for automatically identifying personal accounts based on the user profile. First we define the difference between two user profiles $u_1$ and $u_2$ as the Euclidian distance between two profiles:

$$d(u_1, u_2) = \sqrt{\sum (u_1 - u_2)^2}$$
where

\[
\sum (u_1 - u_2)^2 = \left( u_{objectivity_1} - u_{objectivity_2} \right)^2 \\
+ \left( u_{interactivity_1} - u_{interactivity_2} \right)^2 \\
+ \left( u_{originality_1} - u_{originality_2} \right)^2 \\
+ \left( u_{focus_1} - u_{focus_2} \right)^2
\]

Following Conjecture 4.5.1, we see that the attributes of a personal account are usually closer to a set of mean values while a specific-purpose account usually holds more extreme values. Therefore we propose that:

**Conjecture 4.5.2** Given a set of user profiles U, which contains personal account profiles P and specific-purpose account profiles S, there exists a mean profile \( \bar{u} \), such that \( \sum_{p \in P} d(p, \bar{u}) < \sum_{s \in S} d(s, \bar{u}) \).

While it is difficult to prove Conjecture 4.5.2, we find it generally true in our analysis, as we will show with our experiments. Given a set of user profiles U, and a mean profile \( \bar{u} \), we can separate from U a subset C that is more likely to contain personal accounts, by selecting profiles that have shorter distance to \( \bar{u} \).

We devise an iterative algorithm for finding the mean profile \( \bar{u} \). Intuitively, we can use the mean attribute values of all profiles in U. However, the extreme attribute values of the specific-purpose account profiles can bias the mean significantly, making it inaccurate for deciding personal accounts. In Algorithm 6, we use an iterative approach and a cluster size threshold \( \delta \) for selecting a cluster of \( |U| \times \delta \) profiles that are close to an unbiased \( \bar{u} \). Starting from an initial mean profile \( \bar{u}_0 \), the algorithm alters between cluster updating (line 2, 6) and mean updating (line 4 and 5). In the cluster updating step, a number of profiles close to the mean are selected. In the mean updating step, a new mean is calculated based on the selected profiles. If there are extreme values that cause a bias in the cluster, the mean will move away
from the bias, and replace the extreme value profiles with more average profiles in the cluster. The output of the algorithm, \( F \), is a set of personal account predictions.

**Algorithm 6** Predicting Personal Accounts

**INPUT:** user profiles \( U \), mean profile \( \bar{u}_0 \), selected cluster size \( \delta \)

**OUTPUT:** \( F \)

1. set all \( f \in F \) as false
2. \( C \leftarrow |U| \times \delta \) profiles closest to \( \bar{u}_0 \)
3. while \( C \neq C' \) do
4. \( C' \leftarrow C \)
5. \( \bar{u} \leftarrow \) mean attribute values of profiles in \( C \)
6. \( C \leftarrow |U| \times \delta \) profiles closest to \( \bar{u} \)
7. end while
8. for each \( u \in U \) do
9. if \( u \in C \) then
10. \( f_u \leftarrow true \)
11. end if
12. end for

While Algorithm 6 generally finds a good mean profile that separates personal accounts and specific-purpose accounts. However, depending on the choice of the initial mean \( \bar{u}_0 \), the algorithm sometimes produces undesirable results. To address this issue, we derive a particle swarm optimization (PSO) algorithm for finding the optimal \( \bar{u}_0 \).

PSO is an optimization technique that takes a population of solutions, and iteratively improves the quality of the solutions by moving them toward the best solution in each iteration [171]. A solution in our PSO algorithm is an initial mean \( \bar{u}_0 \) to be given to Algorithm 6. A PSO algorithm requires the definition of two components, namely, the quality measure, and the solution movement. To define the quality of a solution, we rely on our initial observation that personal accounts exhibit higher variance than any types of specific-purpose accounts. Therefore we propose that:

**Conjecture 4.5.3** Given two user profile clusters \( C_1 \) and \( C_2 \), if the profiles in \( C_1 \) are more diverse than \( C_2 \), than \( C_1 \) is more likely to contain personal accounts.
We use pairwise profile differences to calculate the diversity of profiles in a cluster, \( C = \{c_1, c_2, \ldots, c_k\} \),

\[
div(C) = \frac{2 \times \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} d(c_i, c_j)}{k \cdot (k - 1)}
\]

For the solution movement in PSO, we set a moving speed \( v \) so in each iteration, a solution \( p \) moves towards the best solution \( p_b \) as:

\[
p \leftarrow p + (p_b - p) \cdot v
\]

Our PSO algorithm is shown as Algorithm 7. It starts with a number of random solutions (line 1) and for each solution, a profile cluster is generated using Algorithm 6 (line 2 to 4). Then iteratively, the PSO algorithm moves the best solution towards an optimal solution by comparing the cluster diversity with each solution (line 5 to 13).

**Algorithm 7** PSO for Finding Optimal \( \bar{u}_0 \)

**INPUT:** user profiles \( U \), selected cluster size \( \delta \), number of particles \( n \), speed \( v \)

**OUTPUT:** \( p_b \)

1: randomly choose \( n \) solutions \( P \) in the solution space
2: for each \( p \in P \) do
3: \hspace{1cm} generate a cluster \( C_p \) using Algorithm 6
4: end for
5: \( p_b \leftarrow p \) with highest \( div(C_p) \)
6: while \( p_b \neq p'_b \) do
7: \hspace{1cm} \( p'_b \leftarrow p_b \)
8: \hspace{1cm} for each \( p \in P \) do
9: \hspace{2cm} \( p \leftarrow p + (p_b - p) \cdot v \)
10: \hspace{2cm} generate a cluster \( C_p \) using Algorithm 6
11: \hspace{1cm} end for
12: \( p_b \leftarrow p \) with highest \( div(C_p) \)
13: end while

The optimal initial mean produced by Algorithm 7 can then be used in Algorithm 6 for selecting the cluster of personal account profiles. Although Algorithm 7 requires two more parameters, during our experiments we find the effect of changing \( n \) and \( v \) negligible for any
4.6 Experimental Analysis

\( n > 1,000 \) and \( v < 0.2 \), as the solution already reaches optimal values. Therefore we can confidently set \( n \) and \( v \) to fixed values. The only parameter that still affects the classification result is the cluster size parameter \( \delta \), which controls the portion of profiles in the data to be selected as personal account profiles.

4.5.4 Overall Algorithm

Algorithm 8 identifies observations from personal accounts. Given the input of a keyword \( w \) and a set of tweets \( M \), and the control parameter \( \theta \) and \( \delta \), the output is a set of predictions, \( R \), of whether each respective tweet is an observation of the object or event of interest from personal accounts.

**Algorithm 8 Filter Observations from Personal Accounts**

**INPUT:** keyword \( w \), messages \( M \), objectivity threshold \( \theta \), selected cluster size \( \delta \)

**OUTPUT:** \( R \)

1: set all \( r \in R \) as false
2: \( T \leftarrow \) tweet text from \( M \)
3: \( U \leftarrow \) user profiles from \( M \)
4: \( O \leftarrow \) run Algorithm 5 with \( w \), \( T \), \( \theta \)
5: \( p_b \leftarrow \) run Algorithm 7 with \( U \), \( \delta \)
6: \( F \leftarrow \) run Algorithm 6 with \( U \), \( p_b \), \( \delta \)
7: for each \( m \in M \) that has text \( t \) and user profile \( u \) do
8: \hspace{1cm} if \( o_t \land f_u \) then
9: \hspace{1cm} \hspace{1cm} \( r_m \leftarrow true \)
10: \hspace{1cm} end if
11: end for

4.6 Experimental Analysis

We conduct extensive experiments to test our proposed classification methods, including the perspective classification, supervised observation classification with user features, and unsupervised personal observation classification. In each set of experiments, we collect real Twitter data and run our approaches alongside baseline methods, and compare the results.
4.6.1 Message Perspective Classification

We first conducted a set of experiments for testing the perspective classification accuracy of different combinations of features and machine learning models, using a real dataset we collected from Twitter. In this section, we will present our experiment setup, measurement methods, and experimental results.

Data Collection and Preparation

We collected two set of tweets related to the Ukraine Crisis using Twitter’s Stream API\(^9\). The first set contains tweets posted in February and March 2014, over a period of 40 days. The second set contains tweets posted in November and December 2014, during a period of 56 days. The two sets of data contain over a million tweets. We randomly selected a subset of them and manually labelled them according to their perspectives. For each perspective we labelled 1,000 tweets. For the immediate observation perspective, we labelled 500 positive and negative examples. For the affection perspective, we labelled 418 positive examples and 582 negative examples. For the speculation perspective, we labelled 520 positive examples and 480 negative examples. As a result, for each perspective, we have a separate dataset, although the same tweet may be present in more than one datasets.

Measurements

We used three-fold cross-validation when testing classifiers. For each test, we divided the labelled data into three parts, and used two for training the classifier, and one for testing. After the classifier produced the prediction for the testing set, the predicted labels were compared with the original labels. If the classifier predicted positive for a perspective for a tweet, and the tweet was labelled positive in the dataset, we counted the prediction as a true positive. We measured the precision and recall for the positive results. The classification

\(^9\)https://dev.twitter.com/streaming/overview
Table 4.11 Perspective classification accuracy with lexical features

<table>
<thead>
<tr>
<th></th>
<th>Immediate Observation</th>
<th>Affection</th>
<th>Speculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>prec.</td>
<td>recall</td>
<td>f1</td>
<td>prec.</td>
</tr>
<tr>
<td>SVM Linear</td>
<td>0.527</td>
<td>0.622</td>
<td>0.57</td>
</tr>
<tr>
<td>SVM Polynomial</td>
<td>0.585</td>
<td>0.38</td>
<td>0.46</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.54</td>
<td>0.806</td>
<td>0.646</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.529</td>
<td>0.58</td>
<td>0.535</td>
</tr>
<tr>
<td>LDA</td>
<td>0.566</td>
<td>0.538</td>
<td>0.551</td>
</tr>
</tbody>
</table>

Table 4.12 Perspective classification accuracy results with lexical and textual features

<table>
<thead>
<tr>
<th></th>
<th>Immediate Observation</th>
<th>Affection</th>
<th>Speculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>prec.</td>
<td>recall</td>
<td>f1</td>
<td>prec.</td>
</tr>
<tr>
<td>SVM Linear</td>
<td>0.704</td>
<td>0.689</td>
<td>0.695</td>
</tr>
<tr>
<td>SVM Polynomial</td>
<td>0.821</td>
<td>0.325</td>
<td>0.464</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.837</td>
<td>0.339</td>
<td>0.473</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.694</td>
<td>0.7</td>
<td>0.697</td>
</tr>
<tr>
<td>LDA</td>
<td>0.792</td>
<td>0.485</td>
<td>0.6</td>
</tr>
</tbody>
</table>

precision was calculated as the percentage of true-positives in all predicted positives. The classification recall was calculated as the percentage of true-positives in all labelled positives in the dataset for that perspective. As an indicator of general accuracy, the F-value is calculated as $F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. In this study we ignore negative predictions. The use of negative reports for event analysis will be a future study topic.

Results and Discussion

We tested the classifiers with training and testing data in each perspective. First we tested them using lexical features only, then we tested them using both lexical and textual features. The results using lexical features only are shown in Table 4.11 and Fig. 4.3. The results using lexical and textual features are shown in Table 4.12 and Fig. 4.4. The accuracy of best performing algorithm in each test is highlighted in bold.

According to the results, for the Immediate Observation and Affection perspective, using lexical features only allowed a higher classification recall. However, adding textual
Fig. 4.3 Perspective classification accuracy with lexical features
4.6 Experimental Analysis

Fig. 4.4 Perspective classification accuracy results with lexical and textual features
features increased the classification precision significantly. For the Immediate Observation perspective, adding textual features increased the classification precision from 0.585 to 0.837. For the affection perspective, the precision was also increased to above 0.8. With over 0.8 precision, we can confidently trust that the classifier will provide true-positives most of the time. For the Speculation perspective, the addition of textual features was less significant, but it generally improved the classification accuracy for all classifiers, as evident in the increased F-values.

Among the classifiers, the Naive Bayes classifier generally provided the highest classification precision for the Immediate Observation and Affection perspectives, while Random Forest achieved the highest recall and F-value. Interestingly, for the Speculation perspective, the Random Forest classifier achieved the highest precision, while SVM with polynomial kernel achieved the highest recall.

We conclude that using both lexical and textual features generally improves classification accuracy than using only the lexical features, and that the choice of classifier depends on the goal of the task. If the aim is to collect immediate observations with as few false positives as possible, the Naive Bayes classifier should be used. If the aim is find as many as possible users’ emotional reactions towards an event, and wrongly classified messages are more tolerable, then the Random Forest classifier should be deployed.

### 4.6.2 Supervised Classification with User Features

We tested the effectiveness of the proposed user features on real Twitter datasets. Similar to the previous experiment, testing the effectiveness of the user features involves generating feature vectors from the testing data for each set shown in Table 4.6 and the combination of the sets. The feature vectors are then processed by supervised machine learning models. Particularly, we investigate four machine learning models previously used, including Support
Vector Machine (SVM), Linear Discriminant Analysis (LDA), Naive Bayes, and Random Forest.

**Data Collection and Preparation**

We collected two sets of tweets using the Twitter public API. The first set contained tweets that include the keyword “homeless”. The second set contained tweets that include the keyword “shooting”. We labeled a number of tweets in both sets according to whether the tweet is a personal observation of a homeless person or a shooting incident. For the homeless dataset, we labeled 277 tweets, including 143 positive examples and 134 negative examples. For the shooting dataset, we labeled 298 examples, including 128 positive examples and 172 negative examples. For each tweet, we collect up to 100 most recent tweets of the user for generating user features.

**Baseline Methods and Measurements**

The user features were tested combined with a set of base features. The base feature set we chose, called Sakaki features, was proposed by Sakaki et al. [53], which include two features, namely, the number of words, and the index of the keyword. It was found that this feature set was the most effective for classifying observations among other feature sets studied, such as the words in the tweet, and the words before and after the keyword [53]. We added the user features to the Sakaki features, and the feature vectors generated for testing contained the Sakaki features and some or all of the user features. Individual user feature sets and different combinations of them were tested.

We used precision and $f$-value as measurements of the classification accuracy, with regard to positives. As an indicator of overall accuracy, the $f$-value of a classification result is calculated as $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. Higher results in both precision and $f$-value will mean a
comparable recall. For each classification, three-fold cross validation was used to generate the measurements [172].

Results and Discussion

Using feature sets indicated by Table 4.6, the precision and $f$-value for each combination of feature set and machine learning model is listed in Table 4.13 and 4.14. The highest value for each feature set is highlighted in bold, and the highest value among all tests for the dataset is highlighted in italic.

<table>
<thead>
<tr>
<th>Dataset: homeless</th>
<th>SVM</th>
<th>NB</th>
<th>RF</th>
<th>LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sakaki</td>
<td>0.675</td>
<td>0.627</td>
<td><strong>0.730</strong></td>
<td>0.678</td>
</tr>
<tr>
<td>A</td>
<td>0.688</td>
<td>0.647</td>
<td><strong>0.801</strong></td>
<td>0.674</td>
</tr>
<tr>
<td>B</td>
<td>0.682</td>
<td>0.640</td>
<td><strong>0.709</strong></td>
<td>0.686</td>
</tr>
<tr>
<td>C</td>
<td>0.662</td>
<td>0.627</td>
<td><strong>0.728</strong></td>
<td>0.669</td>
</tr>
<tr>
<td>A+B</td>
<td>0.702</td>
<td>0.631</td>
<td>0.697</td>
<td><strong>0.704</strong></td>
</tr>
<tr>
<td>A+C</td>
<td>0.673</td>
<td>0.630</td>
<td><strong>0.732</strong></td>
<td>0.673</td>
</tr>
<tr>
<td>B+C</td>
<td>0.685</td>
<td>0.623</td>
<td><strong>0.720</strong></td>
<td>0.689</td>
</tr>
<tr>
<td>A+B+C</td>
<td>0.688</td>
<td>0.622</td>
<td><strong>0.722</strong></td>
<td>0.681</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset: shooting</th>
<th>SVM</th>
<th>NB</th>
<th>RF</th>
<th>LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sakaki</td>
<td>0.525</td>
<td>0.527</td>
<td><strong>0.590</strong></td>
<td>0.570</td>
</tr>
<tr>
<td>A</td>
<td>0.533</td>
<td>0.564</td>
<td>0.565</td>
<td><strong>0.591</strong></td>
</tr>
<tr>
<td>B</td>
<td>0.525</td>
<td>0.434</td>
<td>0.548</td>
<td><strong>0.572</strong></td>
</tr>
<tr>
<td>C</td>
<td>0.553</td>
<td>0.542</td>
<td><strong>0.651</strong></td>
<td>0.609</td>
</tr>
<tr>
<td>A+B</td>
<td>0.545</td>
<td>0.449</td>
<td>0.588</td>
<td><strong>0.595</strong></td>
</tr>
<tr>
<td>A+C</td>
<td>0.563</td>
<td>0.576</td>
<td>0.601</td>
<td><strong>0.627</strong></td>
</tr>
<tr>
<td>B+C</td>
<td>0.546</td>
<td>0.553</td>
<td>0.585</td>
<td><strong>0.618</strong></td>
</tr>
<tr>
<td>A+B+C</td>
<td>0.560</td>
<td>0.564</td>
<td>0.548</td>
<td><strong>0.608</strong></td>
</tr>
</tbody>
</table>

First we notice that, consistent with previous findings, the classification accuracy did not increase by considering more features. Random Forest was the machine learning model with which Sakaki features obtained the highest precision. With Random Forest, the precision was the highest when user feature set A was added to the Sakaki features for the homeless dataset, and when user feature set C was added for the shooting dataset. Using all user features
Table 4.14 Classification F-value

<table>
<thead>
<tr>
<th>Dataset: homeless</th>
<th>SVM</th>
<th>NB</th>
<th>RF</th>
<th>LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sakaki</td>
<td>0.696</td>
<td>0.711</td>
<td>0.685</td>
<td>0.701</td>
</tr>
<tr>
<td>A</td>
<td>0.700</td>
<td>0.726</td>
<td>0.662</td>
<td>0.698</td>
</tr>
<tr>
<td>B</td>
<td>0.688</td>
<td>0.725</td>
<td>0.700</td>
<td>0.705</td>
</tr>
<tr>
<td>C</td>
<td>0.679</td>
<td>0.707</td>
<td>0.692</td>
<td>0.685</td>
</tr>
<tr>
<td>A+B</td>
<td>0.689</td>
<td>0.708</td>
<td>0.710</td>
<td><strong>0.718</strong></td>
</tr>
<tr>
<td>A+C</td>
<td>0.680</td>
<td>0.716</td>
<td>0.675</td>
<td>0.697</td>
</tr>
<tr>
<td>B+C</td>
<td>0.699</td>
<td>0.704</td>
<td>0.700</td>
<td><strong>0.708</strong></td>
</tr>
<tr>
<td>A+B+C</td>
<td>0.698</td>
<td>0.693</td>
<td><strong>0.708</strong></td>
<td>0.707</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset: shooting</th>
<th>SVM</th>
<th>NB</th>
<th>RF</th>
<th>LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sakaki</td>
<td><strong>0.642</strong></td>
<td>0.609</td>
<td>0.598</td>
<td>0.478</td>
</tr>
<tr>
<td>A</td>
<td>0.649</td>
<td><strong>0.674</strong></td>
<td>0.599</td>
<td>0.552</td>
</tr>
<tr>
<td>B</td>
<td><strong>0.641</strong></td>
<td>0.468</td>
<td>0.528</td>
<td>0.477</td>
</tr>
<tr>
<td>C</td>
<td>0.639</td>
<td><strong>0.658</strong></td>
<td>0.569</td>
<td>0.582</td>
</tr>
<tr>
<td>A+B</td>
<td><strong>0.638</strong></td>
<td>0.523</td>
<td>0.593</td>
<td>0.558</td>
</tr>
<tr>
<td>A+C</td>
<td>0.664</td>
<td><strong>0.675</strong></td>
<td>0.536</td>
<td>0.609</td>
</tr>
<tr>
<td>B+C</td>
<td><strong>0.636</strong></td>
<td>0.601</td>
<td>0.530</td>
<td>0.587</td>
</tr>
<tr>
<td>A+B+C</td>
<td><strong>0.667</strong></td>
<td>0.614</td>
<td>0.515</td>
<td>0.593</td>
</tr>
</tbody>
</table>

resulted in precisions that were worse than using Sakaki features alone. However, we can see that, with the right choice of user features, the accuracy can be consistently improved. User feature set A improved the precision by around 7% for the homeless dataset, while user feature set C improved it by around 6% for the shooting dataset. Set C was more effective for the shooting dataset possibly because a newsworthy incident such as a shooting involves more messages from news accounts, and user feature set C is designed to provide distinction for these accounts. We can also see that the methods that achieved the highest precision also maintained a high overall accuracy without greatly sacrifice recall, indicated by comparable f-values.

Among machine learning models, we can see that Random Forest was the most effective in providing the highest precision. LDA, while unable to achieve the best result, tended to provide higher precision with more features. The highest f-value in both datasets were
achieved by Naive Bayes models. SVM, on the other hand, performed more consistently with different feature sets, and was able to achieve higher $f$-value with more features.

4.6.3 Unsupervised Observation Classification

We tested the effectiveness of our method for filtering personal observations on Twitter with two real Twitter datasets, comprising of a controlled dataset and a crowd-sourced dataset. In this section, we present the setup, measurement, baseline methods, and results of our experiments in detail.

Experiment Setup

We implemented the algorithms presented in the previous section in Java. The experiments were run on a MacBook Pro laptop computer, with 2.3GHz Intel Core i7 CPU and 8 GB 1600MHz DDR3 memory. We deployed an existing implementation for POS tagging. After comparing several existing POS tagging implementations including OpenNLP and LingPipe, we chose StanfordNLP POS module to run our POS tagging because it is relatively fast and provides a high tagging accuracy of around 95% [173].

For parameter $\theta$ in Algorithm 5, we chose the first quartile of overall objectivity in the dataset for all experiments, which generally provides good results. For parameter $\delta$, we compared three different values, including 0.7, 0.8, and 0.9. To ensure the consistency of the experiments, instead of randomly choosing initial values for the particles in Algorithm 7, we chose combinations of evenly distributed values for the four attributes as the initial values, i.e., 0.2, 0.4, 0.6, 0.8, and 1. Our analysis shows that randomly initialized particles provide similar results. For user profiling, up to 1,000 recent tweets were collected for each user using Twitter Timeline API.
Baseline Methods and Comparison Metrics

We compared our approach with three baseline filtering strategies, namely, Accept All, Sakaki filter, and Sriram. Accept All takes all tweets in the dataset as the positive for personal observations. Sakaki classifier, introduced in the previous section, is a supervised method that deploys a Support Vector Machine (SVM) classifier with linear kernel built on manually annotated training data. Among the three feature sets proposed in [53], we implemented the reportedly most effective set, Feature Set A, which is based on word counts and keyword positions. We deployed an existing SVM implementation in an R language package called e1071\(^\text{10}\). We used a weighting function according to class imbalance to ensure optimal performance of the classifier. The performance of the Sakaki classifier was measured using the three-fold cross validation. One drawback of the Sakaki classifier is that it requires the presence of a keyword. The user profiling in our approach, though, does not have this requirement.

Sriram classifier, proposed by Sriram et al. in [16], is also a supervised method that is based on eight features and the Naive Bayes model. The eight features include author name, use of slang, time phrase, opinionated words, and word emphasis, presences of currency signs, percentage signs, mention sign at the beginning and the middle of the message. The evaluation is based on the five-fold cross validation. The Sriram classifier is shown to be effective in classifying tweets into categories such as news, opinions, deals and events, but has not been tested in other applications.

All datasets for evaluation were manually annotated according to whether each tweet is a personal observation of an object or event of interest, which were considered ground truth in our experiments. The output of the filtering methods were compared with the manual annotations. If a filtering output is positive in manual annotations, it is considered a *true positive*. We use *precision*, *recall* and *f – value* as the measurements of filtering accuracy,

\(^{10}\text{https://cran.r-project.org/package=e1071}\)
where given the set of positive filtering results $P$ and the set of true positives in the dataset $TP$, the precision $= \frac{|P \cap TP|}{|P|}$, recall $= \frac{|P \cap TP|}{|TP|}$, and $f$-value $= 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$.

**Effectiveness on Controlled Dataset**

We first tested our method on two controlled datasets. We collected a dataset containing the keyword *hailstorm* during August, 2015, and a dataset containing the keyword *car accident* during September, 2015. Each dataset has several thousands of tweets. After removing retweets and tweets containing links, we manually labelled the remaining tweets as positive or negative examples, according to whether the message is about a direct observation of a hailstorm or a car accident. The resulted *hailstorm* dataset contains 675 tweets, with 251 positive examples and 424 negative examples. The labelled *accident* dataset contains 954 tweets, with 347 positive examples and 607 negative examples.

We tested the filtering methods on two datasets. Accuracy results for the baseline methods, lexical analysis-only filtering (LX), and lexical analysis combined with personal account filtering using three $\delta$ values, PA($\delta = 0.9$), PA($\delta = 0.8$), and PA($\delta = 0.5$), are presented in Table 4.15.

<table>
<thead>
<tr>
<th></th>
<th>Accept All</th>
<th>Sakaki</th>
<th>Sriram</th>
<th>LX</th>
<th>PA ($\delta = 0.9$)</th>
<th>PA ($\delta = 0.8$)</th>
<th>PA ($\delta = 0.5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hailstorm dataset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>precision</td>
<td>0.37</td>
<td>0.43</td>
<td>0.37</td>
<td>0.53</td>
<td>0.62</td>
<td>0.64</td>
<td>0.70</td>
</tr>
<tr>
<td>recall</td>
<td>1</td>
<td>0.70</td>
<td>0.98</td>
<td>0.80</td>
<td>0.76</td>
<td>0.71</td>
<td>0.46</td>
</tr>
<tr>
<td>f-value</td>
<td>0.54</td>
<td>0.53</td>
<td>0.54</td>
<td>0.63</td>
<td><strong>0.68</strong></td>
<td>0.67</td>
<td>0.56</td>
</tr>
<tr>
<td><strong>car accident dataset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>precision</td>
<td>0.38</td>
<td>0.50</td>
<td>0.44</td>
<td>0.53</td>
<td>0.58</td>
<td>0.59</td>
<td><strong>0.60</strong></td>
</tr>
<tr>
<td>recall</td>
<td>1</td>
<td>0.73</td>
<td>0.84</td>
<td>0.76</td>
<td>0.74</td>
<td>0.69</td>
<td>0.43</td>
</tr>
<tr>
<td>f-value</td>
<td>0.55</td>
<td>0.60</td>
<td>0.57</td>
<td>0.63</td>
<td><strong>0.65</strong></td>
<td>0.64</td>
<td>0.50</td>
</tr>
</tbody>
</table>

As shown in the table, the Accept All strategy captured all the positives in the annotations and had the maximum recall of 1. All other methods improved the precision by sacrificing
the recall to some degree. Personal account filtering with $\delta$ set to 0.9 achieved the highest overall performance, indicated by the highest f-value. Using lexical analysis only and PA with $\delta = 0.9$ and $\text{delta} = 0.8$ all performed better than the Sakaki classifier and the Sriram classifier, the latter of which provided almost no filtering effect in the hailstorm dataset. Setting $\delta$ to a lower value improved the precision but also lowered the recall. When setting $\delta$ to 0.5, PA achieved the highest precision, while still held a relatively high f-value. The performance of all methods were consistent across two datasets, with LX improve precision from the Accept All strategy by around 15% and PA($\delta = 0.9$) further improved it by 5%-7%.

**Effectiveness on Crowd-Sourced Dataset**

We tested our approach on a publicly available dataset, which we call crises dataset. The data was produced by Castillo et al. [15], and was made available online\(^{11}\). The dataset contains a number of tweets related to crisis events, such as the Colorado wildfires and the Pablo typhoon in 2012, and the Australia bushfire and the New York train crash in 2013. The tweets were manually annotated by hired workers on Crowdflower, a crowdsourcing platform\(^{12}\). The tweets were labelled according to their relevance to the crisis event, and the type of information they provide. There are four relevance categories, namely, related and informative, related but not informative, not related, and not applicable. The seven information types include Eyewitness, Government, NGOs, Business, Media, Outsiders, and Not applicable.

We consider that the Eyewitness-type tweets in dataset are personal observations, while other types of tweets are not. Hence we expect our approach to filter Eyewitness tweets from other tweets. With this goal, we re-organized the dataset. First we selected two categories of related tweets from the dataset. Then we selected five information types of tweets, including Eyewitness, Government, NGOs, Business and Media. We then produced a list of labels.

\(^{11}\)http://crisislex.org/
\(^{12}\)http://www.crowdflower.com/
with Eyewitness tweets as positives, and other types of tweets negatives. We also removed retweets from the data. Finally, we had a labelled dataset of 3646 tweets that includes 528 positives.

Since the tweets do not contain a specific keyword, we did not run POS and objectivity analysis. The Sakaki classifier is also not applicable without a keyword. As such we ran the originality test in the lexical analysis and the personal account classification, and compared only to the Sriram classifier, as shown in Table 4.16.

Table 4.16 Filtering accuracy for the crisis dataset

<table>
<thead>
<tr>
<th>Accept All</th>
<th>Sriram ($\delta = 0.9$)</th>
<th>PA ($\delta = 0.8$)</th>
<th>PA ($\delta = 0.5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision</td>
<td>0.14</td>
<td>0.32</td>
<td>0.64</td>
</tr>
<tr>
<td>recall</td>
<td>1</td>
<td>0.52</td>
<td>0.50</td>
</tr>
<tr>
<td>f-value</td>
<td>0.24</td>
<td>0.40</td>
<td>0.47</td>
</tr>
</tbody>
</table>

The results are similar to previous experiments, where PA($\delta = 0.9$) achieved the highest f-value and PA($\delta = 0.5$) achieved the highest precision. The lexical analysis was particularly effective for this dataset, improving the precision by around 38%, mainly because the dataset includes a large portion of news messages, which failed the originality test. After the lexical analysis, PA($\delta = 0.9$) further improved the precision by 12%. Both LX and PA($\delta = 0.9$) significantly outperformed the Sriram classifier.

Discussion

Even though our method effectively improves personal observation filtering accuracy, there is still a large gap before perfect filtering accuracy can be achieved. We identify three main reasons. First, the majority of personal users behave similarly to a mean profile, however, due to the large diversity of personal users, there is a small portion who do not follow a mean profile. For example, there are users who regularly post personal observations of surrounding environments, but do not have any interaction with other users, and thus are identified as non-
personal accounts. Second, a personal account does not guarantee that its tweets will always be personal observations. We have listed a number of rules for identifying non-original messages, but they do not cover all patterns a personal user may repeat or cite information. Third, the natural language processing components in our method generally do not provide perfect accuracy. As previously mentioned, the POS tool provided in StanfordNLP only has a precision of 0.95. The objectivity analysis and user profiling are largely based on a number of dictionaries, which generally do not provide perfect coverage and accuracy. Further work is needed to further improve the accuracy of text-based message filtering, for example, by incorporating location information and name-entities analysis.

### 4.7 Summary

One way to see microblogs is that they are social sensing networks with users as social sensors and the observation messages as sensing values. However, we see that not all messages posted on microblogs are personal observations of events or objects of interest. To realize the social sensing aspect of microblogs, the first step is to classify observation messages from other type of messages, and without doing this what we collect will be unrecognizable noisy data. In this Chapter, we present our work on observation classification methods.

It is known that classifying event-related messages is a challenging task, with existing works achieve accuracies around 65%. We research various ways to improve the classification accuracy. First we consider the perspectives on which a message might be composed, which generates high classification precisions around 80%. Then we consider the user features. User features have been largely overlook in existing observation classification works. We propose various user features, including communication features, trending activity features, and writing style features. Experiments shows that incorporating user features in supervised learning methods can consistently improve classification accuracy, providing up to 7% improvement from existing work.
We then propose an unsupervised method for personal observation classification. Our method consists of observation classification and personal user profiling. We run unsupervised tests such as part-of-speech, objectivity, originality tests on messages, and use unsupervised clustering for selecting personal users. Combining the lexical analysis and user profiling, our method provides high classification accuracies. Experiments with controlled datasets show that applying our method generates a 22% improvement on accuracy with 15% contributed by lexical analysis and 7% by user profiling. Experiments with a publicly available cloud source dataset shows an even more significant improvement of around 50%. Accurate observation classification provides a foundation for social-sensing data analysis, and allows us to confidently explore various information extraction with social sensing data.
Chapter 5

User Location Profiling for Localized Event Detection

In the previous chapter, we introduced our solution on observation classification. However, having observation messages is not enough to fully utilize social sensing. To fully convert the observation messages to sensory values, we need associated time and location information. While time information is readily available for every tweet, location information is unfortunately sparse on Twitter. Our investigation shows that, only 0.9% tweets has associated GPS data. Current works proposed to infer locations from tweets usually focus on coarse-grained locations, and generally generate large error when attempting to infer fine-grained locations.

To increase the availability and accuracy of location information on Twitter, we propose methods to extract locations from tweet messages and inferring the current location based on the user’s past locations. We test our approach with extensive experiments, and show that our methods increase the availability of location information to 87%, and significantly improve the inference accuracy at fine-grained levels, with a 16% increase within the 3km error range, and 20% within 5km error range. Based on the inferred locations, we propose Sense and Focus (SNAF), an event monitoring system that detects and localizes emerging events reported on Twitter.
5.1 Overview

In the previous chapter, we discussed methods that filter immediate observation messages from Twitter communication streams. The immediate observations of Twitter users can be seen as reading values of some sensory devices [9]. Take, for example, a user posting a tweet in Times Square on Tuesday morning, saying "the air is fresh". This can be seen as an air quality sensor installed in Times Square that just recorded a good reading on Tuesday morning. An immediate observation posted on Twitter can be converted to sensory values, if the time and location of the tweet are known. Given the high density of Twitter users in urban areas, the perspective of Twitter users as sensors instantly provides a dense virtual sensor network, by which some events and objects of interest can be monitored. Several works have proposed using tweets as sensory values to monitor environmental disasters [9, 14]. However, due to the lack of accurate and fine-grained location information, existing event monitoring systems for Twitter data only identifies city-level or coarser locations, and cannot provide accurate details for local events [14, 161].

As opposed to time information, which is readily available for all tweets, location information is scarce. Current works for obtaining the geo-location information of the tweets rely heavily on the tweets’ GPS data [9, 121]. In one of our experiments, we sampled one thousand random tweets, but found that only 0.9% had GPS data. Other studies found even lower figures [132]. This implies that if only GPS-enabled tweets are considered, most of the observations posted on Twitter will not be noticed. Suppose there is an air pollution around a certain corner of the city, even with ten Twitter users reporting it, the probability of overlooking the pollution can be as high as 0.91\(^1\). To increase the availability of location information, tweet messages and other publicly accessible information on Twitter need to be treated as location sources [14, 10].

\(^1\)As \((1 - 0.009)^{10} \approx 0.91\)
Extraction of location information from tweets has been studied by several works [10, 14, 128-132]. Many of these works focus on the city-level location [10, 130, 131]. However, finer-grained location information is often vital for local events, such as shooting incidents and vehicle crashes. Previous studies show that, exact location extraction from tweet texts often result in large errors [128, 132]. The issues include the use of informal names, mentioning of a placename other than the place where the user is located, and the lack of a comprehensive gazetteer.

One of the issues that makes street-level location inference from tweet messages difficult is the use of informal names. In tweet messages, “can” could mean Canada, “philly” could mean Philadelphia, and “central park” could mean the Central Park in New York or Sydney. It is difficult to find out all associations between locations and their informal names, whether using a learning model [128] or a gazetteer [132]. Even if a comprehensive gazetteer is generated to cover all informal names, there will be another problem, that many words that do not mean placenames will be captured as placenames. Essentially, it is the noises in placenames extracted from tweet messages that prevent more accurate location inference.

We follow the gazetteer approach for solving the message location inference problem, and propose a method that increases both the availability and the accuracy of the location information. Our gazetteer is generated from DBpedia data, a large user-contributed database for name-entities used in the Web [174]. We increase the availability of the location information by looking into placename mentions in user’s past tweets, and improve the accuracy by applying data cleaning techniques. We select and implement two existing distance-based sensor data cleaning algorithms, and conduct comprehensive experiments to test them. Our experimental results with real Twitter data show that, compared to the existing approaches, our method improves the accuracy of location inference by 5% to 20% across different error ranges.
Based on the inferred locations, we propose SNAF (Sense and Focus), an event monitoring system that captures local events reported on Twitter and accurately infers event locations. SNAF aggregates immediate observations of an event or an object of interest based on their locations, and generates an alarm when an emerging event is detected. Given the accurate and high-recall location information provided by the our location inference method, the system is able to successfully capture local events, in many cases several hours faster than news reports, while the locations inferred are accurate compared to the locations reported in the news.

### 5.2 Related Work

We discuss here a current research trend of converting microblog messages to sensory values for monitoring events and objects of interest [10, 9]. Sakaki et al. [9] proposed and investigated the idea of using tweets as sensory values for environmental event monitoring. They collected earthquake-related tweets and filtered them to generate reports about ongoing earthquakes. Using this information, they built a system for predicting the movements of earthquakes in Japan. Their results show that the use of tweets as sensory values is feasible, and that their system can detect earthquakes within a minute after they occur, five minutes faster than announcements from meteorological authorities. In a later work, they extended the system to predict typhoon movements, which achieves a similar accuracy and response time [53]. Machine learning approaches are also used in other works for filtering observations of particular events [16, 175, 160, 15]. Sriram et al. [16] proposed a machine learning-based filter for classifying tweet categories such as news, opinions, events, and private messages, using features including author name, the use of opinionated words, currency signs, and mention signs. Kwon et al. [160] identified effective lexical and temporal features for distinguishing rumors in event-related tweets. Olteanu et al. [15] proposed keyword selection methods for filtering tweets related to particular natural disasters. While these
works identified tweets that can be used for monitoring particular events or objects of interest, they are not sufficient for converting tweets into sensory values, as the majority of tweets are not explicitly associated with location data.

Current researches on location inference on Twitter focus on different granularities. A category of research aims at inferring city-level locations [10, 131, 130]. Li et al. [10] inferred tweet location by first inferring the user’s home location, relying on the home location entries in Twitter’s user profiles, which are usually entered as cities. Graham et al. [130] also exploited home location entries in user profiles for inferring the location of the tweets and users.

Another category of research aims at inferring finer-grained locations [129, 14, 128, 132]. Li et al. [129] trained a classifier to associate placenames found in tweet messages with formal names, based on Foursquare data. Ikawa et al. [128] inferred tweet locations by matching the tweet with tweets with known locations using cosine similarity, and also based on Foursquare data. Foursquare is a location-based service that provides accurate place-coordinate association for commercial places such as hotels and restaurants, but in a recent business operation, the service has been terminated\(^2\). Schulz et al. [132] leveraged DBpedia for identifying places in tweet messages, but resulted in large errors, with the median error distance of 1,100km. DBpedia is a large user-contributed database for name-entities used in the Web [174]. Using a generation tool called DBpedia Spotlight [176], we can generate a gazetteer containing more than 800,000 places with respective GPS coordinates, which covers a large number of street-level places. However, as we will discuss below, such large gazetteer will make location information very noisy.

The earthquake and typhoon detection system proposed in [53] takes all tweets containing the keyword as the tweets for a single event, assuming that earthquakes and typhoons do not occur frequently. Local events such as vehicle crashes occur in much higher frequencies with smaller impact areas. Local event detection thus requires clustering, preferably with location

\(^2\)http://time.com/3024078/foursquare-swarm/
information [177, 10, 161]. Unankard et al. [161] proposed an event detection method based on clustering of words. They relate events to locations, but only at a country-level. Li et al. [10] proposed a classification-based event detection method for crime and disaster events, with a location extraction component focuses on city-level locations. The lack of accurate and fine-grained location information is the main issue that prevents existing event monitoring systems from obtaining local details.

Noisy sensor readings have been extensively studied in sensor networks. Basic techniques include temporal and spatial aggregation using mean or median [78, 133, 142]. Particularly, the median is effective for avoiding extreme outlier values in the dataset [133]. More advanced techniques identify and remove outliers in the dataset, based on distance measures or clustering. Since it is straightforward to interpret the distance of two location points, the distance-based outlier detection is particularly suitable for location data. Subramaniam et al. [79] and Branch et al. [143] both used $k$-Nearest Neighborhood (KNN) for identifying outliers. An issue for using KNN is to choose the proper distance threshold that excludes the outliers. Sheng et al. [146] proposed two KNN-based outlier detection algorithms, using a fixed distance threshold and a relative distance threshold, respectively.

5.3 SNAF: Location Inference and Event Detection

SNAF consists of a Sense component and a Focus component, as illustrated in Fig. 5.1. To Sense, SNAF filters immediate observations of certain events or objects of interest from Twitter streams. To Focus, SNAF infers the location of each report, and aggregates reports based on their locations for detecting local events. As mentioned earlier, the main issue for event detection system for Twitter data is the lack of accurate and fine-grained location information. SNAF challenges this issue by using a large gazetteer and users’ past tweets to increase the availability of location information, and applying sensor data cleaning techniques to remove the noises in the data. In the previous chapter, we have discussed immediate
observation filtering techniques. In this section, we discuss the Focus component’s *placename extraction, location resolution using past locations, and aggregation for event detection.*

Fig. 5.1 The overall architecture of SNAF: Sense and Focus

### 5.3.1 Location Extraction for A Single Tweet

To infer the location of a report, we consider the tweet text as well as past tweets of the user who made the report as sources of locations. For each of these tweets, we run a gazetteer-based name extraction by comparing the words in the tweet text with gazetteer entries. As expected, the accuracy of this approach depends on the quality of the gazetteer. When testing a gazetteer, we found two potential issues, *coverage* and *noise*. The coverage problem occurs when a placename is present in the tweet but not in the gazetteer. The noise problem occurs when a word in the tweet matches an entry in the gazetteer but does not mean a placename in the context of the tweet. We take both issues into account when building our gazetteer.

We generated our base gazetteer using the latest version of DBpedia geo-coordinate dataset[^3] and the DBpedia Spotlight name generation tool [176]. DBpedia is a user-contributed name-entity database for Internet-of-Things data lookup, and in a recent version includes

[^3]: http://wiki.dbpedia.org/Datasets
4.58 million names or things. The geo-coordinate dataset contains a subset of these names that are associated with a geo-coordinate. DBpedia Spotlight can track the usage of these names in Web documents such as Wikipedia pages, and produce a gazetteer containing formal and informal names. Our base gazetteer contains more than 800,000 entries and covers a large number of street-level places, including their formal and informal names, such as “penn station”, “lion’s head”, and “east end”. However, it also contains a large number of names that are usually not referring to a place when people mention them on Twitter. Table 5.1 shows examples of four types of such names.

Table 5.1 Examples of non-placenames in the geo-coordinate dataset

<table>
<thead>
<tr>
<th>Common Words</th>
<th>in, can, bath, young, act, space, orange, king, cook, north, bucks, tower, paradise, marathon, racing, shape, corner, path, estate, sandwich</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person’s Names</td>
<td>george, williams, georgia, lewis, nelson, anderson, stanley, monroe, katherine, clinton, reagan, douglas, harrison</td>
</tr>
<tr>
<td>Brand and Organization Names</td>
<td>nike, microsoft, nintendo, tesco, fbi, dell, nsa, red cross, mercedes, cadillac, java, absolute radio, chipotle, texas tech</td>
</tr>
<tr>
<td>Symbolic Place Names</td>
<td>troy, sparta, wall street, pentagon, third reich, green line, berlin wall, tiananmen square, pearl harbor, notre dame</td>
</tr>
</tbody>
</table>

We use a heuristic consists of two rules to address the two issues mentioned above and refine the gazetteer. First we collect a set of GPS-enabled tweets and extract locations from the messages using the base gazetteer. The extracted locations are compared with the GPS data. If the difference between the extracted location and the GPS data is larger than an acceptable threshold $\lambda$, we count a hit for the location. If the difference is larger than a rejection threshold $\theta$ where $\theta > \lambda$, we count a miss for the location. Usually a common location word will be extracted more than once. After the process, each extracted location will have a hit number and a miss number. We then pick the locations with at least one hit to put in the refined gazetteer, which ensures the coverage of the refined gazetteer. On the other hand, if the number of misses is more than half of the total extraction number, we remove it from the refined gazetteer. The purpose of this heuristic is to keep all the gazetteer
entries relevant to the GPS data, with respect to the training dataset, which reduces the noise of the gazetteer. In our experiments we chose $\theta = 10\text{km}$ and $\lambda = 300\text{km}$. These numbers are relatively stricter because we focus on smaller granularity, but different numbers can be chosen with different requirements in error tolerance.

After we had a refined gazetteer, we evaluated the precision of our gazetteer by comparing the coordinates of the location extracted from a single tweet with the GPS associated with the tweet. The results are shown in Table 5.2, where the precision for each distance range is the percentage of locations in all extracted locations that is within that distance range when compared to the GPS data.

<table>
<thead>
<tr>
<th>Error</th>
<th>$\leq 3\text{km}$</th>
<th>$\leq 5\text{km}$</th>
<th>$\leq 10\text{km}$</th>
<th>$\leq 30\text{km}$</th>
<th>$\leq 100\text{km}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.225</td>
<td>0.283</td>
<td>0.372</td>
<td>0.518</td>
<td>0.587</td>
</tr>
</tbody>
</table>

The precision of the name extraction on single tweets based on the gazetteer is relatively low. While we can use the general knowledge to distinguish, for example, a person’s name from a placename, this judgement is not always accurate. Furthermore, a place mentioned by the user does not always reflect the location of the user or the tweet. For example, when a user tweets that he wants to go to *Gold Coast* for a holiday, this place is probably not related to the current location of the user or the tweet, even if the placename is successfully extracted from the tweet. Table 5.3 shows examples of placenames in tweets that do not indicate the current location of the user, where the error is the distance between the mentioned place and the tweet’s GPS coordinates.
Table 5.3 Examples of placename in tweets not indicating user location

<table>
<thead>
<tr>
<th>tweet</th>
<th>place</th>
<th>error(km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Someone come with me to the South Shore Plaza</td>
<td>South Shore Plaza</td>
<td>8458</td>
</tr>
<tr>
<td>What are the chances of @bassnectar playing Outside Lands this weekend? #ibelieveinmiracles #bayjams</td>
<td>Outside Lands</td>
<td>12800</td>
</tr>
<tr>
<td>He was not armed when taken into custody. Just getting this info in from our CBS affiliate in Rockford.</td>
<td>Rockford</td>
<td>9926</td>
</tr>
<tr>
<td>going to see the new school</td>
<td>The New School</td>
<td>8666</td>
</tr>
</tbody>
</table>

Therefore, even if we provide a gazetteer that accurately extracts placenames from tweets, we cannot avoid a certain level of noises, and these noises prevent location inference from reaching high accuracy. More location information and further processing are required for inferring more accurate locations. We will obtain more location information by looking at users’ past tweets.

5.3.2 Location Resolution Given Past Locations

When considering the past tweets from a user, we can generally extract a handful of past locations. The past tweets of Twitter users are publicly accessible through the timeline API provided by Twitter⁴, with a restricted availability of a maximum of 3,200 most recent tweets. About 5% tweets have a placename that can be identified using DBpedia-based gazetteer [132]. We conducted an experiment to analyze 1,000 random tweets and found that, while only 0.9% of tweets have GPS tags, 87% of the 982 users who made these tweets have at least one extractable placename in their past tweets, and 68% of them have more than ten. By using placenames extracted from past tweets to infer the current location, we can effectively associate location information with a significant portion of tweets.

⁴https://dev.twitter.com/rest/public/timelines
A general intuition is that the locations of a person are spatially correlated, and the locations appear in the past tweets can be used to infer the location of the current tweet. To verify this intuition, we investigated the temporal-spatial pattern of Twitter users based on GPS data. We extracted the past locations of 743 random users with at least 100 GPS-enabled tweets, and divided the location into months, based on number of days past until the last tweet. We then calculated the mean distance of locations in each month to the location of last tweet for each user. The results as the average of all users are shown in Fig. 5.2.

Fig. 5.2 Mean location differences and percentages for tweets in different past months

The results show that, more than 63% of the tweets in the past year are within 100km from the last tweet, regardless how old they are. Also, even in the worst month, the eighth month, there are more than 50% of the tweets are within 30km from the last tweet, and 37% are within 10km. Therefore we conclude that past locations can be a good indicator for inferring current locations.

Various techniques have been proposed to infer the current location of an object based on past locations, such as Kalman filters and Particle filters. Since it is difficult to define
a dynamic model for the movement of Twitter users, the Bayesian Filter-type inference techniques are ineffective. In sensor networks, a common way to aggregate spatial data is using the average. Particularly, median aggregation is effective for avoiding extreme values in the dataset [133]. More advanced techniques identify outliers in the dataset, which are removed before the data aggregation. Since it is straightforward to interpret the distance between location points, we choose two representative distance-based outlier detection techniques from the sensor network literature, called DK-Outlier and NK-Outlier [146].

Both outlier detection techniques are based on the \( k \)-Nearest Neighbor (KNN). Given a set of locations \( H \), and a location \( l \in H \), let \( V(l) \) be the distances from \( l \) to all other locations in \( H \), sorted in ascending order. Then \( V_k(l) \) is the distance from \( l \) to its \( k \)-th nearest neighbor. Introduced in [178], given an acceptable distance threshold \( d \), DK-Outlier is defined as:

**Definition 5.3.1** A location \( l \) is a DK-Outlier if \( V_k(l) \geq d \).

NK-Outlier is a new type of outlier proposed by [146]. NK-Outlier detection sets a relative distance threshold based on comparison between locations in the dataset. Given \( n \), the number of other locations in the data a location needs to be comparable in order to be considered acceptable:

**Definition 5.3.2** A location \( l \) is a NK-Outlier if there are no more than \( n - 1 \) other locations \( p \), such that \( V_k(l) < V_k(p) \).

The choice of \( d \), \( n \) and \( k \) depends on the desired data accuracy and expected errors in the dataset. For example, we consider an acceptable past location is within 100km of the current location, and thus the maximum distance between two acceptable locations is 200km, so we set \( d \) as 200. When testing the accuracy of location extraction on single tweets, we found 58.7% extracted locations are within 100km error range, and 24% locations extracted had an error larger than 1,000km, so we set \( k \) as \( 0.587 \times |H| \), and \( n \) as \( 0.76 \times |H| \). The algorithms for removing DK-Outlier and NK-Outlier are shown below as Algorithm 9 and 10.
Algorithm 9 Removing DK-Outlier

INPUT: $H$, extracted past locations of the user
OUTPUT: $H'$, past locations after outliers removed

1: $R \leftarrow \text{null}$
2: for each $l \in H$ do
3: $V \leftarrow$ distances from $l$ to all other locations in $H$
4: $VS \leftarrow V$ sorted in ascending order
5: if $VS_k < d$ then
6: add $l$ to $R$
7: end if
8: end for
9: $H' \leftarrow$ remove all $l \in R$ from $H$
10: return $H'$

For removing DK-Outlier, we generate a distance list of one location to all other locations, $V$, for each location in the data (line 3). For each location, we sort the distance list in ascending order, and compare the $k$-th element with the predefined threshold $d$ to decide if the location should be removed as an outlier (line 4 - 6).

For removing NK-Outlier, we also generate a distance list of each location to all other locations (line 3), and then take the $k$-th element in the ascendingly-sorted list as the representative value (line 4, 5). Then we compare the representative value of one location to other locations, count how many other representative values are greater than the one associated with this location (line 9 - 13), before deciding if the location should be removed as an outlier (line 14 - 15).

When examining the data, we notice that users may have a burst of interest in a certain location, and post messages about this location intensively within a short period of time. For example, after a football match, an user may engage in an intense conversation with a friend about the match, which involves mentioning of the name of the football team that will be detected as a placename. This phenomenon often distorts the location inference significantly. Therefore we propose a compression algorithm, called burst compression, which basically removes consecutive occurrences of a location within three days from the first occurrence, as
shown in Algorithm 11. This algorithm can be added to the above outlier removal algorithms as a pre-processing step.

To remove repetitive locations, first we associate each past location with the number of days between the message containing the location and the current message (line 1). Then we move along the timeline, making each location as a cursor (line 5). If next location is the same as the current location and the time lapsed is within three days, we mark the location for removal (line 6), otherwise we move the cursor to next location (line 9, 10).

The final inference of the location is thus the median location of all remaining locations extracted from the user’s past tweets, after outliers are removed.
Algorithm 11 Burst Compression

INPUT: $H$, extracted past locations of the user
OUTPUT: $H'$, past locations after outliers removed

1: $P \leftarrow$ days lapsed for each $l \in H$
2: $R \leftarrow null$
3: $curLoc \leftarrow H_1$
4: $curDay \leftarrow P_1$
5: for $i \leftarrow 2$ to $|H|$ do
6:     if $H_i = curLoc$ and $P_i - curDay \leq 3$ then
7:         add $H_i$ to $R$
8:     else
9:         $curLoc \leftarrow H_i$
10:        $curDay \leftarrow P_i$
11: end if
12: end for
13: $H' \leftarrow$ remove all $l \in R$ from $H$
14: return $H'$

5.3.3 Realtime Event Monitoring

Events such as car crashes and shooting incidents attract attention easily, especially if they happen in urban areas. In areas where Twitter is popular, such events usually trigger multiple immediate observation reports posted to Twitter. By aggregating these reports, we can potentially detect the event and raise early alarms.

The final module of SNAF focuses on aggregation-based event detection, which uses the location inference described above to find out the correct location of the event. The aggregator is based on the connected component of the graph theory. In our approach, a connected component connects reports that are geographically close to each other. Reports and their locations are added to the graph one-by-one. An alarm is raised to indicate that an event is detected, if the connected component after adding a new report now has enough connected nodes. We set 5 as the number of reports needed for raising an alarm. Fig. 5.3 illustrates this process.
The figure indicates observation reports and their geographical locations. (a) Adding report $A$. (b) $A$ is connected to the nearest report. The connected component now has five nodes, so an alarm is raised, with location of the event calculated as the mean location of nodes $A$, $B$, $C$, $D$, and $E$.

The relevant information such as all reporting tweets for the event are written to a log file when an alarm is raised, which can be then exported to an online platform such as a website, making detected events publicly accessible in realtime. To maintain the timeliness of the monitoring system, we only store the most recent tweets, and reports older than a day are removed. Algorithm 12 shows details of our method.

To monitor an event, we incrementally added the detected reports from Twitter data stream to a graph (line 2). For each added report, we calculated the distance between the new report to the nearest report in the graph, and if the distance is smaller than a threshold, an edge is added between the two reports (line 3 - 5). Then we check the connected component the new report is connected to (in case no edge is created for the new report, the connected component has only one node), and if the number of nodes in the connect component reaches
Algorithm 12 Event Detection and Alarming

1: initialize an empty graph $G = \{V, E\}$
2: for each incoming report $r$ do
3: find $p \in V$ geographically nearest to $r$
4: if distance from $r$ to $p$ is less than $\theta$ then
5: add $\{r, p\}$ to $E$
6: end if
7: add $r$ to $V$
8: $CC \leftarrow$ the connected component $r$ is connected to
9: $s \leftarrow$ the number of nodes in $CC$
10: if $s = 5$ then
11: compute location as the median of locations of all nodes in $CC$
12: output an alarm, record time, location, and connected nodes to the log file
13: end if
14: end for
15: for once each hour do
16: remove all nodes older than 24 hours from $G$
17: end for

five, we raise an alarm and record the time, location and the reporting tweets (line 8 - 13). We also remove old reports once every 24 hours to keep the information up to date (line 15 - 17).

Our event detection can leverage distance-based connected components because we are confident about the accuracy of our location inference at fine granularity. Since the reports are added to the system individually, SNAF is also capable of processing real-time streams, making it a real-time event detection system.

5.4 Experimental Analysis

To test and validate our approach, we collected two sets of data from Twitter, following *homeless* and *shooting* reports. For each dataset, we used the filter to generate a set of immediate observation reports, then used location inference methods to find out the location of each report. The accuracy of each method based on the comparison between the inferred locations and the GPS data was recorded. The inferred locations were then used in our
event detection system. In this section, we will discuss our experiment setup and results for accuracy tests and event detections.

5.4.1 Datasets

We collected two sets of tweets by monitoring Twitter using the keywords *homeless* and *shooting*, through Twitter’s public stream API\(^5\). The homeless dataset contains around one million tweets dated from August 5 to September 9, 2014. The shooting dataset contains around two million tweets dated from September 24 to October 17, 2014. For simplicity, we choose the Random Forest (RF) classifier described in the previous chapter for filtering the observation reports. As we previously showed, RF has the highest precision in our classification tasks comparing to SVM and Naive Bayes classifiers, though with lower recall due to its rule-based structure. Testing using 400 manually-labeled examples, we found that the RF classifier provided 0.65 precision and 0.55 recall for the homeless dataset, and 0.83 precision and 0.25 recall for the shooting dataset, as shown in Table 5.4.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>homeless dataset</td>
<td>0.65</td>
<td>0.55</td>
<td>0.60</td>
</tr>
<tr>
<td>shooting dataset</td>
<td>0.83</td>
<td>0.25</td>
<td>0.39</td>
</tr>
</tbody>
</table>

Filtering the homeless dataset generated 20,229 reports, 1,696 of which contained GPS data. Filtering the shooting dataset generated 10,216 reports, 359 of which contained GPS data. For each report, we collected the timeline of the user who made the report, using Twitter’s timeline API, unless the report itself contains location information.

---

\(^5\)https://dev.twitter.com/streaming/public
5.4 Experimental Analysis

5.4.2 Measurements and Baseline Methods

We set five distance error buckets of 3, 5, 10, 30, and 100 kilometers, and counted an inferred location in a bucket if the difference between the inferred location and the GPS data of the report was within that error range. If the error of the inferred location was more than 100 kilometers, we counted it as failed. The accuracy of a method in an error range is calculated as the percentage of reports counted in that distance error bucket, in all reports considered. If no location information was found for a report, we still counted it as failed. Since we count all the reports, the accuracy we use here is the same as recall in information retrieval. Because we can extract a location from most of the reports, the precision would be very close to the recall, and is not included in the measurement.

In addition to DK-Outlier removal (DKO), NK-Outlier removal (NKO), and their combination with burst compression (DKO+BC and NKO+BC), as defined in the previous section, we also tested two baseline methods for comparison, the median cleaning method, and the method proposed by Ikawa et al. in [128], which we term Ikawa method. The median method, which uses the median of all past locations as the tweet location, is a popular data cleaning method in sensor networks. The Ikawa method associates messages with locations using messages containing location words, and then infers the location of a new message by matching the message with the trained messages, based on cosine similarity and term frequency. The Ikawa method has two variations, trained by all data, and trained by user. We chose the trained by user variation, because it performs better in smaller error ranges.

5.4.3 Location Accuracy Results

We applied the location inference methods on 1,696 homeless reports and 359 shooting reports, and generated a location for each report. Fig. 5.4 shows some of the locations identified around North America, with dense areas roughly corresponding to large cities that have dense populations and high Twitter usage.
The accuracy of the proposed inference methods for each of the two datasets is shown in Table 5.5. The highest accuracy in each distance error bucket is highlighted in bold font. From the table, we can see that for the homeless dataset, NKO achieved the highest accuracy in smaller error ranges, while NKO+BC achieved higher accuracy in larger error ranges. For the shooting dataset, NKO+BC achieved the highest accuracy in all error ranges.

Compared to the median method, our methods, which essentially cleaned the data before applying median, successfully improved the accuracy by 5% to 20% across different error ranges. Our best achieving method was also more accurate than the Ikawa method in all error ranges. The accuracy of the Ikawa method was lower in smaller error ranges than the numbers reported in their paper, because the Foursquare data, which they relied on to get accurate location extraction, are no longer available, and our automatically generated gazetteer based on DBpedia was used instead. The accuracy of Ikawa method in larger error ranges was nevertheless higher than the numbers reported in their paper, because we extracted locations from more messages using the gazetteer approach, thus allowed the locations of more tweets to be found, even though the errors are larger.
5.4.4 Event Detection Results

In this experiment, we tested the event detection of the SNAF system using the *Shooting* dataset. In particular, we sorted 10,216 reports in chronological order, sent each report to SNAF, from the oldest to the newest, and SNAF generated a number of alarms, using NKO+BC location inference. We manually examined 100 alarms, and found that 54 of them contained reports of actual shooting incidents. Using the same dataset, the Median method only generated 45 alarms containing true reports out of 100. Table 5.6 shows three instances of the detected shooting incidence compared to corresponding news articles.

For each instance, the *time alarmed* is the time of the last report that triggered the alarm. The *earliest news article* is the earliest online news article about the event we found using our best effort. We found these news articles mostly on *Google News*\(^6\), making use of their “sort by date” feature. The *news article time* is the time of the publication of the news. The *time before news article* indicates how much faster our alarm was than the publication of the earliest news article, measured in hours. A positive number means the alarm is faster than

\(^6\)http://news.google.com
Table 5.6 Examples of detected events

<table>
<thead>
<tr>
<th>Detected event: Indiana State University Shooting, 27/09/14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time alarmed</td>
</tr>
<tr>
<td>Location inferred</td>
</tr>
<tr>
<td>News article time</td>
</tr>
<tr>
<td>News article location</td>
</tr>
<tr>
<td>Time before news article</td>
</tr>
<tr>
<td>Location error</td>
</tr>
<tr>
<td>Reporting Tweets</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detected event: Fern Creek High School Shooting, 30/09/14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time alarmed</td>
</tr>
<tr>
<td>Location inferred</td>
</tr>
<tr>
<td>Earliest News article</td>
</tr>
<tr>
<td>News article time</td>
</tr>
<tr>
<td>News article location</td>
</tr>
<tr>
<td>Time before news article</td>
</tr>
<tr>
<td>Location error</td>
</tr>
<tr>
<td>Reporting Tweets</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detected event: Marysville Shooting, 15/10/14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time alarmed</td>
</tr>
<tr>
<td>Location inferred</td>
</tr>
<tr>
<td>News article time</td>
</tr>
<tr>
<td>News article location</td>
</tr>
<tr>
<td>Time before news article</td>
</tr>
<tr>
<td>Location error</td>
</tr>
<tr>
<td>Reporting Tweets</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
news, while a negative one means the alarm is slower. All times are converted to the local time where the event happened. Supported tweets shows tweet messages of the reports in the connected component that triggered the alarm.

In the Indiana State University Shooting case, although the news reacted quickly and reported less than one hour and a half after the incident, our system was able to capture the incident 9 minutes earlier than the news. From the supporting tweets we also identified a mistake in the news, that the incident happened not at 6:30, but at 16:30. In the Fern Creek High School Shooting case, although we were able to detect it just one hour after the shooting, it was half an hour slower than the news. In the third instance, the Marysville Shooting on the night of October 15, 2014, which happened around midnight, was not reported by the news until next morning. However, our monitoring system was able to detect it at mid-night, only an hour after the shooting. In all three instances, the event locations the system inferred were very close to the reported location, with distance errors between a few hundred meters and 1.5 kilometers. The inferred location for the last event was less accurate because the event involved a series of sub-events each had a different location.

5.5 A Prototype Realtime Event Monitoring System

We design and implement a prototype realtime event monitoring system for monitoring shooting incidents, called SNAF-Shooting, based on the methods and techniques proposed in this article. We implement the components shown in Fig. 5.1 using Java 1.7. The Java program continuously listens to Twitter Filter API, and upon capturing a tweet containing the keyword shooting, it classifies, infers the location, and aggregates with previous tweets, and outputs an event record if an event alarm is raised. Except for the final event information output, the program runs entirely in memory. We test run the program on a computer with a pair of Intel Core 2 Duo CPU, 7.9 GiB Memory, 621 GiB hard-drive, and runs a 32-bit
Linux-based operating system. Setting 1024 MB memory for the Java Virtual Machine is proven to be sufficient for running the program continuously over a week.

We then setup a webpage that contains PHP code that reads the event record file and displays them on the webpage. The PHP code converts the coordinates of the inferred event location to pixel values, and display a marker on top of a map shown on the webpage, as can be seen in Fig. 5.5. The numbers above the markers indicates event numbers, and the details of each event, including time, exact location, and reporting tweets, can be found below the map, as shown in Fig. 5.6. We put a “Report Faulty Alarm” button alongside each event to allow the user to provide feedback for faulty event alarms. Each time the webpage is reloaded, the latest information on detected shooting incidents will be displayed on the webpage.
We set 5 as the number of required tweets for raising an event alarm, so the time needed for event detection after the event happened depending on the time of the emergence of the 5th report, and may vary between a few minutes and several hours. Classifying, inferring location, and aggregation for a single tweet, though, generally takes around 10 seconds. Most of this time is spent on waiting on Twitter API’s rate limit. Twitter sets a rate limit of 180 calls in 15 minutes, or five seconds per call, for its timeline API. Each timeline API allows retrieval of up to 200 past tweets. We make two calls to retrieved 400 past tweets for location inference of each tweet, and we set a delay of four seconds between each call, assuming that the location inference itself generally takes more than one second to complete. Actual statistics shows that, for 1,000 location inferences, the average time taken for each inference
is 10.8 seconds. The time needed for classifying and aggregation is negligible, compared to
the time needed location inference.

5.6 Discussion

The event monitoring system we demonstrate shows how fine-grained locations can be used
in event detection and location information extraction. This system can be expanded to
monitor various events other than shooting incidents, such as car crashes, sewer leakages,
or traffic jams. Implementing the system for monitoring a new type of events only requires
constructing a new report filter, while the location inference and aggregation components
can remain the same. Sometimes it is difficult to pick a keyword for monitoring an event,
for example air pollution, as people usually do not say “air pollution” in their tweets, but
rather “dirty air”, “bad air”, or “smog”. In such cases, better keyword choice strategies are
desired, and remains a future study topic. Existing works proposing automatic query keyword
generation include [179] and [10].

SNAF currently assumes that there is only one location for the monitored events. However,
in many real-world cases, events change their locations over time, or occur in multiple
locations at the same time. For example, a shooter may be running through an area and
causing shooting in a series of locations. In a public demonstration, the police may clash with
protesters, at several locations at the same time. In such cases, providing a single location
would not be accurate. Recognizing if an event has multiple locations, and inferring these
locations accordingly, remains a future study topic. Unankard et al. [96] proposed to divide
an event into sub-events based on geo-location. Although their focus is on national-level
events such as elections, they provided interesting insights on location-based sub-events.
Lee et al. [123] investigated the evolution of topic discussion on social networks, which
also provides some interesting insights for evolving events that potentially involve location
changes.
5.7 Summary

To fully convert observation messages on Twitter to sensory values, we need associated time and location information. While time information is readily available for every tweet, location information, unfortunately, is sparse on Twitter. Current works proposed to infer locations from tweets usually focus on coarse-grained locations, and generally generate large error when attempting to infer fine-grained locations. Apart from the scarce GPS data, we find the tweet messages that contain location names can be used as a location information source. In addition, we find that the past locations of a user can be a good indicator for the user’s current location. Consequently, we propose methods to extract locations from tweet messages and inferring the current location based on the user’s past locations. Our inference is based on sensor data cleaning techniques. We test our approach with extensive experiments, and show that our methods significantly improve the inference accuracy at fine-grained levels, with up to 16% within the 3km error range, and 20% within 5km error range.

Based on the inferred locations, we propose Sense and Focus (SNAF), an event monitoring system that detects and localizes emerging events reported on Twitter. Taking shooting incidents as the target event, our prototype event monitoring system captures local events and provides accurate location information, in many cases with less than 1km error. The events detected by our system were actual event of interest more than half of the time, and in many cases faster than the earliest news reports.
Chapter 6

Conclusion

6.1 Thesis Summary

To summarize, we revisit the research objectives stated in Chapter 1. These objectives are accomplished as the following.

6.1.1 Insights of Current Situation

Our first objective was to provide insights of current situation of IoT-inspired environmental sensing and social sensing in smart cities. This goal was reached in two aspects, first in the literature review, then in the experiments with real world data.

For environmental sensing, we studied the causes behind sensor faults present in existing literatures, such as chemical compound decay, battery exhaustion, and calibration problems [13]. Existing works also reveal faulty data patterns such as spikes, intense spikes, and high volatility. We also studied reports from specific sensing projects that reveal data problems, including the air pollution monitoring project in Zurich, and the deployment of the sensor node TASK [40, 18]. These studies provide an overview of the faulty data problem present in current sensing projects. In our experimental analysis, we looked at actual data from real
world sensing projects, such as the Intel Lab Data, the OpenSense Zurich air quality sensing data, and the Melbourne Weather environmental sensing data. We found faulty data patterns of different degree in data produced by these projects. Therefore our study provides insights into data problems in environmental sensing with different levels of details, from high level generalizations to specific data faults in a particular sensing project.

For social sensing, we discuss various studies of social network messaging, including message types [16], user roles [159], and message credibility [100]. These studies reveal the level of noises and related user behavior with regard to collecting observation messages posted on microblogging services. Our analysis showed that the proportion of observation messages in all microblogging message posted is small, and that particular types of users, such as business users and news agencies, are less likely to post personal observations. We next looked at actual data we collected using Twitter API. By examining messages containing object or event keywords such as hailstorm, accident, and shooting, we verified that the noise level is very high with regard to observation collection. Through these studies on literature and actual data, we gained a deep understanding of challenges in opportunistic social sensing, including message filtering, content analysis, and user profiling.

6.1.2 Developing Data Cleaning Techniques

Our second objective was to develop data cleaning techniques for improving the accuracy of environmental sensing and social sensing data. This goal is reached by our proposed data cleaning techniques.

For environmental sensing data, we first proposed a technique that iteratively calculates sensor reliabilities using a frequentist approach, call Influence Mean. Assuming working sensor will generate similar readings, this technique compares each sensor reading with the average reading, before deciding whether the reading is correct, and adjusting the sensor reliability accordingly. Two separate options called IM-Reduce and IM-Remove can be
used to weight down or remove the readings from faulty sensors. Then we proposed a technique that models the sensor reliability as a latent variable using a Bayesian approach. This *Expectation Maximization*-based technique models environmental features as a linear model, and by maximizing the likelihood of the model, the latent sensor reliability states can be calculated, as well as the correct reading values. We verify the effectiveness of the proposed technique using real world data. Our extensive experiments show that the proposed techniques significantly improve the accuracy of correct reading prediction. In particular, in a temperature dataset where working and faulty sensors behave very different, the IM-Reduce and EM technique achieve almost no prediction error, while all other baseline methods produce errors over 30 degrees Celsius.

For social sensing data, we first propose three techniques for observation message classification. The first technique distinguishes messages using their *Perspectives*, including observation, affection, and speculation. It deploys supervised machine learning models incorporating a number of lexical and textual features. The second technique deploys supervised machine learning models incorporating *User Features*, including trending activity, communication status, and writing role. Then we propose an unsupervised technique based on *User Profiling*. This technique identifies personal observations through lexical analysis and personal user classification that is based on four statistic attributes including objectivity, interactivity originality, and topic focus. We test the effectiveness of these techniques using real Twitter data. Comparing to a number of existing observation classification approaches, the experiments show that each technique is capable of improving observation classification to some degree. In particular, for two datasets containing hailstorm and accident observations, the unsupervised method consistently improves observation classification accuracy by around 22%, significantly out-performs popular classifiers proposed by Sakaki and Sriram.

We also proposed a technique to fill missing location data in observation messages. This technique, as a part of SNAF, our proposed event detection system, extracts placenames
in users’ past messages using a generated gazetteer, and used the extracted placenames to infer the message location. It incorporates outlier removal techniques to clean the location data. By using this technique, 87% messages can be associated with a location data. The inferred location is also more accurate than existing location inference approaches, with improvements of 16% in 3km error range, and 20% in 5km error range.

6.1.3 Applying Cleaned Data

Our third objective was to show the potential of using better quality data in various applications. By identifying the faulty sensor, we can immediately improve the correct reading prediction accuracy of the environmental feature, which is useful in many applications such as city infrastructures planning and pollution monitoring. In social sensing, improving the classification and location inference accuracy of the observation messages, similarly, can be useful in many subsequent applications, such as public opinion estimation and rumor detection.

We show the usefulness of the improved data in one particular application, called Sense and Focus (SNAF), which is an object and event monitoring system with accurate location information. SNAF aggregates classified observation messages and inferred locations in a connected graph for detecting events, and its success depends heavily on the observation classification and location inference accuracy. Running a prototype implemented using Java and R, we detected a number of events with SNAF, 52% of which are actual events in interest. Comparing to news media reports, the events detected by SNAF can often appear earlier than the earliest news report, while the location inferred is very close to the location reported in the news. Without improving the observation classification and location inference accuracy with the proposed techniques, though, the correct event detection rate is significantly lower. With SNAF, we achieved showing the potential of better quality data.
6.2 Limitations

This thesis proposes several data analysis techniques for environmental sensing and social sensing data. In practical applications, however, various limitations impose on these techniques due to certain assumptions and data requirements. We discuss these limitations below.

6.2.1 Unreliable Sensor Behavior Assumption

The primary assumption for our reliability-based environmental sensing data cleaning techniques is that reliable sensors behave similarly, and faulty sensors behave very differently. Our observation show that this is generally true. For example, hardware failure and battery weakening can cause two common faulty data patterns, high volatility and occasional spikes [11]. Both types of faulty data will deviate significantly from mean values, if the majority of the data are close to true values. However, there are some cases in which faulty data also have similar behavior to each other. For example, certain hardware failure and connection problems can cause “stuck-at” faulty data pattern, which are readings being kept at a fixed value for a period of time. In our observation of real world data, we found that “stuck-at-zero”, where sensors produce no other reading than zero for a period of time due to some failure, is actually quite common. In this case, all faulty sensors producing “stuck-at-zero” will behave exactly the same, thus violate our primary assumption. To mitigate this, special care for the “stuck-at” data pattern may need to be placed, by, for example, using their no-variance characteristic [11] that are unlikely to be true.

6.2.2 Consistent User Feature Assumption

The primary assumption of our user-profiling-based social sensing data analysis techniques is that user features are consistent over a period of time. This assumption is used by our
approach of taking and analyzing a user’s past messages to generate knowledge of a current state. Using common sense, this assumption can hold true to some degree. A person’s interest, job, and home location generally remain the same during a period of one or two years. However, changes in life do happen. A journalist may quite her job in a newspaper and become a freelancer, and the interest, focused topics, and writing style of her Twitter account can significantly change. In this case, collecting her past messages and considering that they can be good indicators for determining the state of her most recent message would most likely lead to a mistake. A person may also relocate from one city to another, in which case their past locations can longer reflect their current location. Currently, our techniques do not consider such cases.

6.2.3 Multiple Sensing Source Requirement

Our sensing data analysis techniques require multiple sensors observing the same environmental feature. Most IoT sensing projects deploy multiple sensors, usually installed on vehicles and thus mobile. Depending on the environmental feature, how close they need to be to be considered as observing the same feature can vary. Environmental features such as temperature and humidity may stay approximately the same across several kilometers in an open area. Sounds, on the other hand, usually diminish within a few hundred meters. If the vehicle the sensors are being installed on do not have a pre-defined routine (i.e., taxis), whether a few sensors can get close to each other may also remain a matter of chance. The environmental feature to be monitored, the type of mobility, and the total number of sensors deployed in the project may determine whether our technique is applicable. Although given the rise of social sensing and crowd sensing, we will only see sensor number increasing, which gives our techniques an advantage.
6.3 Extending Results

6.2.4 Access to Message History Requirement

Our social sensing method relies on user message histories for generating user profiles. In most microblogging and social network services, access to user message histories is not fully provided. Twitter\(^1\) allows public access to a user’s message history up to 3,200 messages. Instagram\(^2\) only allows access to a user’s most recent post. In Facebook\(^3\), access to user message histories is subject to its application review. Furthermore, due to privacy concerns, most of these services allow users to control whether their message histories can be shared. Thus using message histories limits the applicability of our techniques. One possible mitigation is to use realtime monitoring. Most of these social network services allows realtime monitoring of all their data traffics, and by monitoring the message activities of certain users, we can generate user past messages without using past message API, albeit only for the period of monitoring.

6.3 Extending Results

The studies and techniques presented in this thesis mainly focus on the domain of data mining and knowledge extraction, particularly in the data analysis for smart city sensing data. The solutions we developed, however, may also be applicable to problems in other domains. In this section, we discuss the possible extension of our solutions in domains such as data integration and information retrieval.

6.3.1 Data Integration with Unreliable Sources

In database studies, a challenging problem that is currently attracting research efforts is data integration with unreliable sources that provide conflicting views [180]. For example,
different movie databases may show different actors for a same movie [152], or different bookstore websites may show different authors for a same book [150]. The underlying problem is to detect the poor quality sources and uncover the correct data.

The environmental sensing data cleaning solutions proposed in thesis are applicable to such a problem, as it generally satisfies the assumption and requirement of our solutions. First, it is generally true in such data integration problems that reliable sources provide the same data, while unreliable sources provide different data. Second, the data for a piece of knowledge can generally be found in multiple sources. Therefore it is quite clear that the source reliability calculation techniques present in this thesis, including the incremental reliability update method and the Expectation Maximization method, are also applicable to data integration problems with conflicting sources. To apply, one only needs to find a way to define a mean value for the data and a distance function, so that data from difference sources can be compared to the mean value, which forms the core of our techniques.

6.3.2 Online Rumor Detection

With the increasing popularity of social media, a number of recent studies in information retrieval have focused on the problem of online rumor detection and early prevention [181, 182]. Rumors are unconfirmed stories that could incite public panic and mistrust once wide-spread. For example, Maddock et al. identified four rumors created after the Boston Marathon Bombing incident in 2013 that gained a strong public influence. Current rumor detection approaches including supervised machine learning methods [183], propagation-based methods [184], and statistical methods [185].

In this thesis, we propose a message classification solution that classifies message according to their perspectives, including the perspective of speculation. A speculation is a user’s subjective judgement of an ongoing event that is unconfirmed to be true or otherwise. Classifying speculation can be considered as the first step to rumor detection. To further
6.4 Future Works

In this thesis, we presented a number of techniques for improving the accuracy of noisy and unreliable sensing data. These techniques have been validated with extensive experiments and shown significant improvement compared to existing approaches. In the following, I will discuss some possible directions for further data quality improvement that might be achieved in future works.

6.4.1 Sensor Fault Detection with Multiple Features

Sensor data cleaning techniques presented in this thesis assume each sensor monitors a single environmental feature. In many sensing projects, however, a sensor node combines an array of sensors that monitors multiple environmental features. For example, the sensor node proposed in [75] contains sensors for CO, NO2, SO2 in a single-chip microcontroller. This will lead to some interesting sensor fault states. For example, battery failure and connection problems will cause readings from all sensors to become faulty, while chemical compound decay will only cause one particular sensor to fail. A sensor fault detection technique may take into consider such situations and provide one more layer of analysis that may further improve fault detection accuracy and data quality.
6.4.2 Improving Social Sensing by Exploiting Friend Network

Our social sensing analysis techniques focus on messages, whether a single message or a list of messages from a user. Friend networks, another type of information on social networks could be explored. In most social network services, users may relate to each other through “friending” activities. In Facebook, two users can become friends using the friending function. In Twitter, a user can follow or be followed by another user. The state of friend network is generally available from APIs provided by these social network service. As several works have identified, such friend network may reveal interesting information about the users that are hidden at first [186, 112]. This friend network may also be useful in observation classification and location inference. For instance, a user’s location, which is otherwise unobtainable, may be inferred based the location of her friends.

6.4.3 Mash-up of Environmental Sensing and Social Sensing

In this thesis, we studied IoT-inspired environmental sensing and social sensing, mostly as two separate sensing data sources. It is possible, however, to combine these two sources in one application. For example, suppose that we are monitoring shooting incidents in urban areas, an instance of shooting may trigger a number of messages in microblogs, as well as abnormal readings in nearby noise sensors. By cross-validating between the social sensing and the environmental sensing, it is possible to improve the data accuracy from both sources, as well as the prediction of the real-world phenomenon. Another mash-up maybe using air quality sensors and tweet messages about “smog” or “haze” to accurately locate urban pollutions. The techniques presented in thesis, which convert microblog messages to sensory values and fill the missing location data, have laid the foundation for such mash-up to be achieved. So far we have not found a particular pair of datasets from environmental sensing and social sensing that can be presented in an interesting mash-up. However, given the increasing usage of sensors and microblogs, we should see more and more opportunities for
such mash-up in the future, which will allow another group of interesting techniques and applications to emerge.
References


[150] Xiaoxin Yin, Jiawei Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. IEEE Transactions on Knowledge and Data Engineering, 20(6):796–808, 2008.


165


Appendix A

Curriculum Vitae
Yihong Zhang
PhD Candidate (expected completion: Q1 2016)
School of Computer Science
The University of Adelaide, Australia
Tel: +61 8 83134487 yihong.zhang@adelaide.edu.au
Mobile: +61 403128401 http://cs.adelaide.edu.au/~yihong

RESEARCH INTERESTS
My research interests include large-scale sensing data analysis, the Internet-of-Things, social network-based object monitoring and event detection, Bayesian prediction techniques, and cloud computing.

EDUCATION
PhD in Computer Science The University of Adelaide February 2013 - Present
Supervisors: Prof. Michael Sheng and Dr. Claudia Szabo

M.S. in Computer Science The University of Adelaide November 2012

B.S. in Computer Science The University of New South Wales November 2010

JOURNAL PUBLICATIONS


CONFERENCE PUBLICATIONS

Ali Shemshadi, Lina Yao, Yongrui Qin, Quan Z. Sheng, and Yihong Zhang, 2015: ECS A Framework for Diversified and Relevant Search in the Internet of Things. The 16th International Conference on Web Information Systems Engineering WISE’15. (CORE Ranking A)
Yihong Zhang, Claudia Szabo, and Quan Z. Sheng, 2015: Sense and Focus: Towards Effective Location Inference and Event Detection on Twitter. *The 16th International Conference on Web Information Systems Engineering WISE’15.* (CORE Ranking A)

Yihong Zhang, Claudia Szabo, Quan Z. Sheng, and Xiu Susie Fang, 2015: Classifying Perspectives on Twitter: Immediate Observation, Affection, and Speculation. *The 16th International Conference on Web Information Systems Engineering WISE’15.* (CORE Ranking A)

Yihong Zhang, Claudia Szabo, and Quan Z. Sheng, 2014: Cleaning Environmental Sensing Data Streams based on Individual Sensor Reliability. *The 15th International Conference on Web Information Systems Engineering WISE’14.* (CORE Ranking A)

**WORKS UNDER SUBMISSION**


Yihong Zhang, Claudia Szabo, and Quan Z. Sheng: Effectiveness of User Features in Supervised Observation Classification on Twitter. Submitted to the *39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR’16.*

Yihong Zhang, Claudia Szabo, and Quan Z. Sheng: Extreme User and Political Rumor Detection on Twitter. Submitted to the *39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR’16.*

**RESEARCH EXPERIENCE**

**SNAF: An Object and Event Monitoring System for Twitter**  
2014-2015  
Described and implemented SNAF, a system for location inference and event detection for Twitter. The implemented Java package includes facilities to collect, clean, analyze, and generate knowledge from Twitter data.

**Microblog Message Filtering**  
2014-2015  
Conducted a study of supervised and unsupervised methods for classifying microblog messages. Designed several novel features and a user model. Various machine learning methods such as SVM, LDA, Random Forest, and Naïve Bayes were studied.
Rumor Detection on Twitter 2015
Conducted a study of Twitter-based rumor detection approach. Proposed an extreme user model for identifying political rumors. Experiments with real Twitter data show high detection accuracy.

Bayesian Sensor Data Cleaning 2015
Designed and implemented an Expectation Maximization framework for finding faulty sensors and correct sensing values for sensor network. Smart city datasets such as Melbourne City Weather and Zurich OpenSense were experimented to show accuracy improvement.

Realtime Sensor Data Cleaning 2013-2014
Designed and implemented an incremental algorithm for finding sensor reliabilities and correct sensing values for sensor network. Synthetic pollution data and simulated mobile sensor readings were generated using R. Spike, intense spike, and high variance data faults were studied.

Scientific Workflow Scheduling on Cloud 2012
Conducted a study of scheduling scientific workflow on Amazon EC2 cloud. Executed scientific workflows such as Montage using Condo and Pegasus, and analyzed the runtime and costs.

MENTORING AND TEACHING

Tutoring: Specialized Programming, Semester 1 and 2, 2014

SERVICES AND OTHER ACTIVITIES

Invited Reviewer: IEEE Transaction on Big Data
Presentation: WISE 2014, WISE 2015, ICPADS 2015
Peer Mentor: International Student Peer Mentor in 2014
Member: IEEE, ACM

AWARDS AND HONOURS

The University of Adelaide Faculty of ECMS PhD Scholarship 2013

PERSONAL INFORMATION

Gender: Male
Nationality: Australia
Language: English, Mandarin Chinese, Japanese