

© 2017 by Shiguang Wang.

ON INFORMATION FILTERING IN SOCIAL SENSING

BY

SHIGUANG WANG

DISSERTATION

Submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Computer Science  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:

Professor Tarek F. Abdelzaher, Chair  
Professor Marco Caccamo  
Professor Jiawei Han  
Doctor Lance Kaplan, US Army Research Labs

# Abstract

For decades, from the invention of Sensor Networks, people envisioned a global sensing platform with millions of sensors deployed globally. The platform has finally become real recently with the advent of multiple online social network services where humans act as sensors and the social networks act as sensor networks, a practice named *Social Sensing*. Social sensing was born with the advances of high-level semantics sensing (since humans are the “sensors” with texts or photos as the sensing data) and (almost) zero-cost real-time data infrastructure, which makes this new sensing paradigm very promising in multiple real-world applications including disaster response and global event discovery. However, its global scale results in a massive amount of data generated and collected in applications that far exceeds normal people’s cognitive capability of information consumption, thus we desire a system that can filter the massive sensing data and delivers only information and intelligence to the users with a human-consumable amount.

In this thesis, I focus on designing an information filtering system for social sensing; specifically, I focus on three levels of information filtering. In the first level, we focus on untruthful information removal, also known as fact-finding, where the challenge lies in the unknown reliability of each individual social sensor (i.e. human) *a priori*. In the second level, we focus on event-level information summary, also known as event detection, where the challenge lies in *de-multiplexing* different event instances and fusing social events detected in multiple social networks that previous approaches do not perform well. In the third level, we focus on information-maximizing data delivery to social sensing users, especially on redundancy removal by diversifying the information feed, where the challenge lies in algorithm design that not only works well empirically but also has a theoretical performance guarantee. We address the above challenges by algorithm design and system implementation and real-world data evaluations verify the efficiency of our proposed solutions.

*To my father Zhe Wang, and my mother Yaqin Wang*

# Acknowledgments

First and foremost, I would like to sincerely thank my thesis advisor, Professor Tarek F. Abdelzaher, for his great guidance, continuous support, and persistent encouragement through my PhD study. This thesis would not even be possible without him. It was his inspiring guidance and discussions that led me through the challenges that I encountered. Tarek also encouraged me to be a leader not a follower, and himself is an excellent example of a leader. The thesis topic, social sensing, was actually proposed from him! He also provided lots of opportunities to practice my leadership, such as conference presentations and supervision master students. I have grown tremendously with the practices and become more self-confident being a leader. He will be my lifetime role model.

Next, I would like to thank Professor Jiawei Han, Professor Marco Caccamo, and Dr. Lance Kaplan for serving in my PhD thesis committee. It is my honor to have chances to work with such great scholars and researchers. Their constructive suggestions and comments improved this thesis significantly. I would like to specially thank Dr. Lance Kaplan for his valuable feedbacks on many of my papers.

I would also like to extend my gratitude to the professors, researchers, and colleagues who helped me on this thesis. They are Professor Hengchang Liu, Professor Lu Su, Professor Dong Wang, Yong Yang, Shen Li, Shaohan Hu, Tanvir Amin, Yunlong Gao, Hongwei Wang, Prasanna Giridar, Yiran Zhao, Shuochao Yao, Huajia Shao.

I would also like to thank Philips Research North America, and Facebook for providing three awesome internships. These internships help me taste the flavor of industrial research and engineering, and finally convinced me to go to industry to apply my knowledge making real-world impact.

Finally, and most importantly, I would like to thank my family. My father Zhe Wang and my mother Yaqin Wang brought me to the earth and taught me how to be a man. They always stand

by my side and quietly support me both emotionally and moneytarily. My wife Zongjie Zhang gave up her everything in China, friends, relatives and her family, to accompany with me in the US. She has been taking care of my everything in life during my PhD without a single word of complaint. I deeply appreciate all her contributions to my family. And my sweet daughter Annie Jiani Wang and my bright son Aiden Haolin Wang just brought sunshine to my pale PhD life. Playing with them were simply fun. My family is my life-long motivation of fighting.

# Table of Contents

<b>List of Tables</b> . . . . .	<b>ix</b>
<b>List of Figures</b> . . . . .	<b>x</b>
<b>Chapter 1 Introduction</b> . . . . .	<b>1</b>
1.1 Motivation and Challenges . . . . .	2
1.2 Thesis Overview . . . . .	4
1.2.1 Untruthful Data Removal . . . . .	4
1.2.2 Event-level Data Summarization . . . . .	5
1.2.3 Information-Maximizing Data Delivery . . . . .	6
1.3 Thesis Organization . . . . .	7
<b>I Untruthful Information Removal</b> . . . . .	<b>8</b>
<b>Chapter 2 Fact-finding with Time-Varying System State</b> . . . . .	<b>9</b>
2.1 Problem Formulation . . . . .	11
2.2 Computing Trajectory Probabilities . . . . .	15
2.2.1 Independent State Change . . . . .	15
2.2.2 Markov Model . . . . .	15
2.3 Dynamic State Estimation . . . . .	16
2.3.1 Deriving a Crowd-sensing State Trajectory Estimator . . . . .	16
2.3.2 The EM-VTC Algorithm . . . . .	18
2.4 Accuracy Guarantees . . . . .	18
2.4.1 Confidence Interval of Source Reliability . . . . .	18
2.5 Multivalued Variable Extension . . . . .	20
2.6 Evaluation . . . . .	22
2.6.1 Simulation Study . . . . .	22
2.6.2 A Real-world Case Study . . . . .	27
2.7 Related Work . . . . .	30
2.8 Conclusions . . . . .	32
2.9 Derivations . . . . .	33
2.9.1 Deriving the E-step . . . . .	33
2.9.2 Deriving the M-step . . . . .	34
2.9.3 Deriving Cramer-Rao Lower Bound . . . . .	34

<b>Chapter 3</b>	<b>Fact-finding with Interdependent Variables</b>	<b>37</b>
3.1	Problem Formulation	39
3.1.1	Modeling Interdependent Event Variables	40
3.1.2	Categorized Source Reliability	41
3.1.3	Problem Definition	42
3.2	Generalization of Previous Models	44
3.3	Estimating the States of Interdependent Variables	46
3.3.1	Defining Estimator Parameters and Likelihood Function	46
3.3.2	The EM Algorithm	49
3.4	Evaluation	49
3.5	Related Work	57
3.6	Conclusion	58
<b>II</b>	<b>Event-level Data Summarization</b>	<b>59</b>
<b>Chapter 4</b>	<b>Event Detection and Demultiplexing in Social Spaces</b>	<b>60</b>
4.1	Problem Statement	63
4.2	The Design of StoryLine	64
4.2.1	Design Intuitions	65
4.2.2	Discriminative Keyword Pair Selection	67
4.2.3	The Consolidation Algorithm	69
4.2.4	Event Tracking	70
4.3	System Implementation	71
4.4	Evaluation	72
4.4.1	Twitter Datasets	72
4.4.2	Event Signature Consolidation	73
4.4.3	Event Demultiplexing	74
4.4.4	Case Study – Real-time Earthquake Detection	77
4.4.5	Case Study – Nepal Earthquake Tracking	78
4.5	Related Work	79
4.6	Conclusions	82
<b>Chapter 5</b>	<b>Event Tracking by Integrating Twitter and Instagram</b>	<b>84</b>
5.1	Problem Definition and System Design	85
5.1.1	Problem Formulation	85
5.1.2	System Architecture and Design	86
5.1.3	Event Detection	87
5.1.4	Event Fusion	88
5.1.5	Event Tracking and Summary	89
5.2	Evaluation	90
5.2.1	Datasets from Twitter and Instagram	90
5.2.2	Methodology and Results	90
5.3	Related Work	93
5.3.1	Event Detection using Twitter	94
5.3.2	Event Detection using Instagram	95
5.4	Conclusions	96



<b>III</b>	<b>Information-Maximizing Delivery</b>	<b>97</b>
<b>Chapter 6</b>	<b>Coverage-based Information-Maximizing Data Delivery</b>	<b>98</b>
6.1	State of the Art	100
6.2	System Design	101
6.2.1	System Model	101
6.2.2	Programming Framework	102
6.3	Information-maximizing Prioritization	103
6.3.1	Information Coverage	103
6.3.2	Problem Definition	104
6.3.3	Greedy Algorithm	106
6.3.4	Transmission Protocol Design	107
6.4	Evaluation	110
6.4.1	Experimental Setup and Methodology	111
6.4.2	Overhead of Minerva	111
6.4.3	Large-scale Trace-based Evaluation Results	116
6.5	Conclusions	117
<b>Chapter 7</b>	<b>Tree-based Information-Maximizing Data Delivery</b>	<b>118</b>
7.1	The Information Funnel	120
7.1.1	The Basic Abstraction	120
7.1.2	Data Ordering Challenges	121
7.1.3	Optimal Transmission Order	122
7.1.4	An Example	124
7.1.5	Receiver Feedback	125
7.1.6	The Algorithm	125
7.1.7	Variable Object Length and Differentiated Service	127
7.1.8	System Design and Implementation	128
7.2	Evaluation	130
7.2.1	Methodology	130
7.2.2	Evaluation Results	131
7.3	Related Work	134
7.4	Conclusions	135
7.5	Math Proofs	135
<b>Chapter 8</b>	<b>Conclusion and Future Work</b>	<b>139</b>
8.1	Summary	139
8.1.1	Untruthful Information Removal Module	139
8.1.2	Event-level Information Summarization	140
8.1.3	Information-Maximizing Delivery	140
8.2	Future Research Directions	140
<b>References</b>		<b>142</b>

# List of Tables

2.1	Percentage of sample points falling out of the corresponding confidence interval for different estimators. . . . .	27
2.2	Estimation error in parking case study . . . . .	30
3.1	The summary of notations . . . . .	43
4.1	Prevalence of geotags in tweets and events . . . . .	74
4.2	Event detection precision comparison . . . . .	75
4.3	Real-time earthquake detection summary . . . . .	77
4.4	Nepal earthquake tracking summary . . . . .	78
5.1	Statistics of collected datasets . . . . .	91
5.2	Precision and recall . . . . .	91
5.3	Demultiplexing quality and detection redundancy . . . . .	92
5.4	Segment of a tracked event instance of Delhi protest . . . . .	94
6.1	Overhead of Minerva . . . . .	113
7.1	Overhead study results . . . . .	132
7.2	Coverage study results . . . . .	132
7.3	Coverage study with variable data size . . . . .	133

# List of Figures

1.1	Contrast illustration of traditional broadcasting network and social network. . . . .	2
2.1	An example of CPS system with humans in the loop. . . . .	11
2.2	Performance as the number of sources varies. . . . .	23
2.3	Performance as the talkative factor $s_i$ of sources varies. . . . .	23
2.4	Performance as the source reliability $t_i$ varies. . . . .	23
2.5	Performance as the state transit probability varies. . . . .	25
2.6	Variable value estimation performance as the number of history time-slots being considered varies. . . . .	26
2.7	Performance bounds of EM-VTC in terms of confidence intervals. . . . .	26
2.8	The ground truth of the availability of parking spots. . . . .	29
2.9	The performance of variable state estimation. . . . .	29
3.1	Model connections with previous work. . . . .	45
3.2	An illustration of the Bayesian network. . . . .	48
3.3	Performance as the number of sources varies. . . . .	52
3.4	Performance as the source reliability varies. . . . .	52
3.5	Performance as the source talkativeness varies. . . . .	52
3.6	Performance as the number of edges in the Bayesian network varies. . . . .	54
3.7	Performance as the number of event categories varies. . . . .	54
3.8	Performance as the number event variables varies. . . . .	54
3.9	Computation time comparison with fixed node degree. . . . .	56
3.10	Computation time comparison with fixed number of nodes. . . . .	56
4.1	The Social sensing modality and its analogy with physical sensing . . . . .	61
4.2	Illustration of the non-overlapping sliding window and the overlapping sliding window (with 50% overlapping). . . . .	70
4.3	Event tracking system architecture . . . . .	71
4.4	The consolidation error rate. . . . .	73
4.5	The purity pie charts. . . . .	76
4.6	Tweets with location information. . . . .	82
5.1	Event tracking system architecture . . . . .	86
5.2	CDF of empirical Beta expectation comparison . . . . .	86
5.3	Sliding window . . . . .	90
5.4	F1 score comparison with varied ground truth of total number of events . . . . .	92
6.1	System model . . . . .	101

6.2	Programming framework . . . . .	101
6.3	Illustration of coverage and marginal coverage with 2 features. . . . .	101
6.4	Transmission protocol illustration. . . . .	107
6.5	tdrive data used in simulation . . . . .	112
6.6	Performance of Minerva with different coverage intervals and $1/\beta$ values in phone-based experiments with synthetic data. . . . .	114
6.7	Performance of Minerva with different coverage intervals in large-scale real-world GPS trace simulations. . . . .	115
7.1	An example . . . . .	124
7.2	Information funnel structure . . . . .	128
7.3	Performance of information funnel. . . . .	133

# Chapter 1

## Introduction

*Social Sensing* is a novel sensing paradigm using humans as sensors and using social networks as sensor networks [130]. The proliferation of mobile computing devices in the possession of the average individual and the omnipresent Internet access via 3G/4G/LTE, as well as the globally covered online social network services, like Twitter and Instagram, have granted people the ability and freedom to post anything in anywhere and anytime. The social media posts about world physical events can be viewed as “sensing data”, and the people who posted them actually act as “sensors”, which derives the new sensing paradigm. Thus, the social networks can be viewed as the global sensor networks.

Compared with traditional sensing paradigm with physical sensor nodes, social sensing that directly exploits humans as sensors advances in two main aspects: 1) the sensor data, i.e. human posts of photos and texts, in social sensing is usually of high-level semantics whereas, in traditional physical sensor networks, data is just time series, 2) the novel paradigm is of zero-cost infrastructure even for global deployment thanks to the omnipresent access to online social network services whereas traditional sensor networks have huge monetary cost even in some mid-sized deployment of tens or hundreds of nodes, considering that each physical sensor costs about tens of US dollars. Maintenance of social sensing is also of little overhead compared with the traditional sensor network where the energy constraint of each individual sensor node is always one of the biggest challenges. With such advantages, recently social sensing has been actively studied and making the huge social impact especially in disaster response [102, 120, 140]. Social Sensing also becomes a vital part of the Cyber-Physical Systems (CPS) [75, 101, 110] for data collection and enabled a promising type of CPS, namely Humans-in-the-loop CPS [104].

Although with the advantages, social sensing has intrinsic challenges that we need to address, which motivates the thesis, as elaborated below.

## 1.1 Motivation and Challenges

With the online social network services platform, nowadays, each individual could be a data source broadcasting to a bunch of audiences (or data consumers), which is sharply distinct from the traditional scenario where one a few news media sites are broadcasting. With a simplified analysis, the message volume has grown from  $O(N)$  as in traditional broadcasting scenarios where  $N$  is the number of data consumers to  $O(N^2)$  as in social networks since each individual could be both a data source and a data consumer, as illustrated in Fig. 1.1. As a consequence, the biggest challenge

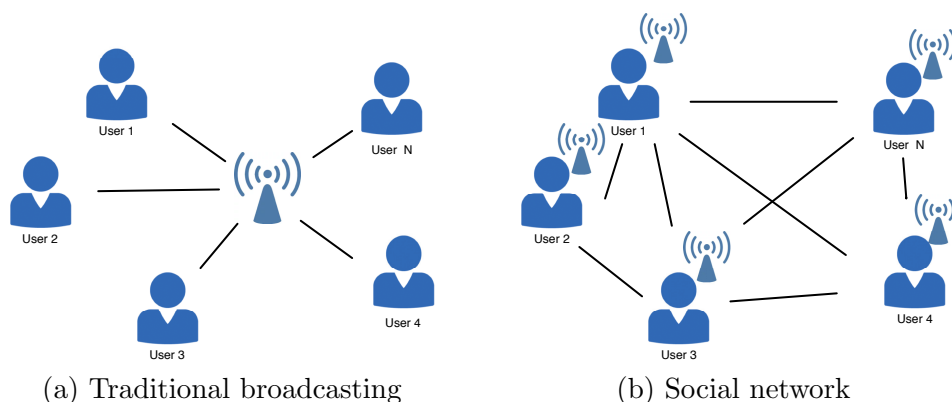


Figure 1.1: Contrast illustration of traditional broadcasting network and social network.

in social sensing is that *the massive data volume has far exceeded the capacity of each individual's cognitive information consumption ability*. Therefore, we need to filter the social sensing data and only deliver useful information and intelligence to the users, which is the motivation of this thesis. Specifically, this thesis studies and proposes three levels of information filtering, each for one aspect of social sensing data. The motivations and challenges of the filtering are presented respectively as follows:

- **Untruthful Data Removal:** The advent of online social network services empowers people the freedom to publish their observations with little regulation. One direct consequence is the untruthful data in online social media including rumors and other untruthfulness. With the purpose of high data quality, one challenge of social sensing is to remove the untruthful information, a practice usually termed as *fact-finding*. Multiple previous research efforts of fact-finding were proposed [124, 127, 130], however, they only considered the case that the system state is static and the underlying variables are independent. In practical, especially

in Cyber-Physical System applications, the system state is usually time-varying and the underlying variables are interdependent, where none of the previous solutions would work.

- **Event-level Data Summarization:** Feeding individual posts (like tweets in Twitter) to social sensing users would not very helpful for their high-level decision making because information organization is a huge overhead for normal individual users. Delivering event-level summarization would greatly reduce the information organization overhead, because, intuitively, people are usually interested in physical events. Furthermore, the users are also interested in how each event instance evolves along the time. Furthermore, we prefer an unsupervised approach since the overhead of acquiring data labels is massive, and language-agnostic for application generality. Therefore, how to detect physical events from the massive social sensing data and track the development of individual event instances across time in an unsupervised fashion without understanding language semantics comprise the challenges in this level of filtering.
- **Information-Maximization Data Delivery:** Sometimes even with our first two levels of filtering the data volume might also be too large to be consumed by some user when s/he has some resource constraints like lacking time to read all 50 events detected but only being able to read 5 of them. Therefore, when delivering data to users, we need to consider which piece of information should be delivered first. Different ranking schemes would have different values of utility under some application scenario. In this thesis, we consider one scenario, where the users would prefer diversifying the information delivered. The rationale behind this application scenario is that the more diversified information delivery will more quickly provide users “big picture”, which is similar to what the outline does in a book. The challenge of this level of filtering lies in the theoretical analysis of the performance of diversified information delivery.

This thesis addresses the above challenges by designing a systematic information filtering system that is overviewed in the next section.

## 1.2 Thesis Overview

This thesis proposes a suite of multi-level information filtering tools for social sensing applications towards the objective of delivering the most informative content from massive data that is consumable upon individual user’s cognitive capacity. In this section, we overview each filtering module and their organization in the proposed system.

### 1.2.1 Untruthful Data Removal

Fact finding aims at removal untruthful data and has been studied in many recent literature, and it is the first level of information filtering module in our proposed system. However, as discussed in previous section, we observed that for the cases that the system state is time-varying or the underlying variables are inter-dependent, there is few work. Since in some social sensing applications, especially in Cyber-Physical Systems, it is common that time-varying system state and/or interdependent variables hold, we need new solutions of fact finding that would work bearing these practical assumptions.

#### **Fact-finding with Time-varying System State**

In our work [141], we proposed a solution aiming at addressing fact-finding with time-varying system state, in human-in-the-loop Cyber-Physical Systems. Here, not only do we assume that the error distribution of data sources is unknown but also that each human sensor has its own possibly different error distribution. Given the above assumptions, we rigorously estimate data reliability in social-sensing, hence enabling their exploitation as state estimator in CPS feedback loops. We first consider applications where state is described by a number of binary variables, then extend the approach trivially to multivalued variables. Evaluation results, using both simulation and a real-life case-study, demonstrate the accuracy of the approach.

#### **Fact-finding with Interdependent Variables**

In our work [140], we proposed a solution aiming at address fact-finding with underlying inter-dependent variables. We extends past social sensing literature by offering a scalable approach for exploiting dependencies between observed variables to increase fact-finding accuracy. Prior work



assumed that reported facts are independent or incurred exponential complexity when dependencies were present. In contrast, this paper presents the first scalable approach for accommodating dependency graphs between observed states. The approach is tested using real-life data collected in the aftermath of hurricane Sandy on availability of gas, food and medical supplies, as well as extensive simulations. Evaluation shows that combining expected correlation graphs (of outages) with reported observations of unknown reliability, results in a much more reliable reconstruction of ground truth from the noisy social sensing data. We also show that correlation graphs can help test hypothesis regarding underlying causes, when different hypotheses are associated with different correlation patterns.

### 1.2.2 Event-level Data Summarization

After the first level of filtering that we have determined the true claims versus false claims, thus all the untruthful data would be filtered out. However, feeding a bunch of individual pieces of data (e.g. 1000 tweets) is not efficient for users consumption, since the information organization (usually clustering) overhead would be very high for normal users. Since intuitively the unit of people's focus is event, we want to provide a service that organizes data into event-level and filters out the non-event information, which in practice is also named *event detection*. Furthermore, we want to also summarize the development of each event instance by event tracking in social space. These are the focuses of this level of information filtering.

We develop a service, called *StoryLine* [137], on top of social media content, exploiting humans as "sensors". The service detects and tracks physical urban events of interest to the user, such as car accidents, infrastructure damage (in the aftermath of a natural disaster), or instances of civil unrest. It offers an interface to client-side software that allows browsing such events in real time, as well as an interface for software applications to a structured representation of the events and their related statistics. The service embodies novel algorithm for real-time detection, de-multiplexing, and tracking of physical events using social media data. In our evaluation with Twitter feeds, we show that our service outperforms two state-of-the-art baselines in event detection and demultiplexing. We also conduct two case-studies to show the effectiveness of the real-time event detection capability and event tracking performance of our system.

We further extends the Storyline service by considering both Twitter tweets and Instagram picture posts [136]. Empirical data suggests that event detection from Instagram streams errs on the false-negative side due to the relative sparsity of Instagram data (compared to Twitter data), whereas event detection from Twitter can suffer from false-positives, at least if not paired with careful analysis of tweet content. To tackle both problems simultaneously, we design a unified unsupervised algorithm that fuses events detected originally on Twitter (called T-events), that occur in adjacent periods, in an attempt to combine the benefits of both sources while eliminating their individual disadvantages. We evaluate the proposed framework with real data crawled from Twitter and Instagram. The results indicate that our algorithm significantly improves tracking accuracy compared to baselines.

### 1.2.3 Information-Maximizing Data Delivery

After the second level of information filtering, instead of individual data pieces, we got a bunch of event summaries, which should be much easier for normal users to consume. However, it is possible that users have other constraints, like limited reading time, that prevents them consuming all the event summaries, which requests a data prioritization scheme that always delivers the most informative data first. In this thesis, we define information maximization as the redundancy minimization, and the rationale behind the definition is from the observation that in Cyber-Physical and sensing applications, where data objects are collected from the physical world, typically exhibit significant redundancy in collected data which calls for data diversification to reduce redundancy.

Miverva [139], an information-centric programming paradigm and toolkit was introduced for social sensing. It proposes an information maximizing data deliver scheme under the assumption that each data item has a coverage of some information space and theoretically proves the optimality of the proposed algorithm. Information Funnel [135] system is designed for the tree-structured social sensing data where each data object has a hierarchical name where the length of the common prefix between two names is a rough measure of similarity between the corresponding objects, and the proposed prioritization algorithm achieves information-maximizing data delivery using policies that diversify the transmitted names. Note that the idea of diversifying data transmission fit seamlessly with a novel networking paradigm called Named-Data Networking [64], therefore, we exploit it in

designing our application-level transport protocol in data delivery.

### 1.3 Thesis Organization

In summary, the aforementioned three levels of filtering modules form an information filtering system with the input of raw social sensing data and the output of information-maximizing prioritized event-level truthful data summaries. The thesis will elaborate on each module in great details, shedding light on its design philosophy, algorithm design and analysis, and system implementation. Specifically,

- Part I is about untruthful information removal. In Chapter 2, we address the fact-finding problem with time-varying system state, and in Chapter 3, we address the problem with inter-dependent variables. For each problem, we propose an Expectation-Maximization (EM) algorithm respectively that jointly learns the reliability of each individual source and the truthfulness of each variable.
- Part II is about event-level information summary. In Chapter 4, we propose an unsupervised solution for event de-multiplexing and tracking with Twitter data. Our evaluations with real-world data demonstrate the better efficiency of our proposed solution compared with state-of-the-art. In Chapter 5, we further extend the event tracking scheme by considering fusing both Twitter and Instagram for better precision and recall, and our real-data evaluations corroborate our claim.
- Part III is about information-maximizing delivery. In Chapter 6, we proposed a prioritization scheme based on an assumption of coverage-based information space, and prove the optimality of the proposed algorithm analytically. In Chapter 7, we proposed a prioritization scheme on tree-structured information space, and prove the optimality of the proposed algorithm in theory.

Finally, Chapter 8 concludes this thesis and provides a discussion of future research.

## Part I

# Untruthful Information Removal . . . . .

## Chapter 2

# Fact-finding with Time-Varying System State

In this chapter, we address the challenge of untruthful data removal in social sensing with time-varying system states, which usually holds in Cyber-Physical Systems (CPS). Modern CPS applications increasingly operate in social spaces, where humans play an important part in the overall system. Hence, future applications should increasingly be engineered with an understanding of the humans in the loop. In this chapter, we focus on the role of humans as *sensors* in CPS systems; a practice that is commonly known as *social-sensing* or *crowd-sensing*. Humans act as sensors when they contribute data (either directly or via sensors they own) that a CPS application can use. For example, drivers may contribute data on the state of traffic congestion at various locales, and survivors may contribute data on damage in the aftermath of a natural disaster. A challenge in this context is that data sources may be unreliable. In fact, the reliability of individual observers in crowd-sensing applications is generally not known.

A common thread in CPS research focuses on *reliability* of cyber-physical systems. Most research focused on two aspects of CPS reliability; namely, correctness of *temporal behavior* and correctness of *software function*. In order for crowd-sensing to become a viable component in CPS feedback loops, one needs to understand correctness of *collected observations* as well. We call this latter challenge the data reliability challenge, to complement the challenges of timing reliability and software reliability mentioned above.

Consider a CPS application that uses crowd-sensing to collect data about a physical environment. The data reliability challenge refers to designing a state estimator that takes raw unreliable crowd-sensing data as input and outputs reliable estimates of the underlying physical state of the environment, as well as appropriate error bounds. Building optimal state estimators from noisy inputs is an old topic in estimation theory and embedded systems. Much like our work, past research often assumed that sources are unreliable and the noise model is not known. However,

in the case of physical sensors, prior research usually assumed that errors of different sensors are drawn from the same distribution (or from a small set of different distributions). In contrast, we assume that each source is unique. Hence, each source has its own error distribution. None of these distributions is known.

In this chapter, we also assume that the state of the observed environment changes over time. Hence, when conflicting observations arrive, it is not clear whether one is wrong, or whether the ground truth changed between observations. Had the reliability of different observation sources been known, it would have been easy to statistically fuse them, but since error distributions are both unknown and unique to each source, reconciling conflicts is a bigger problem. This work is the *first* to offer a rigorous estimation-theoretic approach for state estimation in crowd-sensing applications, where (i) observers have unknown reliability (ii) the error distribution is unique to each observer, and (iii) the observed physical events have time-varying state. It extends prior work by the authors, that solved the problem in the restricted special case when physical state is immutable [126,127,131]. Note that, this restriction is not suited for most cyber-physical systems.

One way to accommodate state changes is to cut time into small observation windows and consider only one observation window at a time, during which state can be assumed constant. One can then apply the former static approach [131] independently within each window. Unfortunately, this reduces the number of observations that can be considered within a given window, making it harder to assess their veracity. A much better approach is to take into account the model of state evolution from one window to the next and reduce trust in observations that are less consistent with that model. Unlike traditional estimation problems, where a model of observation noise is also available, in crowd-sensing, observations can come from different sources whose reliability (i.e., noise model) is not known. Hence, it is hard to tell genuine state changes from incorrect reports. Our contribution lies in taking a model of state evolution into account such that a maximum likelihood estimate can be arrived at, that jointly estimates the reliability of individual observations and the reliability of individual sources, taking only a dynamic model of the underlying observed system as input.

We analytically derive an error bound for the above estimator, by computing the Cramer-Rao lower bound [28] that bounds estimator variance and hence derive confidence intervals. We then

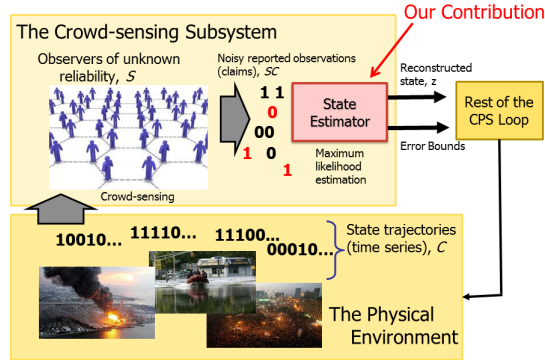


Figure 2.1: An example of CPS system with humans in the loop.

evaluate our algorithm through simulations and a real-world crowd-sensing application in which sources report the availability of street parking spots on a university campus. We show that our algorithm outperforms prior state-of-the-art solutions in both event state estimation accuracy and source reliability estimation accuracy.

The rest of the chapter is organized as follows. Our problem is formulated in Section 2.1. Section 2.2 describes how a dynamic system model is converted to an input for our maximum likelihood estimator. We describe our algorithm in Section 2.3 and its analysis in Section 2.4. We extend our algorithm to the general multivalued case in Section 2.5. Our solution is evaluated in Section 2.6. Section 2.7 reviews the related work. Finally, we conclude the chapter in Section 2.8.

## 2.1 Problem Formulation

Consider a CPS application that uses a crowd-sensing subsystem to estimate the state of a physical environment that changes dynamically over time. An example of such a system is shown in Figure 2.1. It is desired to develop the appropriate state estimator that converts raw noisy crowd-sensing data, from sources of unknown reliability, into state estimates of quantified reliability and error bounds.

We model the physical environment by a set of measured variables,  $C$ , whose values constitute the system state we want to estimate. We consider applications where the state of interest varies over time (i.e., the values of these variables change dynamically).

We focus, in this chapter, on the *harder* case, where state variables are binary. While it may appear to be restrictive, this particular case is more computationally challenging because state, in this case, does not have “inertia”. In continuous systems, such inertia leads to a smooth state evolution that can be leveraged to eliminate outliers, extrapolate trends, and suppress noise. A binary variable, in contrast, can change between the two extremes of its range (0 and 1) at any time. Hence, removing incorrect measurements, predicting correct values, and eliminating noise become harder problems. Indeed we show that solutions to the binary case can easily generalize to the multivalued-state case.

One should also note that exploring systems of binary states is more than just a step towards understanding more general state representations. Binary state is a versatile abstraction. It can indicate, for example, presence or absence of arbitrary conditions, symptoms, features, or alarms in specified locations of the monitored system. More importantly, given the general lack of reliability of human observers in crowd-sensing scenarios, tasking humans with making simple binary observations makes more sense from the perspective of minimizing opportunities for human error. Hence, the authors conjecture that crowd-sensing will likely gravitate to an application space where binary variables are the commonly measured state, assuming that the algorithmic estimation challenge is solved, which is the purpose of this work.

We denote the set of data sources by  $\mathcal{S}$ . Time is slotted, such that all the reports generated within time-slot  $k$  are timestamped with  $k$ . Data is available from multiple time slots. We use a 3D matrix  $SC$  to summarize the reports, where  $SC_{i,j,k} = v$  means that the source  $i$  reports (claims) that variable  $j$  has value  $v$  in the  $k$ -th time-slot.  $SC$  is called the *source-claim matrix* in this chapter.

In a system with time-varying states, we need to account for state transitions. We aim at a general formulation that is able to support a wide range of crowd-sensing applications. Similarly to what’s done in multi-target tracking and hypothesis testing, we translate the dynamic or state transition model of a variable into the joint probability of any given sequence of observed values over a finite time horizon. A different probability is computed for each possible sequence. For example, assuming that a variable has three possible values,  $a, b, c$ , and that the finite time horizon has two time-slots, we have  $3^2 = 9$  possible sequences (or trajectory hypotheses), each has a probability



that is computable from the dynamic system model.

In general, suppose that a variable  $j$  has  $q$  possible values and we consider a window of  $H$  time-slots, then we must consider  $q^H$  hypotheses on its possible trajectory. The probability of each hypothesis can be computed from the dynamic state transition model. We call these probabilities the *trajectory probability vector* for variable  $j$ . Combining the trajectory probability vectors of all variables, we thus have a *trajectory probability matrix* denoted by  $\mathcal{P}$ . Note that, the trajectory probability matrix represents prior beliefs that can be computed in advance from the system model. It remains to combine those prior beliefs with received claims of different observers who report values of *some* variables in *some* time slots.

It is not hard to see that the size of  $\mathcal{P}$  increases exponentially in  $H$ , which prevents us from considering a long history. Fortunately, we only need a relatively short history to estimate current system state within reasonable accuracy as shown in the evaluation, Section 2.6. As per our evaluation results, considering the past 5 time-slots leads to a reasonably accurate estimation on the current state, and considering more history actually results in very small increments in estimation accuracy. This is because that the older state has less influence on the current state, and thus can be omitted without much loss of estimation accuracy. Therefore, our state transition formulation is both general and computationally feasible in practical settings.

Let us denote the two possible values of each binary state variable in our model by  $T$  and  $F$ , respectively. (In Section 2.3, we generalize our model to the multivalued case.) Since in crowd-sensing, participants report state at will, and in no systematic fashion, in the binary case, three values become possible in the source-claim matrix  $SC$ , namely,  $T$ ,  $F$ , or  $U$ , where  $U$  represents *unknown*, meaning “lack of reports”. The default value of the source-claim matrix  $SC$  is  $U$ , which means that we do not assume a default system state.

In contrast to much prior work on state estimation from unreliable sources, we assume not only that source error distribution is not known, but also that each source has a different error distribution. In the case of binary signals, one can summarize the error distribution by a single value,  $t_i$ , denoting the reliability of source  $i$ . It is defined as the probability that *when  $i$  claims that variable  $j$  has value  $v$  at time  $j$ , it is indeed of value  $v$  at that time*. Hence, the probability of error is  $1 - t_i$ . (For multivalued signals, the above is only a partial specification of the error probability

distribution since it does not mention how the probability of error is split across possible error values.) Let  $C_{j,k}$  denote the value of variable  $j$  in time-slot  $k$ , and  $SC_{i,j,k}$  denote the value of variable  $j$  that source  $i$  reports in time-slot  $k$ . The reliability of source  $i$  can be formally defined as  $t_i = \Pr(C_{j,k} = v | SC_{i,j,k} = v)$ . Let's use a short notation  $C_{j,k}^v$  for  $C_{j,k} = v$ , and  $SC_{i,j,k}^v$  for  $SC_{i,j,k} = v$ , then the source reliability is:

$$t_i = \Pr(C_{j,k}^v | SC_{i,j,k}^v).$$

Let  $T_{i,v}$  denote the probability that source  $i$  reports that variable  $j$  is in state  $v$  given that the variable is really in state  $v$  at that time. Let  $F_{i,v}$  denote the probability that source  $i$  reports that variable  $j$  is in state  $\bar{v}$  given that  $j$  is in state  $v$ . Formally,  $T_{i,v}$  and  $F_{i,v}$  are defined as

$$T_{i,v} = \Pr(SC_{i,j,k}^v | C_{j,k}^v), F_{i,v} = \Pr(SC_{i,j,k}^{\bar{v}} | C_{j,k}^v).$$

Note that,  $T_{i,v} + F_{i,v} \leq 1$ , since it is also possible that the source  $i$  does not report the value of the variable. Let  $u$  denote the ‘‘Unknown’’ value in the source-claim matrix  $SC$ , we have:

$$1 - T_{i,v} - F_{i,v} = \Pr(SC_{i,j,k}^u | C_{j,k}^v).$$

We denote the prior probability that a source  $i$  makes a claim by  $s_i$ , and denote the prior probability that any variable at any time is in state  $v$  by  $d^v$ . By the Bayesian theorem, we have:

$$T_{i,v} = t_i \cdot s_i / d^v, F_{i,v} = (1 - t_i) \cdot s_i / d^v. \tag{2.1}$$

Our problem can be formulated as follows: *Given the source-claim matrix  $SC$  for the past  $H$  time-slots, and given the trajectory probability matrix  $\mathcal{P}$ , jointly estimate both the reliability of each source in  $\mathcal{S}$ , and the current state of each variable in  $\mathcal{C}$ .*

## 2.2 Computing Trajectory Probabilities

Before describing our solution to the above problem, in this section, we use two examples (with different state transition models) to illustrate how a trajectory probability matrix  $\mathcal{P}$  is computed.

### 2.2.1 Independent State Change

We first start with a simple state transition model where the value of each variable is independent from that of the other variables as well as its history values. For simplicity, we consider binary variables. The multivalued case can be generalized trivially. In this system, two parameters are enough to model state: (1)  $P_t$ , the probability that a variable is in state  $T$ , and (2)  $P_f$ , the probability that a variable is in state  $F$ .

Given a value sequence of a variable, we can compute the joint probability of all elements of the sequence easily. For example, the joint probability of a value sequence  $TTF$  is simply  $P_t^2 P_f$ . Therefore, if we use the last  $H$  time-slots to estimate the current system state, we can define the trajectory probability matrix  $\mathcal{P}$  using  $2^H$  joint probabilities; each joint probability is for one possible sequence of length  $H$ .

### 2.2.2 Markov Model

We now consider a system whose state transitions follows a Markov model, in which current state (the values of the variables in the system) is determined only by its last state. For simplicity, again, in this example, the variables are binary. The multivalued variables can be easily generalized. In a Markov model with binary variables, two transition probabilities are enough to describe the system dynamics: (1)  $P_{tf}$ , the probability that a variable changes its current state from  $T$  to  $F$  (in the next time-slot), and (2)  $P_{ft}$ , the probability that a variable changes its state from  $F$  to  $T$ . The probability that a variable remains in the  $T$  state in the next time-slot ( $P_{tt}$ ) can be easily computed by  $P_{tt} = 1 - P_{tf}$ . Similarly,  $P_{ff} = 1 - P_{ft}$ .

Given a state trajectory, and the probability of its initial  $T$  state  $P_t^0$  or  $F$  state  $P_f^0$  (such that  $P_t^0 + P_f^0 = 1$ ), we can easily compute its probability. For example, if the trajectory is  $TTF$  the joint probability of the state sequence is  $P_t^0 \cdot P_{tt} \cdot P_{tf}$ , where  $P_{tt}$  and  $P_{tf}$  are the transition probabilities. Therefore, if we exploit the last  $H$  time-slot to estimate the current system state, in

this model, the trajectory probability matrix  $\mathcal{P}$  can be computed using the joint probabilities of  $2^H$  state combinations, where each of the joint probabilities can be easily computed as illustrated above.

The above examples are selected for the ease of illustration. For evaluating trajectory probabilities in the presence of more complex system dynamics, please refer to hypothesis testing and target tracking literature.

## 2.3 Dynamic State Estimation

In this section we describe our state estimator for crowd-sensing applications. We adopt a maximum-likelihood estimation framework, and restate the problem as one of finding the set of (i) source reliability values, and (ii) trajectories of state variables that jointly maximize the likelihood of our observations (i.e., received claims). This problem is then solved using the Expectation-Maximization framework [30]. We call the resulting algorithm EM-VTC (Expectation-Maximization algorithm for the time-Varying ground Truth case with Conflicting claims).

### 2.3.1 Deriving a Crowd-sensing State Trajectory Estimator

Expectation-Maximization (EM) [30] is a machine learning algorithm to find the maximum likelihood estimates of parameters in a statistical model when the likelihood function contains latent variables. To apply the EM algorithm, we need to define the likelihood function  $L(\theta; x, Z)$ <sup>1</sup>, where  $\theta$  is the parameter vector to be estimated,  $x$  is the vector of the observed data, and  $Z$  is the vector of the latent variables. After defining the likelihood function, EM iteratively applies two steps called the E-step and the M-step until they converge to a solution that computes the values of both the parameter vector and the latent variables. The mathematical formulation of these iterations is given below:

- E-step: Given the current (estimated) parameter vector and the observed data, compute the

---

<sup>1</sup>In this chapter, we use capital letters for random variables, such as  $Z$ , and use small letters for the values of random variables, such as  $z$ .

expectation of the latent variables.

$$Q(\theta|\theta^{(n)}) = E_{Z|x,\theta^{(n)}}[\log L(\theta; x, Z)]. \quad (2.2)$$

- M-step: Find the parameters that maximize the  $Q$  function defined in the E-step, and use these parameters for the next iteration.

$$\theta^{(n+1)} = \arg \max_{\theta} Q(\theta|\theta^{(n)}) \quad (2.3)$$

We introduce a latent variable  $z_{j,k}$  for each state variable  $j$  in time-slot  $k$  to denote its estimated value in that time-slot. We use vector  $z_j$  to denote the estimated time-series of state variable  $j$  in the last  $H$  time-slots, where  $H$  is a parameter of the algorithm as described in Section 2.1. We use  $Z_{j,k}$  to denote the random variable corresponding to  $z_{j,k}$ , and the  $Z_{j,k}$ 's,  $\forall j \in \mathcal{C}$  and  $k \in \{1, 2, \dots, H\}$ , constitute the random matrix  $Z$ . We define  $x$  to be the 3-dimension source-claim matrix  $SC$ , where  $x_j$  is the matrix of reported observations of variable  $j$  from all sources in  $\mathcal{S}$  of all of the  $H$  time-slots. Note that, the matrix may be sparse (i.e., containing a lot of ‘‘U’’ values) since many sources will not have observed many variables. We define the parameter set  $\theta$  to be  $\{(T_{i,v}, F_{i,v}) | \forall i \in \mathcal{S}, v \in \{True, False\}\}$ , where  $T_{i,v}$  and  $F_{i,v}$  is defined in Equation (2.1).

The likelihood of receiving the claims reported by all sources in a crowd-sensing application becomes as follows:

$$\begin{aligned} L(\theta; x, Z) &= \prod_{j \in \mathcal{C}} p(x_j, Z_j | \theta) = \prod_{j \in \mathcal{C}} \left\{ \sum_{z_j \in \Lambda^H} p(x_j, z_j | \theta) \cdot \mathbf{1}_{\{Z_j = z_j\}} \right\} \\ &= \prod_{j \in \mathcal{C}} \left\{ \sum_{z_j \in \Lambda^H} p(z_j) \mathbf{1}_{\{Z_j = z_j\}} \cdot \prod_{i \in \mathcal{S}} \prod_{k=1}^H \alpha_{i,j,k} \right\} \end{aligned} \quad (2.4)$$

where  $\Lambda = \{T, F\}$ ,  $\Lambda^H$  denotes the Cartesian product<sup>2</sup> of the set  $\Lambda$  itself for  $H$  times, and  $\mathbf{1}_{\{x\}}$  is an indicator function whose value is 1 only if  $x$  is true otherwise 0. Please note that  $p(z_j)$  is the input (prior) trajectory probability vector (the  $j$ -th row of the trajectory probability matrix  $\mathcal{P}$ ), which is independent of the parameters  $\theta$ . Therefore,  $p(z_j) = p(z_j | \theta)$ . The  $\alpha_{i,j,k}$  is defined as

<sup>2</sup>For example, if  $A = \{1, 2\}$  and  $B = \{a, b\}$ , the Cartesian product of  $A$  and  $B$  is  $A \times B = \{(1, a), (1, b), (2, a), (2, b)\}$ .

follows:

$$\forall v \in \Lambda, \alpha_{i,j,k} = \begin{cases} T_{i,v} & \text{if } z_{j,k} = v, SC_{i,j,k} = v \\ F_{i,v} & \text{if } z_{j,k} = v, SC_{i,j,k} = \bar{v} \\ 1 - T_{i,v} - F_{i,v} & \text{if } z_{j,k} = v, SC_{i,j,k} = u \end{cases} \quad (2.5)$$

where  $u$  denotes the ‘‘Unknown’’ value  $U$  in the source-claim matrix  $SC$ .

The derivations of the E-step and M-step are in the last section. In the next subsection, we present our algorithm in pseudo code.

### 2.3.2 The EM-VTC Algorithm

The pseudo code of our EM-VTC algorithm is shown in Algorithm 1. The inputs of our algorithm are the source-claim matrix  $SC$  with  $H$  time-slots, where  $H$  is a fixed parameter, and the trajectory probability matrix  $\mathcal{P}$  that is learned from history data. Both the source reliability and the estimated variable value in the current time-slot are returned by the algorithm. We estimate the source reliability using Equation (2.1), where  $s_i^v$  can be calculated from the source-claim matrix  $SC$ ,  $d^v$  can be computed by  $\sum_{j \in \mathcal{C}} \sum_{k=1}^H Z_v^c(j, k)$ , and  $T_{i,v}^c$  and  $F_{i,v}^c$  are calculated after the EM iterations are converged.

## 2.4 Accuracy Guarantees

After developing the EM-VTC algorithm, the next natural question is: How accurate is its estimation results? In this section, we answer the above question by first deriving the Cramer-Rao lower bound (CRLB) for the EM-VTC algorithm and then deriving a confidence interval based on the CRLB. In statistics, the CRLB represents a lower bound on the estimation variance of a deterministic parameter [28]. Note that the CRLB derived here is assuming there are enough sources participating in the crowd-sensing application therefore the truth of the variables are known with full accuracy, thus the CRLB is asymptotic. The derivation of the CRLB is in the last section.

### 2.4.1 Confidence Interval of Source Reliability

In this subsection, we derive a confidence interval of source reliability based on the obtained (asymptotic) CRLB (in the last section). Maximum likelihood estimators exhibit several nice properties,

---

**Algorithm 1** EM-VTC: Expectation-Maximization Algorithm with Time-Varying Variables

---

**Input:** The source-claim matrix  $SC$  in the latest  $H$  time-slots, and the trajectory probability matrix  $\mathcal{P}$ .

**Output:** The estimated values of variables in the current time-slot, and the estimated reliability of each source.

Initialize  $\theta^{(0)}$  by setting  $T_{i,v}$  and  $F_{i,v}$  to random values between 0 and 0.5

$n \leftarrow 0$

**repeat**

**for** Each  $j \in \mathcal{C}$ , each  $k \in \{1, 2, \dots, H\}$ , and each  $v$  in  $\{T, F\}$  **do**

    Compute  $Z_v^{(n)}(j, k)$  based on Equation (2.11)

**end for**

**for** Each  $i \in \mathcal{S}$  and each  $v$  in  $\{T, F\}$  **do**

    Compute  $T_{i,v}^*$  and  $F_{i,v}^*$  based on Equation (2.10)

**end for**

$n \leftarrow n + 1$

**until**  $\theta^*$  and  $\theta^{(n)}$  converge

$Z_v^c(j, k)$  is the converged value of  $Z_v^{(n)}(j, k)$ , and  $T_{i,v}^c$  is the converged value of  $T_{i,v}^{(n)}$ ,  $F_{i,v}^c$  is that of  $F_{i,v}^{(n)}$ , for every  $i \in \mathcal{S}, j \in \mathcal{C}, k \in \{1, 2, \dots, H\}$  and  $v \in \{T, F\}$ .

**for** Each  $j \in \mathcal{C}$  **do**

**if**  $Z_T^c(j, 1) > Z_F^c(j, 1)$  **then**

    Variable  $j$  is assigned  $T$  in the current time-slot

**else**

    Variable  $j$  is assigned  $F$  in the current time-slot

**end if**

**end for**

**for** Each  $i \in \mathcal{S}$  **do**

  Compute source  $i$ 's reliability  $t_i$  by Equation (2.1)

**end for**

---

one of which is asymptotic normality that the MLE estimator is asymptotically distributed in Gaussian as the data size is large enough [24]:

$$(\hat{\theta}_{MLE} - \theta^0) \rightarrow^{\mathcal{D}} N(0, J^{-1}(\hat{\theta}_{MLE})) \quad (2.6)$$

where  $J$  is the Fisher information matrix as defined in Equation (2.13),  $\theta^0$  and  $\hat{\theta}_{MLE}$  are the ground truth and MLE of the parameter  $\theta$  respectively. In other words, as the data size growing up, the difference between the true value and the MLE of the parameters follows normal distribution with mean 0, and covariance matrix given by the CRLB  $J^{-1}(\hat{\theta}_{MLE})$ .

The variance of estimation error on parameter  $T_{i,T}$  is  $J^{-1}(\hat{\theta}_{MLE})_{i,i}$ . We know that the reliability of source  $i$  is  $t_i = \frac{d^T}{s^T} T_{i,T}$  by Equation 2.1. Therefore, by the  $\Delta$ -method [24], we have the variance of

reliability estimation error equals to  $(\frac{d^T}{s_i^T})^2 J^{-1}(\hat{\theta}_{MLE})_{i,i}$ . We denote this variance by  $V_i$ . Therefore, the confidence interval to quantify the source reliability  $t_i$  is given as follows:

$$(\hat{t}_i^{MLE} - c_p \cdot \sqrt{V_i}, \hat{t}_i^{MLE} + c_p \cdot \sqrt{V_i}) \quad (2.7)$$

where  $c_p$  is the standard score of the confidence level  $p$ . For example, for the 95% confidence level,  $c_p = 1.96$ .

## 2.5 Multivalued Variable Extension

In this section, we extend our EM-VTC algorithm from the binary case to a general multivalued case, where each variable has  $q$  ( $\geq 2$ ) possible values. Although Wang et al. [129] designed an EM algorithm that takes multivalued variables for the static state case, we found that their algorithm is not suitable in the time-varying state case. The main reason is that the time complexity of *each* EM iteration in their algorithm is  $O(q^H)$ , if the last  $H$  time-slots are considered in estimating the current system state. Please note that the EM iterations are the major time-consuming part of an algorithm under the EM framework. Therefore, for a large  $q$ , the heavy computational overhead of their algorithm makes it not practically applicable, especially in time sensitive systems. One of our goal is to time-efficiently extend our binary solution for the multivalued case. Specifically, we require a solution in the  $q$ -valued case in which the time complexity of each EM iteration grows *no faster than linearly* in  $q$  compared with the binary solution.

The pseudo code is shown in Algorithm 2. Our high-level idea is to reduce the multivalued case to a binary case. Suppose that in the multivalued case the value set  $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_q\}$ . We first construct  $q$  new binary variables for each  $q$ -nary variable (line 1 to line 3). Next, we construct the source-claim matrix  $SC_m^b$  for the binary variables corresponding to each  $q$ -nary variable  $m$  based on the  $SC$  (line 4 to line 12). We then construct the trajectory probability matrix  $\mathcal{P}_m^b$  for the constructed binary variables corresponding to each multivalued variable  $m$  from the trajectory probability matrix  $\mathcal{P}$  that is an input of our algorithm (line 13 to line 20). The combinations of  $SC_m^b$  and  $\mathcal{P}_m^b$  are denoted as  $SC^b$  and  $\mathcal{P}^b$  respectively. Next, we apply Algorithm 1 with inputs  $SC^b$  and  $\mathcal{P}^b$  to get the source reliability and the converged  $Z_T^c$  value of each of the binary variables



---

**Algorithm 2** EM-VTC for multivalued variables

---

**Input:** The source-claim matrix  $SC$  in the last  $H$  time-slots, the trajectory probability matrix  $\mathcal{P}$

**Output:** The estimation of source reliability and the current variable values

```
1: for Each  $q$ -nary variable  $j \in \mathcal{C}$  do
2:   Construct  $q$  binary variables  $j_1, j_2, \dots, j_q$ , such that  $j_i = T$  if  $j = \lambda_i$  and  $F$  otherwise.
3: end for
4: Allocate the memory for the source-claim matrix  $SC_m^b$  for the new binary variables corresponding to each  $q$ -nary
   variable  $m$ . Totally, allocate  $SC^b$  with size  $|\mathcal{S}| \times q \cdot |\mathcal{C}|$ , and initiate  $SC^b$  with the “Unknown” value  $U$ .
5: for Each  $i \in \mathcal{S}, j \in \mathcal{C}, k \in \{1, 2, \dots, H\}$  do
6:   if  $SC_{i,j,k} = \lambda_m$  then
7:      $SC_{i,j_m,k}^b \leftarrow T$ 
8:     for Each  $m' \in \{1, \dots, m-1, m+1, \dots, q\}$  do
9:        $SC_{i,j_{m'},k}^b \leftarrow F$ 
10:    end for
11:   end if
12: end for
13: Allocate the memory for the trajectory probability matrix  $\mathcal{P}_m^b$  for the new binary variables corresponding to each
    $q$ -nary variable  $m$ . Totally, allocate  $\mathcal{P}^b$  with size  $2^H \times q$ , and initiate  $\mathcal{P}^b$  by 0.
14: for Each  $q$ -nary variable  $j \in \mathcal{C}$ , each value  $\lambda_m \in \Lambda$  do
15:   for Each element  $\gamma$  of  $\mathcal{P}_j$  do
16:      $\triangleright$  (Comment:  $\gamma$  is some combination of  $H$   $q$ -nary values, and  $\mathcal{P}_j$  is a column vector.)
17:     Compute the corresponding combination of  $H$  binary values of variable  $j_m$  and the index  $\gamma^b$  in  $\mathcal{P}_{j_m}^b$ .
18:      $\mathcal{P}_{j_m}^b \leftarrow \mathcal{P}_{j_m}^b + \mathcal{P}_j$ .
19:   end for
20: end for
21: Use Algorithm 1 with  $SC^b$  and  $\mathcal{P}^b$  to get the source reliability  $t_i$  for each  $i \in \mathcal{S}$  and  $Z_T^c(j_m, 1)$  for each  $j \in \mathcal{C}, m \in$ 
    $\{1, \dots, q\}$ 
22: for Each  $j \in \mathcal{C}$  do
23:    $m \leftarrow \arg \max_{m=1}^q Z_T^c(j_m, 1)$ 
24:   Variable  $j$  is assigned with value  $\lambda_m$ .
25: end for
```

---

(line 21). Please note that here we do not estimate the state of each binary variable, but only exploit the converged  $Z_T^c$  values. Finally, we assign each of the  $q$ -nary variables with the value whose corresponding binary variable has the highest  $Z_T^c$  value among all the binary variables corresponding to the  $q$ -nary variable (line 22 to line 25).

Please note that here the time complexity for constructing the trajectory probability matrix for the binary variables is  $O(q^{H+1})$ , and this procedure is executed only *once*. With the constructed trajectory probability matrix  $\mathcal{P}^b$  and the source-claim matrix  $SC^b$ , each EM iteration in Algorithm 1 has the time complexity  $O(q \cdot 2^H)$ . Therefore, we observe that the computational complexity for the multivalued case increases almost *linearly* in the number of possible values that each variable can take (i.e.  $q$ ).

## 2.6 Evaluation

In this section, we evaluate the performance of our algorithm compared with other state-of-the-art solutions and a simple baseline algorithm. We first study the performance in a simulation study, then we evaluate our algorithm in a real-world crowd-sensing application.

### 2.6.1 Simulation Study

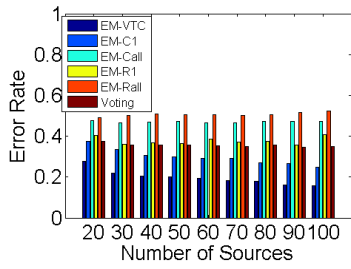
#### Methodology

We build a crowd-sensing simulator in Matlab R2013b. In the simulation, 200 binary variables are created whose initial values are assigned randomly. Each variable represents a physical event with state  $T$  or  $F$ . The initial value of each variable is distributed uniformly at random (i.e., with probability 0.5 the value is assigned to  $T$  and 0.5 it is assigned to  $F$ ). For transition probabilities, we assume a two state Markov model. That is, the value of each variable in one time-slot depends only on its value in the preceding one. There are only two states,  $T$  and  $F$ . The transition probability from  $T$  to  $T$  is  $P_{tt}$ , and from  $F$  to  $F$  is  $P_{ff}$ . These two parameters are enough to determine the other two transition probabilities in the Markov model. While we could have considered more complex and realistic systems, our goal from considering the two-state Markov model was to help understand the fundamental performance trends of our state estimator as a function of parameters of the system model. Results for more complex models would have been harder to interpret due to the multitude of confounding factors at play.

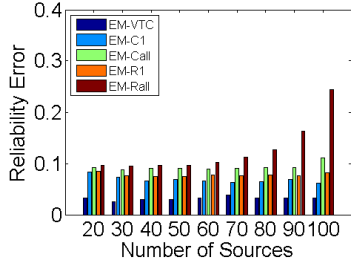
For the sources, the simulator also choose a reliability  $t_i$  for each source  $i$ . We set the reliability of each source randomly distributed in  $[0.5, 1)$ . In the simulation, each source is assigned a probability of making claims,  $s_i$ , meaning the probability that a source reports an observation. The higher the  $s_i$  is, the more “talkative” the source is.

The default values of the parameters are as follows: the number of sources is 30, the expected source reliability  $E(t_i) = 0.6$ , the factor  $s_i = 0.6$ , the number of history time-slots to be considered  $H = 5$ , the state transition probabilities  $P_{tt} = P_{ff} = 0.5$ , and the initial ground bias  $d_j^T = 0.5$  denoting the probability that a variable is assigned  $T$  initially.

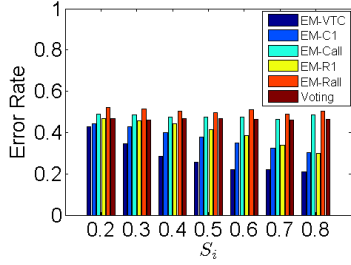
We compare our algorithm **EM-VTC** with two state-of-the-art algorithms proposed in [131]



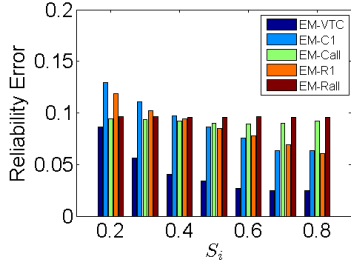
(a) Variable state estimation



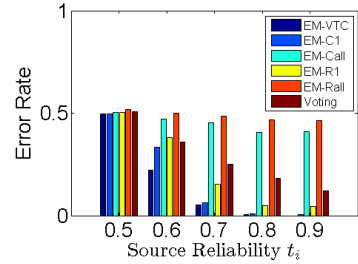
(b) Source reliability estimation



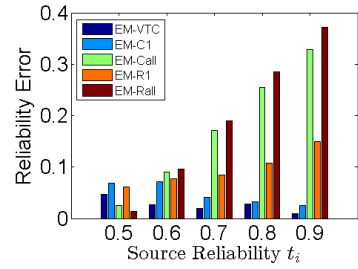
(a) Variable state estimation



(b) Source reliability estimation



(a) Variable state estimation



(b) Source reliability estimation

Figure 2.2: Performance as the number of sources varies.

Figure 2.3: Performance as the talkative factor  $s_i$  of sources varies.

Figure 2.4: Performance as the source reliability  $t_i$  varies.

and [129]. The algorithm proposed in [131] does not consider state changes. It assumes that the default physical state of each variable is “F”, allowing sources to report only “T” values of the observed variables. The algorithm proposed in [129] extends the above algorithm by considering conflicting claims (i.e., both “T” and “F” values). However, it still assumes that system state is immutable. For each of the two algorithm, we further consider two cases: (1) Applying the algorithm with the data in the current time-slot, and (2) Applying the algorithm with the data in the last  $H$  time-slots. The algorithm in [131] is denoted by **EM-R1** when it is fed with the data of the current time-slot, and by **EM-Rall** when it is fed with the data of the last  $H$  time-slots. The algorithm in [129] is denoted by **EM-C1** when used in the current time-slot, and by **EM-Call** when used in the last  $H$  time-slots.

We also compare our algorithm to a simple baseline algorithm **Voting**. Voting estimates the variable to be equal to the majority vote (i.e., most frequently reported value at the time). Each simulation runs 100 times and each result is averaged on the 100 executions.

## Evaluation Results

In Figure 2.2, we evaluate the performance of our algorithm as the number of sources varies from 20 to 100. Other parameters are set to the default values. From Figure 2.2(a), we can observe that the false estimation rate of our algorithm EM-VTC is the smallest among all the algorithms. The EM-Call and EM-Rall algorithms are the worst in estimating the variable values, since they do not consider the fact that the physical state of each variable changes in time. Without considering the time-varying states, it is very unlikely to correctly estimate the source reliability, as shown in Figure 2.2(b). The reason is that the reports from a 100% reliable source might look “self-conflicting” to EM-Call and EM-Rall in a system with time-varying states, because they assume the system state is immutable. Therefore, the EM-Call and EM-Rall assigns a relatively low reliability to the 100% reliable source.

EM-C1 and EM-R1 performs worse than EM-VTC, because they use only the data in the current time-slot, while the EM-VTC algorithm uses all the data in the last  $H$  time-slots. With more data, the source reliability can be learned better as shown in Figure 2.2(b), which in turn results a better estimation of variable values. EM-C1 outperforms than EM-R1, because EM-R1 does not distinguish “Unknown” from “False”, thus EM-R1 gives the  $F$  state a higher weight (so higher false negative rate).

We can also observe from Figure 2.2, as the number of sources increases, the estimation of EM-VTC becomes better and better. The reason is that more sources means potentially less “Unknown” values in the source-claim matrix  $SC$ . Therefore, estimation becomes more accurate.

Figure 2.3 shows the performance as the factor  $s_i$  (probability of reporting) varies from 0.2 to 0.8. Other parameters are set to the default values. As  $s_i$  increases, the estimation error of EM-VTC becomes smaller as shown in Figure 2.3(a). With less unknown values in the source-claim matrix, the EM algorithm can jointly estimate the source reliability (Figure 2.3(b)) and variable value (Figure 2.3(a)) more accurately. In Figure 2.3, we show that EM-VTC outperforms all the baselines in source reliability estimation as well.

Figure 2.4 shows the performance as the expected source reliability  $E(t_i)$  changes from 0.5 to 0.9. As the expected reliability increases, our algorithm performs better. When the expected source reliability is 0.5, the sources essentially make random reports, offering no information. Therefore,

error rate is around 50%. However, when source reliability increases, our EM-VTC algorithm outperforms all the others. When the source reliability is 90%, our algorithm actually estimates the values of variables 100% correctly.

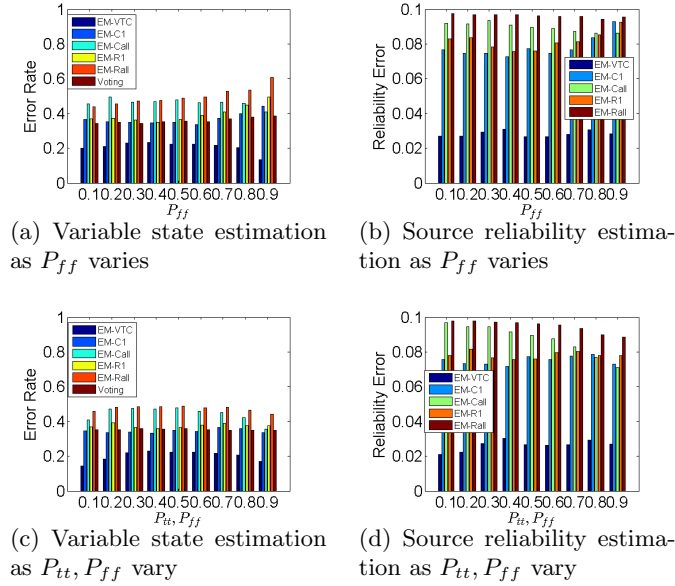


Figure 2.5: Performance as the state transit probability varies.

In Figure 2.5(a) and (b), we evaluate how the *single-sided* system dynamics affect the performance of our algorithm. In this experiment, the probability of staying in one state,  $P_{tt}$ , is fixed at 0.5, and the probability of staying in the other  $P_{ff}$  varies from 0.1 to 0.9, emulating how “sticky” that state is. Figure 2.5(a) and (b) shows that our algorithm consistently performs the best no matter what value that  $P_{ff}$  takes in terms of both the variable value estimation and the source reliability estimation.

In Figure 2.5(c) and (d), we set  $P_{tt} = P_{ff}$  and vary both from 0.1 to 0.9. The other parameters are set to the default values. Please note that since the initial ground bias  $d_j^T$  is set to 0.5, in this simulation the expected number of the  $T$  variables and the expected number of the  $F$  variables will always be the same, although the switching frequency changes. Again, our algorithm consistently performs the best.

In Figure 2.6, we evaluate the performance of our EM-VTC algorithm when the number of time-slots,  $H$ , in the sliding window considered for prediction varies from 1 to 10. Here we compare EM-VTC with EM-C1 that only considers the current time-slot. As shown in Figure 2.6, when

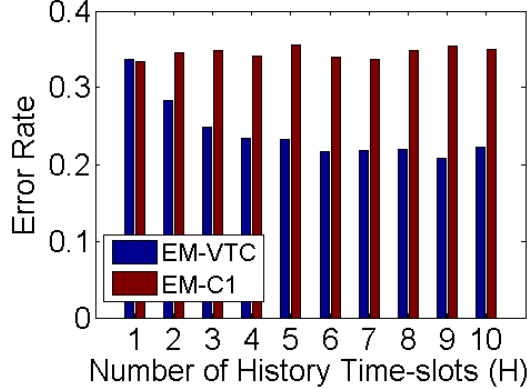


Figure 2.6: Variable value estimation performance as the number of history time-slots being considered varies.

we consider more history, the estimation is more accurate. However, the *marginal increase* in the estimation accuracy decreases.

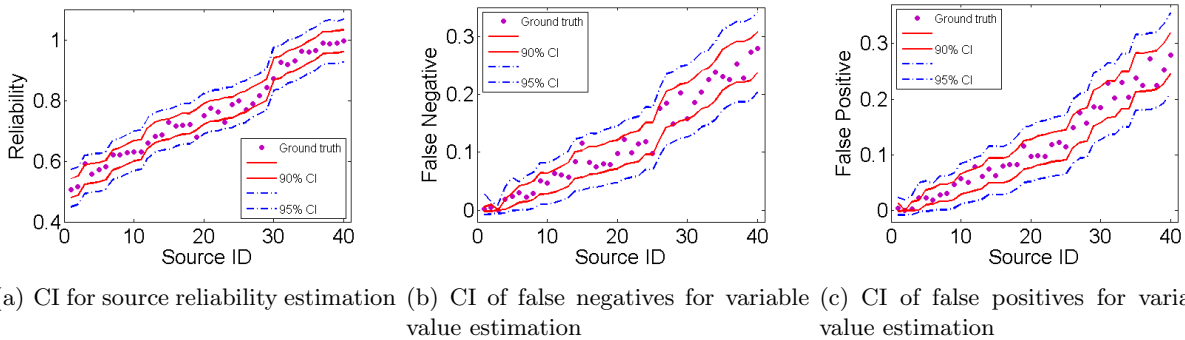


Figure 2.7: Performance bounds of EM-VTC in terms of confidence intervals.

In Figure 2.7 we study the performance bounds of our EM-VTC algorithm in terms of the confidence intervals (CI), which aims at validating our derived CRLB. In this experiment, we set the number of sources to 40, and the rest of the parameters to their default values. The area between the red solid lines is a 90% confidence interval of the estimators, and that between the blue dashed lines is a 95% confidence interval. The confidence intervals are computed from the CRLB as derived in Section 2.4. From CRLB, we can compute the variances of the false positive  $F_{i,F}$  ( $Var(F_{i,F}^{MLE})$ ) and false negative  $F_{i,T}$  ( $Var(F_{i,T}^{MLE})$ ) respectively. By Equation 2.6, the 90% confidence interval for the false negative is  $F_{i,T}^{MLE} \pm 1.65Var(F_{i,T}^{MLE})$ , and the 95% confidence interval for the false negative is  $F_{i,T}^{MLE} \pm 1.96Var(F_{i,T}^{MLE})$ . We can also compute the confidence intervals for the source reliability estimator. Please note that a 90% confidence interval means that

no more than 10% of the sample points are outside the confidence interval with high probability, while a 90% confidence interval means that no more than 5% of the sample points are not included in that interval with high probability.

From Figure 2.7(a), we observe that 3 sample points are out of the 90% CI, which validates our confidence interval since we allow no more than 4 points out of the 90% CI. For the 95% CI, all points are within it, which validates our confidence interval again.

In Figure 2.7(b), 1 points falls out of the 90% CI and no points falls out of the 95% CI. In Figure 2.7(c), 3 points fall out of the 90% CI and no points falls out of the 95% CI. Therefore, our confidence intervals are computed correctly, which also validates the derived CRLB.

We also run the above simulation for 100 times, and compute the average percentage of “bad” sample points among the whole sample, where “bad” sample points means the points fall out of the corresponding confidence interval. The results are shown in the Table 2.1, which also validates our confidence intervals and thus the derived CRLB.

Table 2.1: Percentage of sample points falling out of the corresponding confidence interval for different estimators.

	Reliability	False negative	False positive
90% CI	0.0648	0.0807	0.0865
95% CI	0.0093	0.0138	0.0158

## 2.6.2 A Real-world Case Study

In this subsection, we study the performance of our algorithm through a real-world crowd-sensing application that aims at assisting drivers to find the available street parking spots nearby. Recent work on assisting drivers in street parking relies on pre-deployed infrastructures (e.g. smart meters with communication capability to a remote center server [92]), or relies on special sensors to be embedded on cars (e.g. ultrasonic sensor on the side of a car [90]). We want to test whether a zero-infrastructure overhead crowd-sensing approach can be used in this application with the help of our algorithm.

In this case study, we recorded the availabilities of metered parking spots on two streets near the department of Computer Science at University of Illinois Urbana-Champaign for 8 days (from

April 11th, 2014 to April 18th, 2014) using webcams. 30 participants were involved in this case study to report the availabilities of the parking spots on the two streets from 2pm to 6pm on April 18th, 2014, and 2833 reports were collected. We deliberately picked the period from 2pm to 6pm is because in this period of time the availability of parking spots around the department of Computer Science has some clear time-varying pattern; students drive to attend the afternoon classes, and leave school for dinner around 5pm, then some of them drive back to school for self studies after dinner. The time-varying ground truth of the parking availability is shown in Figure 2.8.

We first define the time-slot, and define the availability (occupied/available) of a parking spot corresponding the slotted time. We partition the time into slots such that each time-slot is 10 minutes. Therefore, during our experiment on April 18th, there are 24 time-slots totally. Each parking spot is assigned with a corresponding variable with values  $T$  (representing that it is occupied) and  $F$  (representing that it is available). There are totally 15 parking spots in our experiment, therefore 15 variables are defined. In each time-slot, if more than half of the time a parking spot is occupied and the corresponding variable is assigned  $T$ , else the parking spot is defined free and the corresponding variable is assigned  $F$ .

Next, we define the source-claim matrix  $SC$ , and the trajectory probability matrix  $\mathcal{P}$  that are the inputs of our algorithm. Each report from the participants is associated with the time-slot during which it is generated. Both the sources and the events (variables) are indexed by integers. Therefore, we can naturally define the source-claim matrix  $SC$  to contain all the reports. In our case study, we only consider past two time-slots when estimating the variable values in the current time-slot, which means that the  $H$  is set 2 here. The state transition is modeled by the trajectory probability matrix  $\mathcal{P}$  where each element is a joint probability of the current value and the last time-slot value of a variable. The joint probabilities are learned from history data between 2pm and 6pm from April 11th to April 17th. Please note that we did not use the data on April 18th to learn the trajectory probability matrix. In this case study, we set all the variable share the same trajectory probability matrix  $\mathcal{P}$ . Please note that this is not exactly accurate, since different variables might have different dynamics behaviors. However, our evaluation results show that our EM-VTC algorithm still achieves an accurate estimation under this imperfect setting.

We compare with 4 baseline algorithms: (1) **EM-C1** as described in the previous subsection,



(2) **EM-R1** that is also described previous, (3) **Voting**, and (4) **History** where we only use the state in the last time-slot to estimate the current state for each variable. The experiment works in a “sliding-window” style: We start from estimating the variable values of the second time-slot (2:10pm-2:20pm) using the data (the source reports) in the first time-slot and the second time-slot, then move on to estimate the system state in the third time-slot (2:20pm-2:30pm) using data from the second and the third time-slots, and so on.

The ground truth is shown in Figure 2.8. From Figure 2.8, we can observe that the system state (the availability of parking spots) did change overtime. During the dinner time (from 5:00pm to 6:00pm), the system state changes more frequently than that during the working time (before 5:00pm).

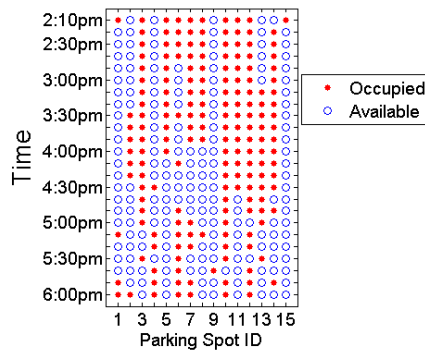


Figure 2.8: The ground truth of the availability of parking spots.

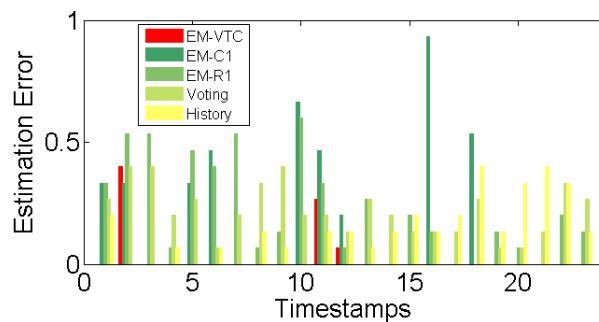


Figure 2.9: The performance of variable state estimation.

The estimation results are presented in Figure 2.9. In the figure, the  $x$ -axis represents the time-slots that are indexed using integers, and the  $y$ -axis is the estimation error. The red bar shows the estimation error of our EM-VTC algorithm. From Figure 2.9, we hardly observe the red bars,

which shows that our solution in most of the time has *zero* estimation errors. The average errors are summarized in Table 2.2.

Table 2.2: Estimation error in parking case study

Algorithm	EM-VTC	EM-C1	EM-R1	Voting	History
Est. Error	<b>0.0319</b>	0.1855	0.2261	0.2203	0.1420

Please note that the EM-C1 and EM-R1 algorithms perform worse than the History heuristic that using the variable value in the last time-slot to estimate its current value, which is because that our parking data has high correlation between the last time-slot and the current time-slot as shown in Figure 2.8, and that those two algorithms only use the data in the current time-slot which is not enough to estimate the system state with high accuracy. Our algorithm uses more data and considers the system state trajectory, therefore performs the best.

## 2.7 Related Work

Reliability is a critical requirement for cyber-physical systems [75]. Much prior research focused on temporal and functional reliability of CPS applications. For example, Eidson et al. [32] presented a programming model called PTIDES for the reliable timing control of the cyber-physical systems. Clarke et al. [27] applied formal analysis technique on autonomous transportation control for cars, trains, and aircraft. Faza et al. [35] suggested the use of software fault injection combined with physical failures in identifying integrated cyber-physical failure scenarios for the smart grid. Sha et al. [106] developed a hybrid approach that combines fault-tolerant architectures with formal verification to support the design of safe and robust cyber-physical systems. Different from prior efforts, this chapter addresses the *data reliability* challenge that arises in cyber-physical systems operating in social spaces, where significant amounts of data are collected from the “crowd”. In this scenario, the “crowd” functions as a noisy sensor of large amounts of physical states. A state estimator is needed to optimally recover reliable data and accurately estimate error bounds.

The work is motivated by human-in-the-loop cyber-physical systems; a challenging and promising class of CPS [104]. Many examples of such systems appear in recent literature, where humans plays important roles in feedback loops, such as operator, load, disturbance, or controlled plant. For example, Lu et al. developed a smart thermostat system to monitor the occupancy and sleep

pattern of the residents and turned off the HVAC when not needed [86]. Huang et al. designed a mathematical model to determine the insulin injection by closely monitoring glucose level when it reaches a threshold, a key challenge to design an artificial pancreas [59]. Our work is complementary to the efforts mentioned above in that we investigate the role of humans as *sensors*. Hence, we are interested in addressing the reliability challenge that ensues when data is obtained from unvetted sources, where the reliability of data sources is unknown and the states of observed variables may evolve over time.

Our work is related to the system state estimation problem with unreliable sensors in CPS. Sinopoli et al. designed a discrete Kalman filter to estimate the system state when the sensor reports were intermittent [109]. Ishwar et al. estimated the states of the sink node with noisy sensor nodes in WSNs [62]. Mathematical tools were proposed by Schenato et al. to control and estimate the states of physical systems on top of a lossy network [103]. Masazade et al. proposed a probabilistic transmission scheme to near-optimally estimate the system parameter in WSN with sensing noises [89]. However, the sensor error is either assumed known [89], or generated from a common distribution with known parameters [62, 103, 109]. In contrast, in crowd-sensing applications, not only do we assume that the error distribution is unknown but also that each (human) sensor has its own possibly different error distribution. Therefore, none of the prior work is applicable in crowd-sensing.

The importance of crowd-sensing as a possible data input in cyber-physical systems is attributed to the proliferation of mobile sensors owned by individuals (e.g., smart phones) and the pervasive Internet connectivity. Hence, humans can be sensor carriers [73] (e.g., opportunistic sensing), sensor operators [23], or sensor themselves [127]. Wang et al. proposed data prioritizing schemes to maximize the data coverage in an information space [139] and to maximize the collected data diversity [134, 135] for crowd-sensing applications. An early overview of crowd-sensing applications is described in [7]. Examples of early systems include CenWits [58], CarTel [61], BikeNet [34], and CabSense [105].

Recently, the problem of fact-finding, which refers to ascertaining correctness of data from *sources of unknown reliability*, has drawn significant attention. It has been studied extensively in the data mining and machine learning communities. One of the earliest efforts is Hubs and

Authorities [70] that presented a basic fact-finder where the belief in a claim and the truthfulness of a source are jointly computed in a simple iterative fashion. Later on, Yin et al. introduced TruthFinder as an unsupervised fact-finder for trust analysis on a providers-facts network [152]. Pasternack et al. extended the fact-finder framework by incorporating prior knowledge into the analysis and proposed several extended algorithms: Average.Log, Investment, and Pooled Investment [97]. Su et al. proposed supervised learning frameworks to improve the quality of aggregated decision in sensing systems [111–113]. Additional efforts were spent in order to enhance the basic fact-finding framework by incorporating analysis on properties or dependencies within claims or sources.

The above work is heuristic in nature; it does not offer optimality properties and does not allow computation of error bounds. The latter is an important requirement for a state estimator in CPS applications. Towards an optimal solution, Wang et al., proposed a Maximum Likelihood Estimation (MLE) framework [129,131] that offers a joint estimation on source reliability and claim correctness based on a set of general simplifying assumptions. In their following work, Wang et al. further extended the framework to handle streaming data [125] and source dependencies [127]. The approach was compared to several of the aforementioned fact-finders and was shown to outperform them in estimation accuracy, while also offering error bounds. However, their work is unsuited for CPS, since it did not consider the evolving event states which is common in CPS applications.

The algorithms proposed in this chapter extend the above MLE based framework by being the first to study the state estimation problem (with unknown source reliability) in crowd-sensing applications with time-varying system states. We further derive a Cramer-Rao lower bound for the resulting novel maximum likelihood estimator.

## 2.8 Conclusions

In this chapter, we developed a state estimator for crowd-sensing applications, where humans act as data sources reporting observed variables in a dynamic environment. We demonstrated how a model of environmental dynamics can significantly enhance our ability to estimate correct state even though the reliability of observers is not known. We also analytically studied its performance by deriving a Cramer-Rao lower bound for our estimator. Our solution was evaluated using both

simulations and a real-world crowd-sensing application. The results show that the solution outperforms prior approaches that do not properly account for differences in source reliability (e.g., voting) or do not properly leverage knowledge of the system model.

## 2.9 Derivations

In this section, we provide the details of the math derivations.

### 2.9.1 Deriving the E-step

We plug-in the likelihood function, given by Equation (2.4), into Equation (2.2) to derive the E-step.

$$\begin{aligned}
Q(\theta|\theta^{(n)}) &= E_{Z|x,\theta^{(n)}}[\log L(\theta; x, Z)] \\
&= \sum_{j \in \mathcal{C}} E_{Z_j|x_j,\theta^{(n)}} \left[ \log \left\{ \sum_{z_j \in \Lambda^H} p(z_j) \mathbf{1}_{\{Z_j=z_j\}} \prod_{i \in \mathcal{S}} \prod_{k=1}^H \alpha_{i,j,k} \right\} \right] \\
&= \sum_{j \in \mathcal{C}} \sum_{z_j \in \Lambda^H} p(z_j|x_j, \theta^{(n)}) \left\{ \log p(z_j) + \sum_{i \in \mathcal{S}} \sum_{k=1}^H \log \alpha_{i,j,k} \right\}
\end{aligned} \tag{2.8}$$

, where

$$\begin{aligned}
p(z_j|x_j, \theta^{(n)}) &= \frac{p(x_j, z_j|\theta^{(n)})}{p(x_j|\theta^{(n)})} \\
&= \frac{p(x_j|z_j, \theta^{(n)}) \cdot p(z_j|\theta^{(n)})}{\sum_{z_j \in \Lambda^H} p(x_j|z_j, \theta^{(n)}) \cdot p(z_j|\theta^{(n)})} \\
&= \frac{p(z_j) \prod_{i \in \mathcal{S}} \prod_{k=1}^H \alpha_{i,j,k}^{(n)}}{\sum_{z_j \in \Lambda^H} p(z_j) \prod_{i \in \mathcal{S}} \prod_{k=1}^H \alpha_{i,j,k}^{(n)}}.
\end{aligned} \tag{2.9}$$

Please note that in the  $Q$  function, the parameters are embedded into the  $\alpha_{i,j,k}$  that is assigned different values under different  $(i, j, k)$  settings as defined in Equation (2.5). Also note that  $p(z_j|x_j, \theta^{(n)})$  is a constant with respect to  $\theta$ .

### 2.9.2 Deriving the M-step

In the M-step, we set the partial derivatives of the  $Q$  function to 0 to get the  $\theta^*$  that maximizes the value of  $Q$ . That is by solving  $\frac{\partial}{\partial T_{i,v}}Q = 0$ , and  $\frac{\partial}{\partial F_{i,v}}Q = 0$ , we can get  $T_{i,v}^*$  and  $F_{i,v}^*$ , for each  $i \in \mathcal{S}$  and  $v \in \{T, F\}$ .

$$\begin{aligned} T_{i,v}^* &= \frac{\sum_{j \in \mathcal{C}} \sum_{k: SC_{i,j,k}=v} Z_v^{(n)}(j, k)}{\sum_{j \in \mathcal{C}} \sum_{k=1}^H Z_v^{(n)}(j, k)} \\ F_{i,v}^* &= \frac{\sum_{j \in \mathcal{C}} \sum_{k: SC_{i,j,k}=\bar{v}} Z_v^{(n)}(j, k)}{\sum_{j \in \mathcal{C}} \sum_{k=1}^H Z_v^{(n)}(j, k)} \end{aligned} \quad (2.10)$$

, where  $Z_v^{(n)}(j, k)$  is defined as:

$$Z_v^{(n)}(j, k) = \sum_{z_j: z_{j,k}=v} p(z_j | x_j, \theta^{(n)}). \quad (2.11)$$

The parameter  $\theta^*$  is used in our EM-VTC algorithm as shown in Algorithm 1.

### 2.9.3 Deriving Cramer-Rao Lower Bound

By definition, CRLB is the inverse of the Fisher information matrix  $J(\theta)$ , where  $J(\theta) = E_X[\nabla_{\theta} \log p(X|\theta) \nabla_{\theta}^H \log p(X|\theta)]$ . Here matrix  $X^H$  is the conjugate transpose of matrix  $X$ . The CRLB derived in this subsection is asymptotic by assuming that the values of variables are correctly estimated by the EM algorithm. This assumption is valid when the number of sources is high, as shown in [128]. The loglikelihood function under this assumption is  $\ell_{em}(\theta; x)$ , as defined in

Equation (2.12).

$$\begin{aligned}
\ell_{em}(\theta; \mathbf{x}) = & \sum_{j \in \mathcal{C}} \sum_{k=1}^H \left\{ \mathbf{1}_{\{z_{j,k}=T\}} \left( \sum_{i \in \mathcal{S}} \left( \mathbf{1}_{\{x_{i,j,k}=T\}} \log T_{i,T} \right. \right. \right. \\
& + \mathbf{1}_{\{x_{i,j,k}=F\}} \log F_{i,T} + \mathbf{1}_{\{x_{i,j,k}=U\}} \log(1 - T_{i,T} - F_{i,T}) \left. \left. \left. \right) \right) \right. \\
& + \mathbf{1}_{\{z_{j,k}=F\}} \left( \sum_{i \in \mathcal{S}} \left( \mathbf{1}_{\{x_{i,j,k}=F\}} \log T_{i,F} + \mathbf{1}_{\{x_{i,j,k}=T\}} \log F_{i,F} \right. \right. \\
& \left. \left. \left. + \mathbf{1}_{\{x_{i,j,k}=U\}} \log(1 - T_{i,F} - F_{i,F}) \right) \right) \right\}. \tag{2.12}
\end{aligned}$$

We compute the Fisher information matrix from its definition, similar as [128]. The parameters in the loglikelihood is order such that the  $T_{i,T}$ 's come first, then  $T_{i,F}$ 's, following by  $F_{i,T}$ 's, and finally  $F_{i,F}$ 's. Let denote the number of sources by  $M$ , so  $M = |\mathcal{S}|$ . So, we have  $4M$  parameters in total, and the size of the Fisher information matrix is  $4M \times 4M$ . Let's denote the  $n$ -th parameter by  $\theta_n$ . For example, when  $n \leq M$ ,  $\theta_n = T_{i,T}$  where  $i = n$ , and when  $n \in [M + 1, 2M]$ ,  $\theta_n = T_{i,F}$  where  $i = n - M$ , so on. We denote the total number of variables by  $N$ , i.e.,  $N = |\mathcal{C}|$ . Let  $J(\theta)_{m,n}$  denote the element in the  $m$ -th row and  $n$ -th column of the Fisher information matrix  $J(\theta)$ . The

Fisher information is defined in Equation (2.13).

$$\begin{aligned}
J(\theta)_{m,n} &= \begin{cases} -E[\frac{\partial^2}{\partial\theta_m\partial\theta_n}\ell_{em}] = 0 & m \neq n \\ -E[\frac{\partial^2}{\partial T_{i,T}^2}\ell_{em}] & 0 < m = n \leq M, \\ & \text{where } i = n \\ -E[\frac{\partial^2}{\partial T_{i,F}^2}\ell_{em}] & M < m = n \leq 2M, \\ & \text{where } i = n - M \\ -E[\frac{\partial^2}{\partial F_{i,T}^2}\ell_{em}] & 2M < m = n \leq 3M, \\ & \text{where } i = n - 2M \\ -E[\frac{\partial^2}{\partial F_{i,F}^2}\ell_{em}] & 3M < m = n \leq 4M, \\ & \text{where } i = n - 3M \end{cases} \\
&= \begin{cases} 0 & m \neq n \\ \frac{d^T \cdot N \cdot H(1-F_{i,T})}{T_{i,T}(1-T_{i,T}-F_{i,T})} & 0 < m = n \leq M, \\ & \text{where } i = n \\ \frac{d^F \cdot N \cdot H(1-F_{i,F})}{T_{i,F}(1-T_{i,F}-F_{i,F})} & M < m = n \leq 2M, \\ & \text{where } i = n - M \\ \frac{d^T \cdot N \cdot H(1-T_{i,T})}{F_{i,T}(1-T_{i,T}-F_{i,T})} & 2M < m = n \leq 3M, \\ & \text{where } i = n - 2M \\ \frac{d^F \cdot N \cdot H(1-T_{i,F})}{F_{i,F}(1-T_{i,F}-F_{i,F})} & 3M < m = n \leq 4M, \\ & \text{where } i = n - 3M \end{cases} \tag{2.13}
\end{aligned}$$

The Fisher information matrix  $J(\theta)$  is actually diagonal. Therefore, the CRLB defined by  $J^{-1}(\theta)$  can be computed efficiently by inverting each non-zero element in  $J(\theta)$ .

Note that the CRLB should be computed by the actual ground truth value of the parameters. However, in real-world applications, due to lack of the ground truth of those parameters, we feed the maximum likelihood estimations of the parameters to approximate the CRLB, i.e. CRLB is  $J^{-1}(\hat{\theta}_{MLE})$ .



## Chapter 3

# Fact-finding with Interdependent Variables

This chapter extends prior work by offering scalable algorithms for exploiting known dependency graphs between observed variables to improve the quality of ground truth estimation for social sensing applications.

Consider, for example, post-disaster scenarios, where significant portions of a city’s infrastructure are disrupted. Communication resources are scarce, rumors abound, and means to verify reported observations are not readily available. Survivors report to a central unit the locations of damage and outages, so that help may be sent. Some reports are accurate, but much misinformation exists as well. Not knowing the individual sources in advance, it may be hard to tell which reports are more reliable. Simply counting the number of reports that agree on the facts (called *voting* in prior literature) is not always a good measure of fact correctness, as different sources may have different reliability. Hence, a different weight should be associated with each report (or vote), but that weight is not known in advance.

Prior work of the authors addressed the above problem when the reported observations are independent [131] and considered the case where second-hand observations were reported by other than the original sources [127]. In work that comes closest to this work, an algorithm was presented for the case, where the reported variables are correlated [126]. Unfortunately, the computational and representational complexity of the correlation was exponential in the number of correlated variables. Hence, in practice, it was not feasible to consider more than a small number of correlated variables at a time.

In sharp contrast to the above results, in this chapter, we consider the case where reported variables have non-trivial dependency graphs. For example, upon the occurrence of a natural or man-made disaster, flooding, traffic conditions, outages, or structural damage in different parts of a city may be correlated at large scale. Furthermore, the structure of the correlations might

be partially known. Areas of the same low elevation may get flooded together. Nearby parts of the same main road may see correlated traffic conditions. Buildings on the same power line may suffer correlated power outages. Gas stations that have the same supplier might have correlated availability of gas. Correlations (e.g., among failures) can also shed light on the root cause. For example, in a situation where a supply line simultaneously feeds several consumers, a failure in the line will result in correlated failures at the downstream consumers. If the topology of the supply lines is known, so is the correlation structure among expected consumer failures. If consumers build products that need multiple suppliers, knowing the pattern of the correlated consumer failures can give strong evidence as to which one of the suppliers may have failed.

Clearly, if the aforementioned correlation structure is not known, we cannot use this approach. Scenarios where correlations structures are not known can be addressed using prior work that simply views the underlying variables as uncorrelated [131]. This chapter offers performance and accuracy improvements in the special (but important) case, where hypotheses regarding possible correlations are indeed available. Exploiting such correlations reduces problem dimensionality, allowing us to infer the state of points of interest more accurately and in a more computationally efficient manner.

What complicates the problem (in social sensing scenarios) is that actual state of the underlying physical system is not accurately known. All we have are reports from sources of unknown reliability. This chapter develops the first scalable algorithm that takes advantage of the structure of correlations between (large numbers of) observed variables to better reconstruct ground truth from reports of such unreliable sources. We show that our algorithm has better accuracy than prior schemes such as voting and maximum likelihood schemes based on independent observations [131]. It also significantly outperforms, in terms of scalability, previous work that is exponential in dependency structures [127].

The general idea behind the scalability of the new scheme lies in exploiting conditional independence, when one catalyst independently causes each of multiple consequences to occur. Identification of such conditional independence relations significantly simplifies reasoning about the joint correlations between observed values, thus simplifying the exploitation of such correlations in state estimation algorithms. Although previous work [126] considers correlated variables in social

sensing applications, it does not exploit conditional independence. The computational complexity of the previous solution increases exponentially in the number of correlated variables, which makes it applicable only to applications with a small number of such variables. By modeling the structural correlations of variables as a Bayesian network and exploiting conditional independence, our algorithm is more computationally efficient. Its computational complexity depends on the size of the largest clique (i.e., complete sub-graph) in the Bayesian network, while the complexity of the previous solution [126] depends on the total number of nodes in the Bayesian network making the latter intractable for applications with a large number of correlated variables.

The rest of the chapter is organized as follows. We formulate our problem in Section 3.1. In Section 3.2, we argue that our solution is general and can be applied to solve previous social sensing challenges by showing that all the previous models are special cases of our Bayesian model. We propose our solution in Section 3.3 and evaluate our algorithm in Section 3.4. A literature review is presented in Section 3.5. The chapter concludes in Section 3.6.

### 3.1 Problem Formulation

Social sensing differs from sensing paradigms that use in-field physical sensors (e.g. Wireless Networked Sensing [100]) in that it exploits sensors in social spaces. Examples include sensor-rich mobile devices like smartphones, tablets, and other wearables, as well as using *humans as sensors*. The involvement of humans in the sensing process enables an application to directly sense variables with higher-level semantics than what traditional sensors may measure. However, unlike physical devices, which are usually reliable or have the same error distribution, the reliability of human sources is more *heterogeneous* and may be *unknown a priori*. This source reliability challenge in social-sensing systems was recently articulated by Wang et al. [131]. Solutions that estimate source reliability were improved in follow-up publications [126, 127, 141].

In recent work, the authors modeled human sources as *binary* sensors, reporting events of interest. The rationale behind the binary model is that humans are much better at categorizing classes of observations than at estimating precise values. For example, it is much easier to tell whether a room is warm or not than to tell its exact temperature. Binary variables can be easily extended to multivalued ones [141], which makes the binary model versatile. In this paper, we

adopt the binary model and assume that a group of human sources, denoted by  $\mathcal{S}$ , participate in a sensing application to report values of binary variables, we call the *event variables*. For example, they may report the existence or absence of gas at a set of gas stations after a hurricane. These variables are collectively denoted by  $\mathcal{C}$ . The goal of this paper is to jointly estimate both the source reliability values and ground-truth measured variable values, given only the string of noisy reports. In contrast to prior work, we assume that the underlying variables are structurally correlated at scale. The question addressed in this paper is how to incorporate knowledge of these correlations into the analysis.

### 3.1.1 Modeling Interdependent Event Variables

In previous work, event variables were either assumed to be independent [127, 131] or were partitioned into groups of small size [126, 141] with no dependencies among groups. Solution complexity grew exponentially with the maximum group size. In practice, it is not uncommon that all or a large portion of event variables are interdependent. For example, in an application that monitors traffic conditions in a city, pertinent variables might denote weather conditions (e.g., snowy or rainy weather), local entertainment events that impact traffic (e.g., football games or concerts), road surface conditions (e.g., potholes on road surfaces), and traffic speed, among others. These variables are correlated. Bad weather results in slow traffic. So do the local entertainment events and bad road surfaces. Traffic congestion on one road segment might cause congestion on another road segment. The pervasive dependencies among variables make previous work (e.g., Wang et al. [126]) inapplicable due to intractability, thus calling for a better model to handle them. This paper is the *first* to study the reliable social-sensing problem with interdependent variables, at scale.

Our solutions are based on the insight that although independence is uncommon in real applications, *conditional independence* does often arise. As stated in [88], dependencies usually expose some structure in reality. The dependency structure encodes conditional independence, that can be leveraged to greatly simplify the estimation of values of variables. In the previous application example, given that the weather is snowy, the resulting traffic congestion on two road segments can be assumed to be conditionally independent. Both are caused by snowy weather but neither

is affecting the other (assuming they are sufficiently far apart). However, without knowing the state of the weather, we are not able to assume that congestion on both segments is independent. Measuring congestion on those segments, it will tend to be correlated (in the presence of snow events).

In this paper, we model dependencies among variables by a Bayesian network [94]. The underlying structure of a Bayesian network is a directed acyclic graph (DAG) in which each node  $V$  corresponds to a variable,  $v$ , and each arc  $U \rightarrow V$  denotes a conditional dependence such that the value of variable  $v$  is dependent on the value of  $u$ . The Bayesian network is a natural way to model causal relations between variables. Since Bayesian networks are well-established tools for statistical inference, we can leverage prior results to solve our reliable social-sensing problem.

Of course, in some cases, the underlying dependences can form a complete graph in which any pair of variables are *directly* interdependent. In this extreme situation, there would be no efficient inference algorithm with computational complexity inferior to  $\Theta(2^N)$ , where  $N$  denotes the total number of variables (i.e.,  $|\mathcal{C}|$ ). All inferences should be made by considering the joint distribution of all variables. However, as stated in [88], the complete graph structure does not often happen in real applications, and so we are not interested in this extreme case.

### 3.1.2 Categorized Source Reliability

Although previous work in social sensing assumes that sources have different reliability, for a specific source, its reliability is assumed to be fixed (e.g., [126, 127, 131]). This fixed-reliability assumption does not hold in many practical scenarios. For example, a diabetic person who is in need of insulin might be a better source to ask about pharmacies that remained open after a natural disaster, than a person who is not in need of medication. The same diabetic person might not be a good source to ask about gas stations that are open, if the person does not own a car. In the above scenario, if we assume that a single source has the same reliability in reporting all types of variables, the performance of estimating the ground truth of these variables might be degraded. To make the source reliability model more practical, and thus the estimation more accurate, we assume that source reliability differs depending on the variable reported. Measured variables are classified into different categories. Source reliability is computed separately for each category. We call it

*categorized source reliability.*

With the categorized-source-reliability model, the reliability of each source is represented by a vector (where each element is corresponding to the reliability for some reported category of variables, rather than a scalar as in previous work. Please note that the previous reliability model is a special case of our model as a single-element vector.

### 3.1.3 Problem Definition

Next, we formally define our reliable social-sensing problem with interdependent variable at scale. We denote the  $j$ -th measured variable by  $C_j$ , and  $C_j$  is assumed to be binary. More specifically,  $C_j \in \{T, F\}$  where  $T$  represents True (e.g., “Yes, the room is warm”), and  $F$  represents False (e.g., “No, the room is not warm”). One can think of each variable as the output of a different application-specific True/False predicate about physical world state. Each variable  $C_j$  belongs to some category  $\ell$ , denoted by  ${}^\ell C_j$ . We use  $\mathcal{L}$  to denote the category set.

In social-sensing, a source reports the values of variables. We call those reports, *claims*. We use a matrix  $SC$  to represent the claims made by all sources  $\mathcal{S}$  about all variables  $\mathcal{C}$ . We call it the *source-claim matrix*. In the source-claim matrix, an element  $SC_{i,j} = v$  means that the source  $S_i$  claims that the value of variable  $C_j$  is  $v$ . It is also possible that a source does not claim any value for some variable, in which case the corresponding item in the source-claim matrix  $SC_{i,j}$  is assigned value  $U$  (short for “Unknown”) meaning that the source did not report anything about this variable. Therefore, in the source-claim matrix,  $SC$ , each item  $SC_{i,j}$  has three possible values  $T$ ,  $F$  and  $U$ .

We define the reliability of source  $S_i$  in reporting values of variables of category  $\ell$  as the probability that variables belonging to that category indeed have the values that the source claims they do. In other words, it is the probability that  ${}^\ell C_j = v$ , given that  $SC_{i,j} = v$ . In the following, we shall use the short notation  $X^v$  to denote that the variable  $X$  is of value  $v$  (i.e.,  $X = v$ ). Let  ${}^\ell t_i$  denote the reliability of source  $S_i$  in reporting values of variables of category  $\ell$ . We formally define the source reliability as follows:

$${}^\ell t_i = \Pr \left( {}^\ell C_j^v | SC_{i,j}^v \right). \quad (3.1)$$

Let  ${}^\ell T_i^v$  denote the probability that source  $S_i$  reports the value of variable  ${}^\ell C_j$  correctly. In

Table 3.1: The summary of notations

Set of sources	$\mathcal{S}$
Set of variables	$\mathcal{C}$
Binary variable, $j$	$C_j$
Variable $X$ of category $\ell$	${}^\ell X$
Binary value set	$\{T, F\}$
Source-claim matrix	$SC$
Source reliability	${}^\ell t_i = \Pr\left({}^\ell C_j^v   SC_{i,j}^v\right)$
Correctness probability	${}^\ell T_i^v = \Pr\left(SC_{i,j}^v   {}^\ell C_j^v\right)$
Error probability	${}^\ell F_i^v = \Pr\left(SC_{i,j}^{\bar{v}}   {}^\ell C_j^v\right)$

other words, the probability that  $S_i$  reports value  $v$  for variable  ${}^\ell C_j$  given that its value is really  $v$ . Furthermore, let  ${}^\ell F_i^v$  denote the probability of an incorrect report by  $S_i$ . In other words, it is the probability that  $S_i$  reports that  ${}^\ell C_j$  has value  $\bar{v}$  given that its value is  $v$ . Here  $\bar{x}$  is the complement of  $x$  ( $\bar{T} = F$  and  $\bar{F} = T$ ).  ${}^\ell T_i^v$  and  ${}^\ell F_i^v$  are formally defined below:

$${}^\ell T_i^v = \Pr\left(SC_{i,j}^v | {}^\ell C_j^v\right), \quad {}^\ell F_i^v = \Pr\left(SC_{i,j}^{\bar{v}} | {}^\ell C_j^v\right). \quad (3.2)$$

Note that,  ${}^\ell T_i^v + {}^\ell F_i^v \leq 1$ , since it is possible that the source  $S_i$  does not report anything of a variable. Therefore, we have:

$$1 - {}^\ell T_i^v - {}^\ell F_i^v = \Pr\left(SC_{i,j}^U | {}^\ell C_j^v\right). \quad (3.3)$$

We denote the prior probability that source  $S_i$  makes a positive claim (i.e., claims a value  $T$ ) by  $s_i^T$  and denote the prior probability that source  $S_i$  makes a negative claim (i.e., claims a value  $F$ ) by  $s_i^F$ . We denote the prior probability that variable  $C_j$  is of value  $v$  by  $d^v$ . By the Bayesian theorem, we have:

$${}^\ell T_i^v = \frac{{}^\ell t_i \cdot s_i^v}{{}^\ell d^v}, \quad {}^\ell F_i^v = \frac{(1 - {}^\ell t_i) \cdot s_i^{\bar{v}}}{{}^\ell d^v}. \quad (3.4)$$

Table 3.1 summarizes the introduced notations.

The dependencies between variables are given by a Bayesian network. In the underlying dependency structure of the Bayesian network (i.e., a DAG, denoted by  $G$ ), each vertex  $V_j$  corresponds to a variable  $C_j$ , and each arc  $V_j \rightarrow V_k$  corresponds to a causal relation between variables  $C_j$  and

$C_k$  in which the value of  $C_k$  is dependent on that of  $C_j$ . For any variable  $C_j$ , we use  $\text{par}(C_j)$  to denote the set of variables on whom the value of  $C_j$  directly depends (i.e., not including transitive dependencies). Since the causal relation is encoded in the Bayesian network,  $G$ , there is an arc from each node denoting a variable in  $\text{par}(C_j)$  to the node denoting  $C_j$  in  $G$ . Please note that each node  $V_j$  in the Bayesian network is associated with a probability function that takes, as input, a particular set of values of  $\text{par}(C_j)$ , and gives, as output, the probability of  $C_j$  being true. In other words, given the Bayesian network, we know the conditional probability  $\Pr({}^\ell C_j^v | \text{par}(C_j))$  for any event variable  $C_j$ . We assume that the Bayesian network is known from application context (e.g., we might have a map that says which outlet depends on which suppliers), or can be empirically learned from historic data by the algorithms such as those introduced in [94]. Hence, our estimation algorithm assumes that the Bayesian network is an input.

Finally, we formulate our reliable social-sensing problem as follows: *Given a source-claim matrix  $SC$ , a category label  $\ell$  for each reported variable, and a Bayesian network  $G$ , encoding the dependencies among variables, how to jointly estimate both the reliability of each source and the true value of each variable in a computationally efficient way?* Here an algorithm is defined as efficient if its time complexity is sub-linear to the exponential (i.e.,  $o(2^{|G|})$ ) for a Bayesian network that is not a complete graph, where  $|G|$  is the total number of nodes in the Bayesian network.

## 3.2 Generalization of Previous Models

Before we propose our estimation algorithm, in this section, we show that our social-sensing model is more general than those proposed in our previous work [126, 127, 131, 141]. In other words, the social-sensing models proposed by the previous work are all special cases of ours. Therefore, thanks to the general model, the estimation algorithm proposed in our work can be directly applied to any of the problems defined in the previous work.

First, we show that the model proposed by Wang et al. [131] is a special case of our model. In their model, both the event variables and the sources were assumed independent. Thus, the structure of a Bayesian network for this model is just a DAG with arcs only connecting the event variable and its corresponding claims from the sources, as shown in Figure 3.1(a).

In [126], the model was extended to consider physical constraints of the sources (i.e., a variable



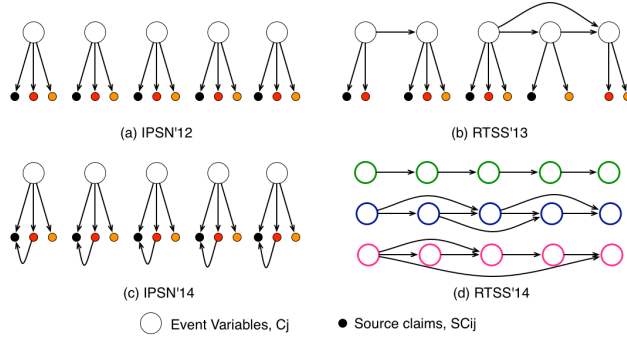


Figure 3.1: Model connections with previous work.

might not be observed by some source), as well as correlated variables that fall into a bunch of independent groups. The structure of a Bayesian network for this model has disjoint cliques (complete sub-graphs) where each clique has a constant number of nodes, as shown in Figure 3.1(b). In this figure, there are two cliques; one has two nodes and the other has three. Furthermore, since the physical constraints of the sources are considered, there are some variable that can only be observed by a subset of sources. Therefore, in the corresponding Bayesian network, if the variable is not observed by some source, then there is no arc between them in the DAG (such as the rightmost one that is observed only by the red source and the orange source).

Source dependencies were considered in [127], where a claim made by a source can either be original or be re-tweeted from some other source. The variables are assumed independent. The corresponding Bayesian network for this model is shown as in Figure 3.1(c). If a source  $i$  is dependent on some other source  $j$ , which means that the claim made by  $j$  actually affects that made by  $i$  as shown in Figure 3.1(c). In Figure 3.1(c), the black source is dependent on the red source, therefore there is an arc from the red node to the black node for each event variable. The arc from the  $SC_{j_i}$  to  $SC_{i_i}$  is enough to model this dependence.

Recently, Wang et al. [141] further extended the previous model by considering time-varying ground truth, in which the value of each variable could vary over time. They proved that given the evolving trajectory of each variable, by considering a sliding window of past states, the estimation result is greatly improved compared with estimators that only consider the current state. Their model can be represented by a dynamic Bayesian network with time-varying dependency structures. Figure 3.1(d) gives an example of a Bayesian network representation of their model. Here, we omit

the vertices corresponding to claims made by sources  $SC_{i,j}$ . In the figure, the variable nodes with the same color are corresponding to a variable in different time-slots. The evolving trajectory of each variable can be represented by some dependency structure among all its history states, as shown in Figure 3.1(d).

The above examples illustrate how previous models can be special cases of our model. Therefore, once we solved the problem with the general model, using the same algorithm, we are able to solve all the previous problems as defined in [126, 127, 131, 141]. We propose our estimation algorithm in the following section.

### 3.3 Estimating the States of Interdependent Variables

In this section, we describe our ground truth estimation algorithm for social-sensing applications with the interdependent variables at scale. Our algorithm follows the Expectation-Maximization (EM) framework [30] that jointly estimates (1) the reliability of each source, and (2) the ground truth value of each reported variable. Here we assume that sources independently make claims; for dependent sources, we can apply the algorithm proposed in [127]. We call the proposed algorithm EM-CAT (EM algorithm with CATegory-specific source reliability).

#### 3.3.1 Defining Estimator Parameters and Likelihood Function

EM is a classical machine-learning algorithm to find the maximum-likelihood estimates of parameters in a statistical model, when the likelihood function contains latent variables [30]. To apply the EM algorithm, we first need to define the likelihood function  $L(\theta; x, Z)$ , where  $\theta$  is the parameter vector,  $x$  denotes the observed data, and  $Z$  denotes the latent variables. The EM algorithm iteratively refines the parameters by the following formula until they converge:

$$\theta^{(n+1)} = \arg \max_{\theta} E_{Z|x, \theta^{(n)}} [\log L(\theta; x, Z)] \quad (3.5)$$

The above computation can be further partitioned into an *E-step* that computes the conditional expectation of the latent variable vector  $Z$  (i.e.,  $Q(\theta) = E_{Z|x, \theta^n} [\log L(\theta; x, Z)]$ ), and an *M-step* that finds the parameters  $\theta$  that maximize the expectation (i.e.,  $\theta^{(n+1)} = \arg \max_{\theta} Q(\theta)$ ). In our

problem, we define the parameter vector  $\theta$  as:

$$\theta = \{(\ell T_i^v, F_i^v) | \forall i \in \mathcal{S}, v \in \Lambda, \ell \in \mathcal{L}\}$$

where  $\Lambda = \{T, F\}$  denotes the set of binary values and  $\mathcal{L}$  is the set of event categories. The data  $x$  is defined as the observations in the source-claim matrix  $SC$ , and the latent variable vector  $Z$  is defined as the values of the event variables.

After defining  $\theta, x$  and  $Z$ , the likelihood function is derived as follows:

$$\begin{aligned} L(\theta; x, Z) &= \Pr(x, Z | \theta) = \Pr(Z | \theta) \Pr(x | Z; \theta) \\ &= \Pr(Z_1, \dots, Z_N) \cdot \Pr(x_1, \dots, x_N | Z_1, \dots, Z_N; \theta). \end{aligned} \tag{3.6}$$

Here  $N = |\mathcal{C}|$  is the number of event variables, and  $x_j$  denotes all the claims made by the sources about the  $j$ -th variable. In Equation (3.6),  $\Pr(Z | \theta) = \Pr(Z)$  because the joint probability of the event variables  $\Pr(Z)$  is independent from the parameters  $\theta$ .

Next, we are going to simplify the likelihood function by proving that for any event variables  $j_1$  and  $j_2$ ,  $x_{j_1}$  and  $x_{j_2}$  are conditionally independent given the latent variables  $Z$ . We use  $X \perp Y$  to denote that  $X$  and  $Y$  are independent, and similarly  $X \perp Y | Z$  to denote that  $X$  and  $Y$  are conditionally independent given  $Z$ . Before proving  $x_{j_1} \perp x_{j_2} | Z$ , we first introduce the definition of  $d$ -separation.

**Definition 1** ( $d$ -separation). *Let  $G$  be a Bayesian network, and  $X_1 \rightleftharpoons \dots \rightleftharpoons X_n$  be a trail in  $G$ . Let  $Z$  be a subset of the observed variables. The trail  $X_1 \rightleftharpoons \dots \rightleftharpoons X_n$ <sup>1</sup> is active given  $Z$  if*

- *Whenever we have a V-structure  $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$  in the trail, then  $X_i$  or one of its descendants are in  $Z$ , and*
- *no other node along the trail is in  $Z$ .*

*If for any trail between  $X_1$  and  $X_n$  is not active, then  $X_1$  and  $X_n$  are  $d$ -separated in  $G$  by  $Z$  [71].*

Here a trail between  $X_1$  and  $X_n$  is an undirected path that is computed by simply ignoring the directions of the directed edges in the Bayesian network  $G$ . Note that if  $X_1$  or  $X_n$  is in  $Z$ ,

---

<sup>1</sup>We use  $X \rightarrow Y$  to denote the directed edge (arc) that points from  $X$  to  $Y$  in  $G$ , and  $X \rightleftharpoons Y$  to denote the arc that connects  $X$  and  $Y$  whose direction, however, is not of interest.

the trail is not active. Next, we introduce a classical lemma showing that the  $d$ -separation implies conditional independence.

**Lemma 1.** *If  $X_i$  and  $X_j$  are  $d$ -separated in the Bayesian network  $G$  given  $Z$ , then  $X_i \perp\!\!\!\perp X_j|Z$  [71]*

Now we are ready to prove that  $x_{j_1}$  and  $x_{j_2}$  are conditionally independent given the latent variables  $Z$  for any  $j_1, j_2 \in \mathcal{C}$  in Theorem 1.

**Theorem 1.** *For any pair of event variables  $j_1$  and  $j_2$ ,  $x_{j_1}$  and  $x_{j_2}$  are conditionally independent given the latent variables  $Z$ , i.e.,  $\forall j_1, j_2 \in \mathcal{C}, x_{j_1} \perp\!\!\!\perp x_{j_2}|Z$ .*

*Proof.* To prove the theorem, we first need to define the causal relationship between a claim  $SC_{i,j}$  and the value  $C_j$  of event  $j$ . Obviously, the value  $C_j$  of the event is independent of how a source claims it, but the claim  $SC_{i,j}$  made by a source does rely on the value  $C_j$  of event  $j$ . Therefore, it is clear this causal relationship between  $SC_{i,j}$  and  $C_j$  should be modeled by an arc from  $Z_j$  to  $x_{i,j}$  in the Bayesian network, as illustrated in Figure 3.2. Here we do not distinguish the variable  $Z_j$  and its corresponding vertex in  $G$ , and the same for  $x_{i,j}$  and its corresponding vertex.

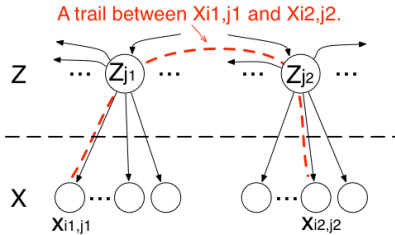


Figure 3.2: An illustration of the Bayesian network.

Therefore, for any pair of  $x_{i_1, j_1}$  and  $x_{i_2, j_2}$ , and for whatever event dependency graph  $G$  of the event variables, we can find two vertices  $Z_{j_1}$  and  $Z_{j_2}$  in  $G$  such that all the trails have the same structure:  $x_{i_1, j_1} \leftarrow Z_{j_1} \rightleftharpoons \cdots \rightleftharpoons Z_{j_2} \rightarrow x_{i_2, j_2}$ , as shown in Figure 3.2. Since  $Z_{j_1}$  and  $Z_{j_2}$  are in  $Z = \{Z_1, \dots, Z_N\}$ , and by Definition 1, we know that  $x_{i_1, j_1}$  and  $x_{i_2, j_2}$  are  $d$ -separated by  $Z$ . Please note that this  $d$ -separation is valid for any pair of sources  $i_1$  and  $i_2$ , thus  $x_i \perp\!\!\!\perp x_j|Z$  by Lemma 1.  $\square$

By Theorem 1 and the independent source assumption, the likelihood function in (3.6) can be

simplified as:

$$L(\theta; x, Z) = \Pr(Z_1, \dots, Z_N) \prod_{j \in \mathcal{C}} \prod_{i \in \mathcal{S}} \Pr(x_{i,j} | Z_j; \theta). \quad (3.7)$$

### 3.3.2 The EM Algorithm

Given the likelihood function, following (3.5), we can derive the EM algorithm. We omit the detailed mathematical derivations here since it is a standard procedure, and directly show the final results of how to update the parameters  $\theta = \{\ell T_i^v, \ell F_i^v | \forall i \in \mathcal{S}, v \in \Lambda, \ell \in \mathcal{L}\}$  in (3.8).

$$\begin{aligned} \ell T_i^v(n+1) &= \frac{\sum_{j \in \ell \mathcal{C}_i^v} \Pr(Z_j = v | x; \theta^{(n)})}{\sum_{j \in \ell \mathcal{C}} \Pr(Z_j = v | x; \theta^{(n)})}, \\ \ell F_i^v(n+1) &= \frac{\sum_{j \in \ell \mathcal{C}_i^{\bar{v}}} \Pr(Z_j = v | x; \theta^{(n)})}{\sum_{j \in \ell \mathcal{C}} \Pr(Z_j = v | x; \theta^{(n)})}. \end{aligned} \quad (3.8)$$

In (3.8),  $\theta^{(n)}$  denotes the parameters in the  $n$ -th iteration of the EM algorithm,  $\ell \mathcal{C}$  denotes the set of event variables with label  $\ell$ , and  $\ell \mathcal{C}_i^v$  is a subset of  $\ell \mathcal{C}$  with each element that the source  $i$  claims its value being  $v$  (i.e.  $\ell \mathcal{C}_i^v = \{j | SC_{i,j} = v, L(j) = \ell\}$ , where  $L(j)$  denotes the label of event variable  $j$ ).

In Equation (3.8), the key step for refining the parameters is to compute  $\Pr(Z_j = v | x; \theta^{(n)})$  for each  $j \in \mathcal{C}$  and  $v \in \Lambda$ . Once we have computed this value, the rest of the computation for updating the parameters becomes trivial. Since we are using a Bayesian network to encode the dependences between variables, we know the conditional probability for each variable given a particular set of values of its parent variables. These are given as an input of our algorithm. Hence, we can compute  $\Pr(Z_j = v | x; \theta^{(n)})$ , the marginal probability of variable  $Z_j$  given the evidence  $x$ . Similarly,  $\Pr(Z_j = v | x; \theta^{(n)})$  can be computed. The pseudocode of our estimation algorithm is given in Algorithm 3.

## 3.4 Evaluation

In this section, we study the performance of our EM-CAT algorithm through extensive simulations. While empirical data is always better, such data often constitutes isolated points in a large space of possible conditions. The simulation, in contrast, can extensively test the performance of our algorithm under very different conditions that are impractical to cover exhaustively in an empirical

---

**Algorithm 3** EM-CAT: Expectation-Maximization Algorithm with Category-specific Source Reliability

---

**Input:** The source-claim matrix  $SC$ , the Bayesian network  $G$ , and event category  $\ell \in \mathcal{L}$  for each event  $j \in \mathcal{C}$ .

**Output:** The estimated variable values, and the reliability vector of each source.

```
1: Initialize  $\theta^{(0)}$  with random values between 0 and 0.5.
2:  $n \leftarrow 0$ 
3: repeat
4:    $n \leftarrow n + 1$ 
5:   for Each  $j \in \mathcal{C}$ , each  $v \in \Lambda$  do
6:     Compute  $\Pr(Z_j = v|x; \theta^{(n)})$  from the Bayesian network  $G$ .
7:   end for
8:   for Each  $i \in \mathcal{S}$ , each  $v \in \Lambda$  and each label  $\ell \in \mathcal{L}$  do
9:     Compute  ${}^\ell T_i^{v^{(n)}}$  and  ${}^\ell F_i^{v^{(n)}}$  from (3.8)
10:  end for
11: until  $\theta^{(n)}$  and  $\theta^{(n-1)}$  converge
12: for Each  $j \in \mathcal{C}$  and  $v \in \Lambda$  do
13:    $Z(j, v) \leftarrow \Pr(Z_j = v|x; \theta^{(n)})$ .
14:   if  $Z(j, T) > Z(j, F)$  then
15:     Assign variable  $j$  with value  $T$ 
16:   else
17:     Assign variable  $j$  with value  $F$ 
18:   end if
19: end for
20: for Each  $i \in \mathcal{S}$ , each category  $\ell \in \mathcal{L}$  do
21:   Compute its reliability  ${}^\ell t_i$  from (3.4)
22: end for
```

---

manner. Evaluation results show that the new algorithm offers better estimation accuracy compared to other state-of-the-art solutions.

## Methodology

We simulated 100 interdependent binary variables. The underlying dependency graph is a random DAG that changes in each simulation run. The Bayesian network is created with the dependency structure defined by the DAG and parameters randomly generated using the toolbox. The expected ground truth for all variables is set to 0.5 (i.e., with probability 0.5, the variable will be True). The actual (marginal) probability distribution for each variable is defined by the Bayesian network.

The ground truth values of variables are generated based on the Bayesian network in a topologically-sorted order. That is, we wait to generate the value of variable  $v$  until the values of all of its parents,  $\text{par}(v)$ , have been generated. Therefore, the ground truth value distribution of our variables follows the Bayesian network. Each event variable is also assigned a label  $\ell$  randomly

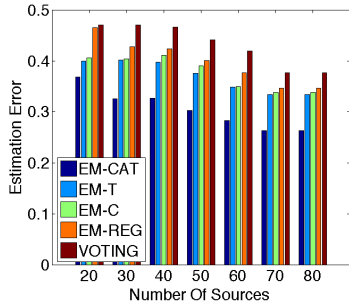
from a label set  $\mathcal{L}$  to simulate the event category.

The simulator randomly assigns a reliability vector for each source. We randomly select a set of the sources to be “experts” at some category. Hence, for each, we choose a category,  $\ell$ , and give the source a high reliability value in reporting variables of that category. The other values in the reliability vector are assigned lower values, making the average reliability of each source roughly the same. We use  $t_i$  to denote the average reliability of source  $S_i$ . In the simulation,  $t_i$  is in the range  $(0.5, 1)$ . We also simulate the “talkativeness” of the sources, which denotes the probability that a source would make a claim, denoted by  $s_i$ .

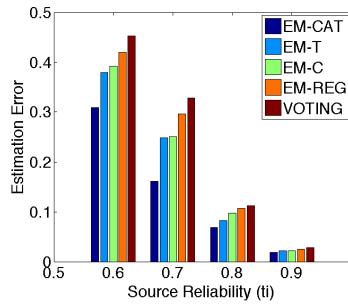
The source-claim matrix  $SC$  is then generated according to the reliability vector of each source,  $t_i$ , and the talkativeness of each source,  $s_i$ . For each source  $S_i$  and each event variable  $C_j$ , we first decide whether the source will make a claim about the variable by flipping a biased coin with probability  $s_i$  that the source will claim something. If it does not claim, then  $SC_{i,j} = \textit{Unknown}$ , otherwise we generate the value of  $SC_{i,j}$  based on  $T_i^v$  and  $F_i^v$  which can be computed from the reliability vector of the source.

The source-claim matrix  $SC$  is the evidence  $x$  in the Bayesian network. To include the claim nodes in the Bayesian network, we extend the DAG such that for each vertex  $V_j$  in the DAG  $G$  we create one vertex  $V_j'$  and add an arc  $(V_j, V_j')$  to  $G$ . We tried to add one vertex  $V_{i,j}$  for each  $x_{i,j}$  and directly set the parameters of  $V_{i,j}$  to the corresponding  $T_i^v$  and  $F_i^v$  in the parameter vector  $\theta^{(n)}$ . This implementation is straightforward. However, it adds too many extra (that is  $|\mathcal{S}| \times |\mathcal{C}|$ ) vertices to the Bayesian network, which greatly slows down the inference computation. Therefore, in our implementation, we just add one vertex  $V_j'$  for each variable  $V_j$  in  $G$ , and set the evidence (the observed value) of  $V_j'$  to False (which means  $\Pr(V_j' = \textit{False} | V_j) = p(x_j | Z_j; \theta^{(n)}) = \prod_{i \in \mathcal{S}} p(x_{i,j} | Z_j; \theta^{(n)})$ ). In this implementation, we only double the size of the DAG, which makes the inference computation of the Bayesian network much more efficient.

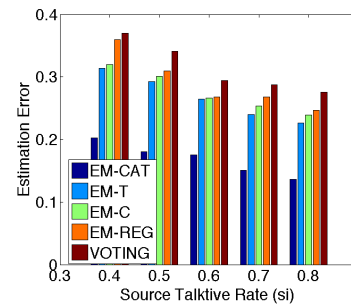
The default values of the simulation parameters are as follows: the number of sources is 40, the expected source (average) reliability  $t_i$  is 0.6, the talkativeness of the source  $s_i$  is 0.6. The number of event variables is set to 100, and we randomly generate the Bayesian network parameters such that, in expectation, the probability that each variable is True, is set to 0.5. The number of edges in the Bayesian network is 100. There are 2 categories by default.



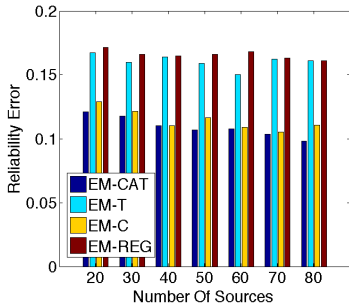
(a) Variable State Estimation



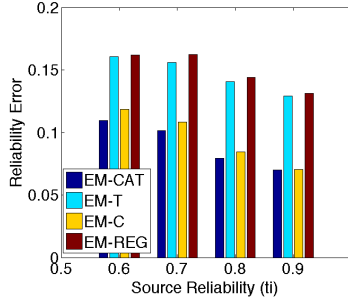
(a) Variable State Estimation



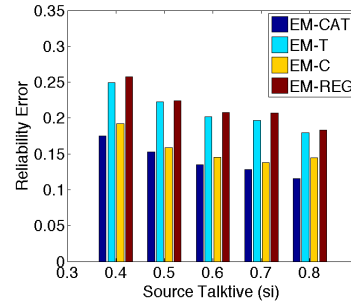
(a) Variable State Estimation



(b) Source Reliability Estimation



(b) Source Reliability Estimation



(b) Source Reliability Estimation

Figure 3.3: Performance as the number of sources varies.

Figure 3.4: Performance as the source reliability varies.

Figure 3.5: Performance as the source talkativeness varies.

We compare our algorithm to the algorithm proposed in [131] and two intermediate extensions towards the current solution. We also include a simple baseline. We use **EM-CAT** to denote our algorithm, and **EM-REG** to denote the algorithm proposed in [131]. Note that, EM-REG assumes that variables are independent, and all the variables share the same category (i.e., it assumes that there is only one category). The first extension of EM-REG is to add the Bayesian dependency structure to the event variables. We call this extension **EM-T** (EM algorithm with structured variables). The second extension of EM-REG is to consider event categories, that is called **EM-C**. The simple baseline algorithm is just voting, and is denoted by **VOTING**. VOTING estimates each variable to be equal to the majority vote. Each simulation runs 100 times and each result is averaged over the 100 executions.

## Evaluation Results

Figure 3.3 shows the performance of our EM-CAT algorithm as the number of sources varies from 20 to 80. In Figure 3.3(a), we observe that our EM-CAT algorithm has the lowest estimation error,



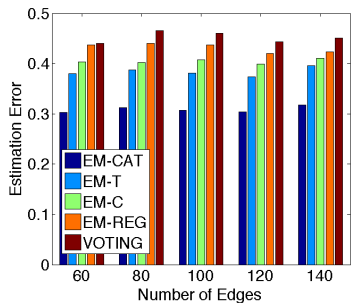
and EM-T and EM-C work better than the regular EM algorithm which is better than simple baseline voting. The reason is that when the underlying event variables follow some dependency structure, exploiting this piece of information will result a better estimator. EM-CAT also considers the category-specific reliability of each source. For each event category, EM-CAT will always select the sources with higher reliability for the category. Therefore, it achieves higher accuracy. Please note that as the number of sources increases, the accuracy of all the estimators increases. More data sources will result in more data. Therefore, the accuracy of the learning algorithm will be improved.

Figure 3.3(b) shows the error in estimating source reliability. Both the EM-CAT and EM-C are better in estimating source reliability than the other two algorithms. The reason is that the other algorithms ignore event categories. Thus, the information regarding differences in source reliability across different categories of observations is not exploited.

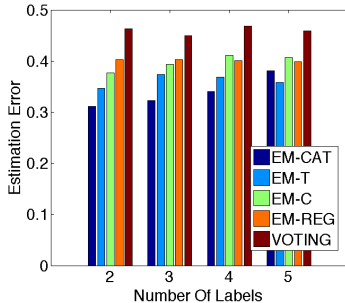
Figure 3.4 shows the performance of the estimators as a function of source reliability. From the figure, we observe that with more reliable sources, the accuracy of the estimators is greatly improved. Even the voting can result in very reliable estimates when source reliability is 0.9 or above (i.e., 90% of their reports are true). Among all the estimators, our new EM-CAT is the best at both estimating the ground truth values of reported variables and the reliability of sources.

Figure 3.5 explores the effect of “talkativeness,”  $s_i$ , of the sources on estimation accuracy. As mentioned earlier, the talkativeness of a source denotes the probability that the source will make a claim regarding some variable. In the experiment, talkativeness is varied from 0.4 to 0.8. With higher talkativeness, we have more data. This is the reason why accuracy of the estimators improves as  $s_i$  increases. Again, our EM-CAT algorithm is the best among all the estimators.

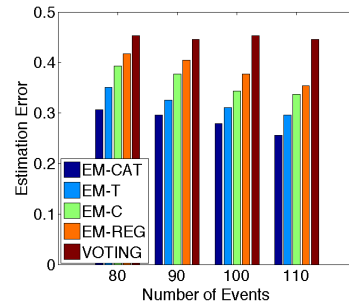
In Figure 3.6, we study the performance of the estimators when the number of edges in the dependency structure (a DAG) varies. A larger number of edges in the DAG means more dependencies among variables. Figure 3.6 shows that performance of the state estimators does not change much with the number of dependencies. The reason could be that the parameters of the Bayesian network are generated *uniformly* at random in  $(0, 1)$ . Therefore, in expectation, the bias of the ground truth is around 0.5. However, if the bias of the ground truth is skewed, we will observe a difference among different dependency structures, since the value of a variable will be affected more



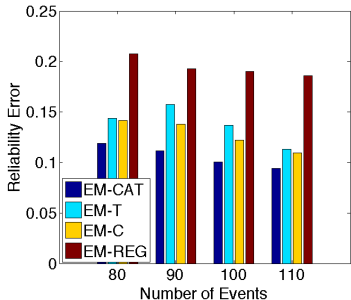
(a) Variable State Estimation



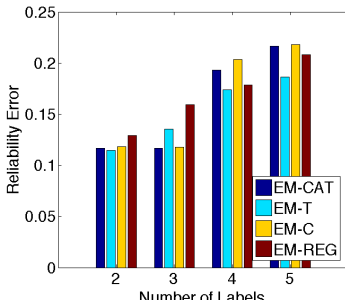
(a) Variable State Estimation



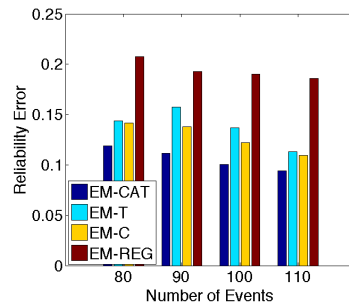
(a) Variable State Estimation



(b) Source Reliability Estimation



(b) Source Reliability Estimation



(b) Source Reliability Estimation

Figure 3.6: Performance as the number of edges in the Bayesian network varies.

Figure 3.7: Performance as the number of event categories varies.

Figure 3.8: Performance as the number event variables varies.

by its depended variables. Our algorithm is the best among all the algorithms since we exploit the dependency structure of the variables. Although the accuracy of estimators does not vary much as the structure changes, exploiting this information leads to a more accurate state estimator.

We study the performance of the estimators as the number of category labels varies from 2 to 5 in Figure 3.7. From Figure 3.7(a), we observe that as the number of labels increases, the performance of EM-CAT becomes worse. This is because we end up with fewer and fewer data in each category. With fewer data, the parameters of the estimator cannot be learned accurately. Therefore the performance of estimation degrades. This figure suggests that when the data size is small, it is better to ignore category, but with a large data size, it would be better to exploit it.

Figure 3.8, we study the performance of the estimators as the number of variables varies from 80 to 110. From the figure, we observe that the accuracy of the estimators improves as the number of variables increases. The reason is that we have more data to learn the estimation parameters more accurately. Actually, the voting algorithm does not vary much, since in voting each source has the same weight as the others. Therefore, even when the number of variables increases, the weight

of the sources does not change, leaving performance the same. Again, our algorithm EM-CAT is the best among all the algorithms in both the estimation of ground truth values of variables and estimation of source reliability.

Next, we study the scalability of our EM-CAT algorithm. We mainly compare our algorithm with the algorithm proposed in [126]. Their algorithm considers the full joint distribution of all the correlated variables.

We compare three inference algorithms: 1. the **junction tree algorithm (JTree)**, 2. the **variable elimination algorithm (VarElim)**, and 3. the method used in [126], i.e., inference with the **full joint distribution (Total)**. Please note that all the three algorithms compute the exact inference probability of the Bayesian network, there are also algorithms that compute the approximate inference probability [71] which compromises the inference accuracy but has a better computational complexity.

In Figure 3.9, we fixed the expected node degrees to be 2, and varied the number of nodes in the Bayesian network from 10 to 25. Note that, the  $y$ -axis is in log-scale. From Figure 3.9, we clearly observe that the computation time of the Total algorithm increases exponentially as the number of nodes increase linearly. The computation time of both the JTree algorithm and the VarElim algorithm grows in a much less rapid way. The time complexity of both the algorithms actually depends on the size of the largest clique (the complete sub-graph in the Bayesian network); since the expected node degree is 2, it is possible that the resulting graph has a clique of size  $n$ , where  $n$  is less than the total number of nodes  $N$  in the Bayesian network. As the total number of nodes  $N$  increases, the gap between  $n$  and  $N$  will also increase as shown in the Figure 3.9. The JTree algorithm is more efficient than the VarElim algorithm, since it maintains a data structure which can simultaneously update the potential of local cliques but the the VarElim algorithm eliminates the variables sequentially. Furthermore, the time complexity of the VarElim algorithm also depends on the order of the variables to eliminate. It is NP-hard to find the optimal order based on which the VarElim algorithm eliminates the variables. When the total number of nodes  $N$  is 25, we can observe that the Total algorithm needs 20 seconds in average, while VarElim needs 2 seconds and JTree needs only 0.2 seconds in average. We can use JTree algorithm in our EM-VTC algorithm for the Bayesian inference computation, compared with the previous solution that does not exploit

the dependence structure of the variables [126] (i.e. using the Total algorithm), it is not hard to observe how scalable our algorithm is as the number of total variables varies.

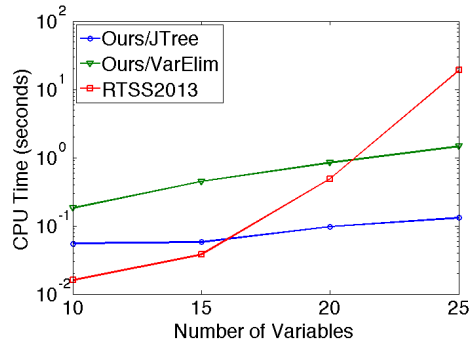


Figure 3.9: Computation time comparison with fixed node degree.

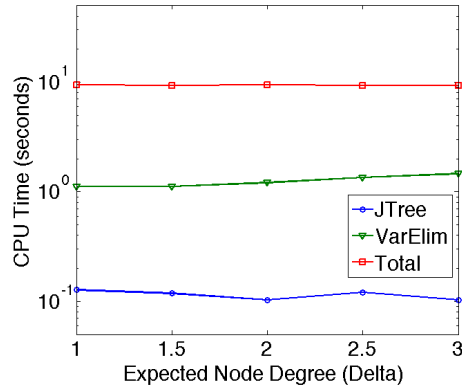


Figure 3.10: Computation time comparison with fixed number of nodes.

Figure 3.10 shows the CPU time of the three inference algorithm when the number of nodes is fixed at 24 but the expected degree of each node varies from 1 to 3. From the figure, we can observe that the CPU time of the VarElim algorithm increases as the number of node degree increases. This is because that with a larger node degree, the chance that the VarElim algorithm selects a bad variable eliminating order become larger. Thus, the time complexity of VarElim increases. However, the time complexity of the JTree algorithm only depends on the size of each local clique, when the node degree is small such as less than 3 the complexity of JTree actually changes very little which does not show in Figure 3.10. Figure 3.10 shows the scalability of our algorithm compared with previous solution [126] as the expected node degree varies in the Bayesian network.

### 3.5 Related Work

Inferring the structure of Bayesian Network is, in general, an NP-complete problem [25]. In order to learn a Bayesian Network in a tractable way, various algorithms are proposed. There are mainly two categories of approaches, score-based and constraint-based [116]. The former one tries to search for the optimum structure based on goodness-of-fit. The latter one utilizes conditional independence to build the network. Depending on the different data types and relationships, various hypothesis tests are available. For continuous data, if the relationship among variables is believed to be linear, tests based on Pearson's correlation are widely used. Asymptotic  $\chi^2$  tests can also be used to test independence between two continuous variables [66]. In cases of categorical variables, one of the most classical tests is Pearson's  $\chi^2$  test [10]. It works on the contingency table and tests if paired observations from two categorical variables are independent. In addition, likelihood-ratio statistic (or  $G^2$ ) can also be used on either categorical or continuous variables; Jonckheere's trend test provides an independence test on ordinal variables [68]. Although in some cases separate test statistics can be used by different tests, they usually provide the same conclusions. If two variables are conditionally dependent, an edge between these two variables should be drawn in the Bayesian Network.

The problem studied in this chapter bears some resemblance to the fact-finding problem that has been studied extensively in recent years. The goal of fact-finding, generally speaking, is to ascertain correctness of data from *sources of unknown reliability*. As one of the earliest efforts in this domain, Hubs and Authorities [70] presented a basic fact-finder, where the belief in a claim and the truthfulness of a source are computed in a simple iterative fashion. Later on, Yin et al. introduced TruthFinder as an unsupervised fact-finder for trust analysis on a providers-facts network [152]. Pasternack et al. extended the fact-finder framework by incorporating prior knowledge into the analysis and proposed several extended algorithms: Average.Log, Investment, and Pooled Investment [97]. Su et al. proposed semi-supervised learning frameworks to improve the quality of aggregated decisions in distributed sensing systems [111, 112]. Towards a joint estimation on source reliability and claim correctness, Wang et al. [127, 131] and Li et al. [79, 114] proposed expectation maximization and coordinate descent methods to deal with deterministic and probabilistic claims, respectively. Though yielding good performance in many cases, none of these

approaches considers situations where the variables in question have wide-spread dependencies. To address this problem, Wang et al. [126, 141] further extended their framework to handle limited dependencies. However, their algorithm has exponential computational complexity in the number of correlated variables, and thus can only be applied in scenarios where the number of dependencies is small. In contrast to their work, we consider a model for more general scenarios where a considerable number of dependencies exists among the variables reported by the unreliable sources.

### **3.6 Conclusion**

In this chapter, we addressed the reliable crowd-sensing problem with interdependent variables. Crowd-sensing is a novel sensing paradigm in which human sources are treated as sensors. The challenge is that the reliability of sources is unknown in advance. Recently, several efforts tried to address this reliability challenge by formulating the problem given different source and event models. However, they did not address the problem when the reported variables are interdependent at large scale. In this chapter, dependencies between reported variables were formulated as a Bayesian network. We demonstrated that our formulation is more general than previous work; previous models being special cases of ours. Evaluation results showed that our EM-CAT algorithm outperforms the state-of-the-art solutions.

## Part II

# Event-level Data Summarization . . . . .

## Chapter 4

# Event Detection and Demultiplexing in Social Spaces

In this chapter, we propose an event detection framework named *StoryLine* that summarizes social sensing data in event-level. *StoryLine* is a novel social sensing (back-end) service that exploits real-time content posted on social media to detect, demultiplex, and track instances of physical events of interest to the user. The user may specify the category of events of interest, such as car accidents, road closures, concerts, or urban protests. The current version of the tool uses Twitter. It is intended to complement services that collect data from physical sensors. We leverage the intuition that Twitter posts (and, by implication, possibly similar microblogging media) can be exploited as a novel *sensing modality*, not unlike acoustic sensing, vibration sensing, or magnetic sensing. The analogy is straightforward as illustrated in Figure 4.1. Much the way physical objects induce distinguishable signals in their physical environment that can be detected by observing the physical medium, socially-relevant events (such as car accidents, attacks, natural disasters, parades, or protests) induce distinguishable signals in their social environment that can be detected by observing the social medium. The chapter develops an IoT service that exploits this social modality of sensing, motivated by the proliferation of users who post in real-time to describe their surrounding world. The service offers a client-side interface and a programmers interface to browse and retrieve detected events, receive alerts when certain events occur, and compute historical statistics.

*StoryLine* makes a fundamental contribution to event detection literature on Twitter; namely, to the authors' knowledge, it is the first chapter that distinguishes between concurrent *instances* of a user-specified category of events (which we call event *demultiplexing*) in a manner that (i) does not need location information and (ii) is entirely unsupervised (i.e., does not need prior training or remote supervision techniques). None of the prior work offers event demultiplexing that has both of the above properties.



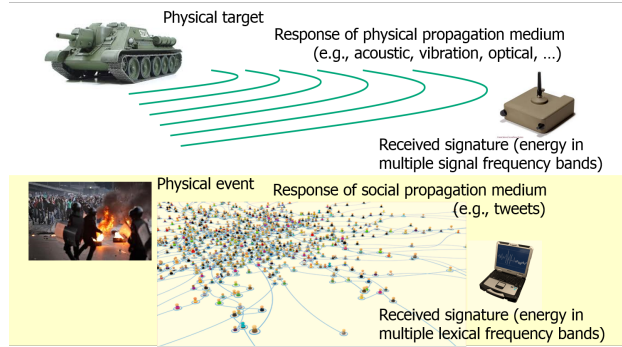


Figure 4.1: The Social sensing modality and its analogy with physical sensing

Demultiplexing is essential to our IoT service, where a city planner, for example, might want to know statistics of events occurrence over time, which implies knowing how many events (say, car accidents) occurred. Not relying on location metadata means we can identify more events, since more than 98% of tweets are not geotagged [115, 142]. Not using language features and related training means the service can be deployed internationally at little or no additional cost, regardless of local language. It will demultiplex events described in most languages<sup>1</sup> (although will not be able to merge descriptions of the same event across different languages).

The idea that social media posts collectively constitute a form of sensing is not new. It dates back to the beginning of the decade. In their pioneering work, Sakaki et al. [102] proposed an algorithm to detect and track natural disasters, such as earthquakes and hurricanes. The work exploited the spatio-temporal footprint of media posts to detect and localize events. Since then, a large volume of literature on event detection was published. Surveys of these techniques recently appeared both for Twitter-based detection specifically [16], and for detection from social media in general [46].

Work on Twitter-based event detection generally falls into three categories. First, some algorithms do detection but not demultiplexing [9, 49, 77, 96]. Demultiplexing is a somewhat different problem from mere *detection* in that one needs to distinguish one concurrent event instance (e.g., a car accident) from another. An algorithm that does not do demultiplexing can detect, for example that a major traffic accident occurred, and can separate traffic accidents from other types of events, such as floods, but cannot easily differentiate between two concurrent traffic accidents.

<sup>1</sup>Since we still rely on white space as word/token separators, it will not work well for languages with no spaces like Chinese.

Many chapter in this category do a form of burst detection and text-similarity-based clustering on tweets. Hence, for instance, tweets containing words related to traffic accidents end up in the same cluster (but can include descriptions of multiple accidents).

A second category of work does demultiplexing (separation of concurrent events of the same type) by clustering tweets based on time and location [19, 80, 123]. They often use some notion of coherence (increased frequency of keywords that are *semantically related*) at a given location as an indicator that an event occurred at that location [158]. Unfortunately, on Twitter, less than 2% of tweets are geotagged [115, 142], so this approach can easily miss small events. While user account registration information commonly includes location (about 25% of accounts have it), it is course-grained (city-scale only), and hence cannot help distinguish different local events.

Finally, previous researches indeed do demultiplexing without location metadata [115, 142]. However, they use natural-language processing or machine learning, and thus are language-specific and/or need prior training. For example, some use shallow analysis of text to identify location keywords (e.g., references to specific streets, cities, or landmarks) [45, 55, 115, 142], and cluster tweets based on locations referred to in the text. In contrast, our approach is unsupervised and hence does not require classifier training [65, 102, 149, 159], bootstrapping [13], or significant pre-processing [21, 121].

This chapter thus opens up a new category of event detection methods that can *demultiplex* events, *without use of location* information, in an entirely *unsupervised* NLP-free fashion. We demonstrate the effectiveness and efficiency of our algorithm in the evaluation section by comparing with state-of-the-art baselines using four real Twitter feeds.

The rest of this chapter is organized as follows. We define our problem more formally in Section 4.1. We propose our solution to unsupervised event detection, demultiplexing, and tracking in Section 4.2. The implementation of the resulting service is described in Section 4.3 and its evaluation is presented in Section 4.4. We discuss the related work in Section 4.5 and conclude the chapter in Section 4.6.

## 4.1 Problem Statement

The purpose of *StoryLine* is to do for Twitter posts what back-end aggregation/fusion services do for crowd-sourced sensor data with the purpose of detecting and tracking physical events in urban spaces. We envision services like *StoryLine* complementing more traditional sensor data fusion services in IoT applications. Towards that end, *StoryLine* represents the monitored environment as a set of event instances, each given by an instance identifier, a general class label, and an observation summary that accumulates chronologically sorted posts (namely, Twitter messages, called *tweets*) regarding the event instance. While *StoryLine* stores the demultiplexed stream of tweets that describes each event instance, this stream – the story – is not interpreted by the service.

New events may be generated over time and old events are eventually removed. Each event has a finite lifespan during which the event is said to be *ongoing*. For the purposes of this chapter, an event instance is broadly defined as an *incident, independently observable by multiple humans within limited time and space*. The term “independently observable” suggests that retweets be ignored, as they do not constitute independent observations. The term “multiple humans” suggests that a threshold could be used on the rate of reported observations, below which an event is of no interest for the purposes of this chapter. Finally, “limited time and space” suggests that an event has a start time, an end time, and a location trajectory. Event locations described by a single point in space constitute a special case of a trajectory. Hence, vehicular traffic accidents, shootings, demonstrations, rallies, funeral processions, insurgent attacks, bombings, and sports events, are different examples that satisfy the definition of events used in this chapter.

In this chapter, we restrict our attention to the problem of demultiplexing of different *instances* of the same (user-specified) event category, together with related instance detection and instance tracking algorithms. To do so, we look for *co-occurrence surprises*; that is, spikes in keywords that *do not* commonly co-occur. An information gain metric is derived to detect such spikes in an unsupervised fashion. For example, in description of car accidents, a particular car accident involving a drunk driver who ran over a dog on a bridge, might be described by tweets containing such keywords as “drunk” and “dog”. These words do not commonly co-occur in the same microblog post. Hence, if such an uncommon combination of words spikes today in the context of tweets about car accidents, it is an indication that a new event instance occurred. We show in the chapter that

co-occurrence surprise leads to better demultiplexing of event instances than techniques based on finding spikes in semantically related or commonly co-occurring words (e.g., “car accident”).

In our problem, *StoryLine* discretizes time into slots, and abstracts the current state of the monitored environment at any discrete time instant,  $k$ , by a dynamically evolving set of ongoing event instances  $\mathcal{E}(k)$ , where an event instance  $E_i$  has a detection (or start) time,  $S_i$ , and a finish time,  $F_i$ . We say that  $E_i \in \mathcal{E}(k)$  for  $S_i \leq k \leq F_i$ . Each event instance is further associated with a chronologically sorted list of all timestamped tweets that describe it up to the current time, called its cumulative observation summary,  $Summary_i[k]$ .

The social medium is said to emit a signal. The signal emitted in slot  $k$  (i.e., the slot ending at time instant,  $k$ ) is the body of text emitted on the social medium in slot  $k$ . In the case of Twitter, this would be the set of tweets time-stamped in slot  $k$ . Our service uses the Twitter programming API to collect tweets in real time as they are emitted. The signal emitted on the social medium in slot  $k$  is denoted  $Signal(k)$ . Given the stream,  $Signal(k)$ , the problem addressed in this chapter is to determine for each time slot,  $k$ , (i) the set of ongoing event instances,  $\mathcal{E}(k)$ , and (ii) the observation summary,  $Summary_i[k]$ , for each event instance,  $E_i \in \mathcal{E}(k)$ .

## 4.2 The Design of StoryLine

In this section, we present informal intuitions, followed by descriptions of our unsupervised detection, demultiplexing and tracking algorithms. To use *StoryLine*, the user issues a *StoryLine* query such as “traffic” and “accident”.<sup>2</sup> This query is like a subscription to a newsfeed that filters content specific to the query terms. A process is started that repeatedly uses Twitter API to obtain the latest tweets (subject to Twitter rate limits) that contain the specified keywords (i.e., match the filter). The resulting real-time stream of arriving tweets is then demultiplexed to separate descriptions of different events (e.g., different accidents), which is the focus of the discussion below. The process continues indefinitely until terminated by the user. At any given time, multiple such queries may be ongoing, depending on the categories of events that the user is interested in following. In principle, other work in current literature can be used to help the user select appropriate keywords

---

<sup>2</sup>The query terms are presumably expressed in the user’s language and hence are language-specific. The point we made earlier, however, is that none of our processing mechanisms use any language assumptions. Hence, they work regardless of the language in which the user expresses the query.

for each query to better filter the desired event category. A substantial amount of work, for example, exists on *topic modeling* [157] that can be leveraged for help with topic-specific queries. This help is outside the scope of our work. In this chapter, we start at the point where a query has been formulated and a stream of tweets matching the query filter has started arriving, and needs to be demultiplexed.

### 4.2.1 Design Intuitions

Perhaps the most important contribution of our demultiplexing approach is its *simplicity*. It is indeed based on a very simple intuition. The intuition underlying the approach lies in a *sparsity argument*; specifically, we find the simplest sparse feature space in which (by virtue of sparsity) event instances are sufficiently separated. To illustrate what this means, consider the lexicon of commonly used words in a language, such as English. Such a domain may contain around 10,000 words. We may want to distinguish 1000s of concurrent event instances, each described by multiple characteristic words. In this case, the set of event instances populate the space of words rather densely. (That is to say, there may be partial overlap between sets of words commonly used in describing *different* event instances.) The same is not true, however, of word pairs (i.e., the “second power” or Cartesian product of the lexical domain). In a language of 10,000 words, there are 100 million possible word pairs. This is several orders of magnitude larger than the number of event instances we might need to demultiplex within any given time slot. Hence, within a given time slot, the set of word pairs that characterize ongoing event instances populate *very sparsely* the feature space of all possible word pairs. The probability of overlap (i.e., different event instances being characterized by the same word pair) is negligible.<sup>3</sup> Two caveats must be understood regarding our sparsity observation.

First, the validity of the sparsity observation in the feature space of keyword pairs hinges on the lack of strong correlations between keywords used in the chosen pairs. The probability of seeing two words,  $W_1$  and  $W_2$ , on the medium is  $P(W_1, W_2) = P(W_1)P(W_2|W_1)$ . If these words often come together as a single term, such as “Dodgers Stadium” or “Angela Merkel”, the probability

---

<sup>3</sup>In a prior literature [43], an empirical study was conducted analyzing tweets about car accidents in three major California cities. The study indeed showed that 2-keyword signatures tend to uniquely distinguish different car accidents. The above general argument presents a signal-sparsity justification of this phenomenon.

$P(W_2|W_1)$  may be close to 1 and thus,  $P(W_1, W_2) \approx P(W_1)$ . In other words, the term should be considered as a single keyword. Hence, we remove from consideration keyword pairs, where the individual keywords co-occur with a much higher probability than the product of the probabilities of their occurrence individually. With that simple filtering, we ensure lack of strong correlations between keywords in a pair.

Second, sparsity ensures that if two event instances are different, their discriminative keyword pairs are different with high probability. The inverse is not always true. Given two different discriminative keyword pairs, they may or may not be of two different event instances. This will be the case, for example, if the event instance has more than two high frequency keywords, allowing for multiple alternative subsets of two keywords to uniquely characterize the event. Such subsets would have to be consolidated.

As tweets arrive, new spikes in keyword pairs are detected and “bins” are associated with spiking pairs, called *discriminative* pairs. Thereafter, subsequent tweets are inspected for discriminative keyword pairs they contain and placed into the corresponding bins. The words in the pair may appear in any order within the tweet and need not be contiguous. A tweet may be placed in multiple bins if it contains multiple discriminative keyword pairs. Note that, identifying discriminative keyword pairs is *not* a quadratic problem in the number of words or tweets in a time slot. This is because the only candidate pairs are those that occur together somewhere in a tweet. Hence, the problem is quadratic in the number of words in a tweet, but *linear* in the number of tweets in a timeslot. Since tweets are of short bounded size, the former component can be bounded by a manageable constant. Accordingly, computationally efficient solutions (linear in the number of tweets) are possible. Importantly, *no prior training is needed*.

Two questions remain. First, how are discriminative keyword pairs selected? Second, how to consolidate bins pertaining to the same event instance? (The latter is needed because an event instance may give rise to multiple discriminative keyword pairs.) These questions are addressed below.

### 4.2.2 Discriminative Keyword Pair Selection

*Information gain* is a common measure for detecting discriminative features that we leverage here. When a new event occurs, keyword pairs characteristic to that event will be present disproportionately in the current window compared to the previous one. We thus compute information gain of a keyword pair in a window as the amount of information gained in distinguishing this window from previous windows if we were told whether or not the given keyword pair occurred in that window. Clearly, pairs that occur more disproportionately in the current window offer more information gain. These are pairs of words that *do not normally co-occur*. Hence, information gain is a measure of co-occurrence surprise.

Let  $X_j$  denote the event whether a tweet contains the keyword pair  $s_j$ , where  $X_j = 1$  means it contains  $s_j$  and  $X_j = 0$  denotes it does not. For simplicity, we omit the script  $j$  when it is clear from the context. Let  $Y_k$  denote the event whether a tweet is posted in the current time slot  $k$ , where  $Y_k = 1$  means it is posted in the current time slot, and  $Y_k = 0$  means it is posted in the previous time slot  $k - 1$ . Again, we omit the script  $k$  for simplicity. The tuple  $(X, Y)$  thus denotes whether a tweet contains the keyword pair  $s_j$ , and whether it is posted in the current time slot  $k$ . It can have four distinct values  $(0, 0), (0, 1), (1, 0), (1, 1)$  that have the straightforward physical meaning respectively.

$H(W)$  denotes the entropy of the variable  $W$  and is defined as:

$$H(W) = - \sum_{w \in \mathcal{W}} p(w) \log p(w),$$

where  $\mathcal{W}$  is the value set of variable  $W$ .

More specifically, let there be  $w_k$  distinct tweets emitted in window  $k$ , and  $w_{k-1}$  distinct tweets emitted in window  $k - 1$ . Hence, the probability of a tweet (taken at random from the tweets in either window) to be present in the current window,  $k$ , is  $p(k) = w_k / (w_k + w_{k-1})$ . Similarly, the probability of a tweet (taken at random from the tweets in either window) to be present in the previous window,  $k - 1$ , is  $p(k - 1) = w_{k-1} / (w_k + w_{k-1})$ .

Let some keyword pair,  $s_j$ , be present in  $w_k^j$  distinct tweets in window  $k$ , and  $w_{k-1}^j$  distinct tweets in window  $k - 1$ . Hence, the probability of a tweet that contains the pair  $s_j$  (taken at

random from those containing that pair in either window) to be from the current window,  $k$ , is  $p_j(k) = w_k^j / (w_k^j + w_{k-1}^j)$ . Similarly, the probability of a tweet that contains  $s_i$  (taken at random from those containing that pair in either window) to be from the previous window,  $k - 1$ , is  $p_j(k - 1) = w_{k-1}^j / (w_k^j + w_{k-1}^j)$ .

Let the entropy of the variable referring to window identity,  $Y$ , be denoted  $H(Y)$ , where  $Y$  is either  $k$  or  $k - 1$ . By definition,  $H(Y)$  is given by:

$$\begin{aligned} H(Y) &= -p(k)\log_2 p(k) - p(k-1)\log_2 p(k-1) \\ &= -\frac{w_k}{(w_k + w_{k-1})}\log_2 \frac{w_k}{(w_k + w_{k-1})} \\ &\quad -\frac{w_{k-1}}{(w_k + w_{k-1})}\log_2 \frac{w_{k-1}}{(w_k + w_{k-1})} \end{aligned} \tag{4.1}$$

Similarly, the conditional entropy of  $Y$ , given that we know whether pair  $s_j$  occurred, is denoted  $H(Y|s_j)$ . By definition,  $H(Y|s_j)$  is given by:

$$\begin{aligned} H(Y|s_i) &= -p_i(k)\log_2 p_i(k) - p_i(k-1)\log_2 p_i(k-1) \\ &= -\frac{w_k^i}{(w_k^i + w_{k-1}^i)}\log_2 \frac{w_k^i}{(w_k^i + w_{k-1}^i)} \\ &\quad -\frac{w_{k-1}^i}{(w_k^i + w_{k-1}^i)}\log_2 \frac{w_{k-1}^i}{(w_k^i + w_{k-1}^i)} \end{aligned} \tag{4.2}$$

Finally, the information gain,  $IG_j$ , associated with pair  $s_j$ , is given by:

$$IG_j = H(Y) - H(Y|s_j) \tag{4.3}$$

Equation (4.3) can be used to compute information gain for each keyword pair,  $s_j$ , in each time slot  $k$ . In computing information gain we do not count retweets, since they do not offer additional first-hand information on events. This helps remove rumors, opinion tweets and slogans that propagate primarily by retweeting, as opposed to descriptions of independently observable events. Only the keyword pairs with information gain greater than a threshold would be selected as discriminative keyword pairs.



The above discussion focused on detection of discriminative keyword pairs; those with high information gain. Remember that high information gain indicates that the words in the pair do not normally co-occur. We show that this insight allows us to find new event instances.

Besides detecting new discriminative pairs in the current window, the system also continues demultiplexing based on discriminative pairs found in previous windows. Those correspond to events detected earlier. Therefore, in each time slot  $k$ , we first inherit all discriminative keyword pairs used in the previous slot whose clusters were still growing, (i.e., the cumulative number of tweet containing that pair by time slot  $k - 1$  is greater than that by slot  $k - 2$ ). We then augment that inherited set with new keyword pairs found discriminative in the current window.

### 4.2.3 The Consolidation Algorithm

Events may contain more than one discriminative keyword pair. Therefore, it is important to be able to consolidate different bins when their tweets are about the same event. Consider the set of discriminative keyword pairs used in slot  $k$ . Each such pair,  $s_j$ , is associated with a bin of tweets,  $C_j$ , in which the pair occurs. Our approach for consolidating bins referring to the same physical event lies in detecting similarity between their respective data clusters. In our drunk driver example, presented earlier, a cluster of tweets about an accident involving a drunk driver killing a dog on a bridge might be distinguished by discriminative keyword pairs (“drunk”, “dog”), (“drunk”, “bridge”) and (“bridge”, “dog”). Each pair might end-up associated with a bin that contains largely the same tweets. A distance metric can thus be defined between content of different bins based on the statistical distribution of words in the bins. The distance between two bins will decide if they are about the same event. Four common distance metrics between statistical distributions of words are compared. Namely, the *Jaccard Distance*, the *Term Frequency Difference Ratio*, the *Cosine Similarity Distance*, and the *KL Divergence*. For the detailed definitions of the distance functions, please refer to the appendix.

We observed that Jaccard distance performs consistently the best among the four, and is also the simplest metric since it is the only one that does not consider word frequency (this empirical comparison is shown in the evaluation section). Interestingly, the metric that depends most heavily on the distribution of words, the KL divergence metric, performed the worst. The reason, we

believe, is that the tweet clusters (the individual bins) are small enough that it is inaccurate to estimate the true probability of each keyword solely based on its frequency of occurrence in a bin. Hence, the more we depend on having to know a true probability distribution, the less accurate is the resulting consolidation.

Another benefit of applying the Jaccard distance is that the resulting consolidation threshold was found to be largely insensitive to the different types of events, due to its simplicity and discreteness. Other lexical distance functions do not have this property. Hence, in our system, we use the Jaccard distance for bin consolidation and pre-configure the threshold as a *static parameter*. New installations of the system need no further “training”. The result of consolidation in slot  $k$  is the set of event instances,  $\mathcal{E}(k)$ , where each event  $E_i \in \mathcal{E}(k)$  is associated with a set of tweets.

#### 4.2.4 Event Tracking

Event tracking extends the consolidation algorithm in a straightforward manner by applying bin consolidation across successive time slots. That is, after consolidating the bins in the current time slot  $k$ , we consolidate the bins between the time slot  $k$  and  $k - 1$ . One challenge in event tracking is that the event signature, defined by the corresponding consolidated keywords, might evolve due to the evolution of the event and thus the way people describe it.

To catch that change, we use an *overlapping sliding window*. It smoothes out the changes in the lexical frequency distribution of fast developing events over time, as illustrated in Figure 4.2. With overlapped windows, some part of the event signature remains the same across the two slots. (Note that, the compared slots are overlapping here as in Figure 4.2.) Therefore, by selecting a proper overlap, we can track the event smoothly and be able to consolidate relevant clusters properly, even as its signature changes gradually over time.

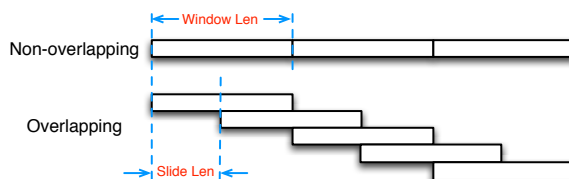


Figure 4.2: Illustration of the non-overlapping sliding window and the overlapping sliding window (with 50% overlapping).

### 4.3 System Implementation

In this section, we present the architecture of our social event tracking system as shown in Figure 4.3. The targeted social medium of our system is Twitter [4]. The system is implemented in Python<sup>27</sup> and integrated into an existing social sensing tool, Apollo<sup>4</sup>, developed by a subset of the authors. StoryLine provides four interfaces, CREATE, PULL, KILL, and STATS. CREATE enables the user to start an event-tracking task, and PULL enables the user to get the real-time event tracking results. The key parameters of CREATE are (i) a list of keywords for crawling tweets, for example [protests, confrontation], and (3) a user-customized window length (with default value of 24 hours). After the user creates a tracking task, a task ID is returned, which is used in PULL to get the real-time tracking results and in KILL to terminate the existing tracking task. Finally, STATS allows retrieval of a set of statistics about the event type, such as the frequency of occurrence of event instances over time.

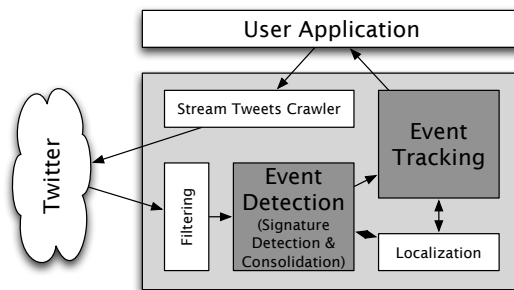


Figure 4.3: Event tracking system architecture

Once the tracking task is created, the crawling parameters are passed to the crawler that uses the Twitter API to crawl tweets that satisfy the conditions defined in the parameters in real time. For the tweets returned, we first filter out the redundant tweets, such as the retweets, and then the filtered tweets are fed to our event detection module, where the event signature detection and consolidation are performed. The text clusters are then passed to the event-tracking module. When the user calls the PULL function with the task ID, the most recent tracking results are returned encoded using the JSON format. An optional localization module is included (to pin the events on a map, for example, by Giridhar et al. [45]), but it is not relevant to this chapter. Please note that unless the user calls KILL, the tracking task keeps working.

<sup>4</sup><http://apollo3.cs.illinois.edu>

## 4.4 Evaluation

In this section, we report the experience of using our tool on event detection and tracking on four datasets crawled from Twitter. We first describe the statistical details of the four datasets, and then discuss the performance of our event signature consolidation for the selected Jaccard distance metric. Next, we study the performance of event detection compared with the state-of-the-art baselines. Finally, we conduct two case studies of Earthquake events and show the real-time event detection capability and event tracking performance of our proposed StoryLine system.

### 4.4.1 Twitter Datasets

For repeatability, we collected four data sets from Twitter using the API described in the previous section. These were then replayed as the feeds used in the subsequent experiments to enable fair comparisons across multiple algorithms and conditions. We summarize data collected by the four tasks we created, labeled by (i) *Disaster*, (ii) *Protest*, (iii) *Traffic*, and (iv) *Armed Conflict* below.

- Disaster The dataset is collected with keywords “disaster”, “humanitarian”, “earthquake”. In this dataset, 1,800,952 tweets were collected after filtered out retweets, and the time span is from Apr. 19th 19:41:08 UTC, 2015 to Feb. 03rd 06:07:15 UTC, 2016.
- Protest The dataset is collected with keywords “protest”, “confrontation”. In this dataset, 1,211,920 tweets were collected after filtered out retweets, and the time span is from Oct. 16th 05:41:02 UTC, 2015 to Feb. 01st 11:15:43 UTC, 2016.
- Chicago Traffic The keywords used here include “traffic”, “accident”, “chicago”. And all tweets in the Chicago area were also collected in this dataset. In this dataset, 8,013,649 tweets were collected after filtered out retweets, and the time span is from May. 15th 13:58:09 UTC, 2015 to Feb. 19th 17:33:43 UTC, 2016.
- Armed Conflict The keywords used here include “rebels”, “attack”, “bombing”. In this dataset, 2,739,363 tweets were collected after filtered out retweets, and the time span is from Oct. 16th 05:52:28 UTC, 2015 to Mar. 07th 02:27:03 UTC, 2016.

In the evaluation, each dataset is fed into our StoryLine system in real-time (i.e., we discretize

the time into slots and in each slot the tool only considers the current data or that of the past slots but never in the future). Here, each time slot (i.e. window) spans 6 hours, and slides 1 hours in each step.

#### 4.4.2 Event Signature Consolidation

We test the performance of event signature consolidation based on each of the four lexical frequency domain distance functions introduced earlier, namely *Jaccard distance (Jaccard)*, *Term Frequency Difference Ratio (Tfreq)*, *Cosine Distance (Cosine)*, and *KL Divergence (KL)*. The consolidation error rate is defined as the ratio between the number of incorrectly grouped 2-keyword signature pairs to the total number of signature pairs. Note that, a 2-keyword signature pair is said to be incorrectly grouped if two signatures corresponding to the same event are put into different groups or if two signatures corresponding to different events are put into the same group. Ground truth labeling is done manually.

Figure 4.4 shows the results, from which we observe that the Jaccard distance function consistently performs the best for all the four datasets, which corroborates our selection of Jaccard distance as the lexical frequency domain distance in Section 4.2.3. The error rate of signature

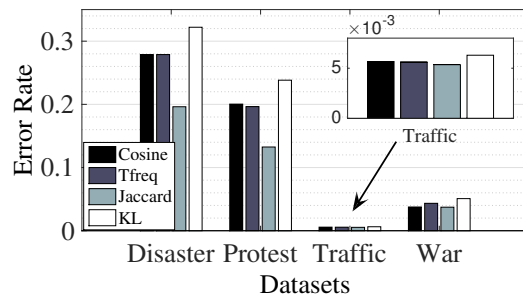


Figure 4.4: The consolidation error rate.

consolidation for the Traffic dataset is the smallest among the four datasets. This is because traffic accidents have a relatively small social media footprint. Often a single 2-keyword signature is associated with the traffic event, therefore only a very small amount of consolidation occurs for this specific event class. We expect that urban events of interest to IoT applications will mostly have small footprints. Examples may be urban fires, shootings, traffic accidents, or road closures. It is therefore encouraging to see that the algorithm is better at detecting and demultiplexing such

small-footprint events. The error rate of consolidation of Jaccard for the war dataset is less than 4%. For the protest dataset and the disaster dataset, the error rates are 14% and 20%, respectively.

### 4.4.3 Event Demultiplexing

In this subsection, we first eliminate geotagging-based demultiplexing techniques based on recall. We then include in the comparison those techniques that do not need location information, illustrating an advantage in precision and purity of demultiplexing (i.e., correct separation of instances).

Table 4.1 shows the percentage of tweets in our data sets that are geo-tagged. We also cluster the tweets into events and show the number of event clusters that carry zero, one, or more geo-tagged tweets. We consider fine-grained events here. For example, a war event might refer to a cluster of tweets discussing a single explosion. The table clearly shows that dependence on location information can render most of the events invisible, as they contain no geotagged tweets.

Table 4.1: Prevalence of geotags in tweets and events

Metric	Traffic	Disaster	Protest	Armed Conflict
Total tweets	8013649	1800952	1211920	2739363
Geotagged tweets	726663 (9%)	6068 (0.33%)	2323 (0.19%)	6381 (0.23%)
Events with no Geotagged tweet	90.7%	99.4%	99.6%	99.6%
Events with 1 Geotagged tweet	3.9%	0.5%	0.4%	0.4%
Events with multiple Geotagged tweets	5.4%	0.05%	0	0

Next, we study the precision of event detection and demultiplexing in our StoryLine system. We compare our StoryLine with the following baselines:

1. *ET* [96]: In this work, an event is detected using common bi-grams, where the bi-grams are selected from among adjacent pairs of tokens, which is an example of techniques that do not demultiplex well. The reason is that in looking for adjacent bi-grams that have a high chance of co-occurrence (for example, “traffic alert” or “crime scene”) one often ends up with bi-grams characteristic of a whole *category* of events. In contrast, in our solution, we look for unusual (i.e., rarely co-occurring) pairs of keywords. Results will confirm that those are more characteristic of an event instance.
2. *TopicModel* [74]: This work proposes an online variation of LDA (Latent Dirichlet Allocation).

tion) [18], a famous topic modelling technique. Events are defined and detected by a topic model. This work is a representative event detection solution based on training a text coherence metric (around a topic).

3. *GeoTag*: In this baseline, we only consider the geo-tagged tweets, and cluster them by physical Euclidean distance. If two tweets are posted within 30 miles, then we cluster them together. A limit is imposed on cluster size to prevent formation of geographically diffuse clusters. This baseline is an example of demultiplexing approaches based on location information.

We randomly selected one week data from our dataset, and compare the precision of event detection/demultiplexing. Here, precision is defined by the ratio between the number of true events output by the algorithm and the total number of events output by the algorithm. Note that, some of the text that the algorithm bins as a separate event might in fact be a false positive. For example, tweets such as “Can you recommend anyone for this #job?” or “these rumors about louis coming to chicago are making me stressed” do not constitute legitimate (geo-)events as defined in this chapter.

Table 4.2: Event detection precision comparison

Algorithm	Traffic	Disaster	Protest	Armed Conflict
StoryLine	<b>72.55%</b>	<b>76.92%</b>	80.95%	<b>88.24%</b>
ET	57.14%	36.36%	<b>86.36%</b>	61.90%
TopicModel	55.10%	60.87%	65.22%	69.57%
GeoTag	66.67%	23.33%	47.37%	41.38%

Table 4.2 summarizes the precision results of all the algorithms. From this table, we can observe that our algorithm has the highest average performance rank of 1.25 (i.e. it ranks first in Traffic, Disaster, and Armed Conflict datasets and second in Protest dataset), whereas ET has average performance rank of 2.5, TopicModel has 2.75 and GeoTag has only 3.5. In the Protest dataset, most of the events are related to some protests. The number of tweets increases greatly when the protest starts, and at the same time, the total number of tweets also increases. Therefore, the increase of the percentage of the event related tweets and the total tweets is not that significant,

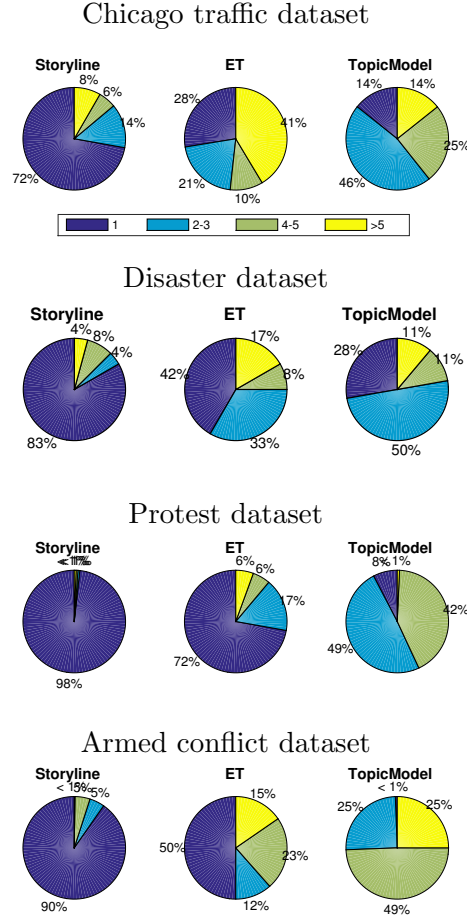


Figure 4.5: The purity pie charts.

thus some true events were not detected by our information gain based approach. But some noisy events were not affected, thus the precision of our algorithm is not the best. ET is based on the absolute increase of the number of event related tweets, therefore, it beats our algorithm. We also notice that geo-tagging does not perform well. We therefore drop it from further comparison.

Figure 4.5 shows the results of purity comparisons for the remaining algorithms, for all the datasets. Purity is a measure of demultiplexing quality into different event instances. Sometimes, the algorithm will output one event that might contain multiple instances. For example, three instances of traffic accidents were output by the TopicModel algorithm: (1) “I 70 now reporting 2 INJURY ACCIDENTS near OH 37”, (2) “When things go BOOM on the US 60 @ArizonaDOT #12News”, and (3) “@WKYTTraffic tracking an ongoing closure along I-75 near the TN stateline.” The purity is defined by a vector, that is the percentage of output events that contain only (1)



one event instance, (2) two to three instances, (3) four to five instances and (4) greater than five instances. Ground truth is labelled manually by two different people and conflicts are resolved by a third one.

From the pie charts, we clearly observe that our algorithm has the highest percentage of output events that only contain one instance, which shows that our algorithm does better at *demultiplexing* event instances compared with the baselines.

#### 4.4.4 Case Study – Real-time Earthquake Detection

In this subsection, we conduct a case study to evaluate the delay in event detection. Here, we select Earthquake events because it is easy to find out the exact (ground-truth) time at which they occurred.

Table 4.3: Real-time earthquake detection summary

Earthquake Location	Happened Time	Detection Time	Delay
Midoro, Philippines	10/19/2015 13:50	10/19/2015 18:26	4:16
Vanuatu	10/20/15 21:52	10/21/2015 02:40	4:48
Afghanistan	10/26/15 09:09	10/26/15 10:17	1:08
Molucca islands	01/11/16 16:38	01/11/16, 20:41	4:03
Afghanistan	01/12/16 20:05	01/12/16 22:59	2:54
Alberta, Canada	01/12/16 17:30	01/12/16 22:59	5:29
Urakawa, Japan	01/14/16 03:30	01/14/16 04:09	0:39
Alaska	01/24/16 10:30	01/24/16 11:37	1:07
Morocco	01/25/16 04:22	01/25/16 10:44	6:22
Taiwan	02/06/16 19:57	02/06/16 21:39	1:42
Fiji	02/06/16 01:39	02/06/16 02:41	1:02
Indonesia	02/12/16 10:02	02/12/16 13:28	3:26
Oklahoma	02/13/16 17:07	02/13/16 22:37	5:30
NewZealand	02/14/16 00:13	02/14/16 04:38	4:25
Wasco, CA	02/24/16 00:02	02/24/16 00:37	0:35
Antarctica	02/23/16 18:08	02/24/16 00:37	6:29
Cebu, Phillipine	03/01/16 14:52	03/01/16 17:14	2:22
Sumatra, Indonesia	03/02/16 12:49	03/02/16 14:19	1:30

Table 4.3 shows a summary of the ground-truth occurrence time and detection time (in UTC) of recent earthquake event instances. From the table, we observe that for most of the instances, our algorithm can detect it within 4 hours. For earthquakes occurring in regions with large numbers of active Twitter users, like Japan and California, we can detect earthquakes within 1 hour. (Note that our window sliding length is just 1 hour, so 1 hour is the smallest delay feasible in this configuration.) The results confirm utility of the system for detection of urban events.

#### 4.4.5 Case Study – Nepal Earthquake Tracking

Finally, we conduct a case study of the Nepal earthquake to help the readers intuitively understand the performance of the tracking functionality of our StoryLine system. The result is summarized in Table 4.4. An earthquake happened on April 24th 2015 that resulted on the death of more than 8,000 people in Nepal. The event was detected due to the rise of tweets with new high-information-gain keyword pairs on the social medium. New keyword pairs were associated with the same event as it evolved. The table shows detected keyword pairs and example tweets from their clusters.

Table 4.4: Nepal earthquake tracking summary

Date	New Detected Keywords	Sample Tweets
04/25/2015	nepal, earthquake, death	Powerful magnitude-7.8 earthquake that rocked Nepal triggered an avalanche on Mount Everest <a href="http://t.co/MULEuWhx3Q">http://t.co/MULEuWhx3Q</a> <a href="http://t.co/QeRKg8QgYp">http://t.co/QeRKg8QgYp</a> RT @BBCBreaking: At least 876 killed in Nepal #earthquake; deaths also reported in India, Tibet & Bangladesh <a href="http://t.co/3BT09l1QZ4">http://t.co/3BT09l1QZ4</a> <a href="http://t.co/2026">http://t.co/2026</a>
04/26/2015	help, nepalearthquake	RT @cnnbrk: At least 2,263 people have died in Nepal from massive #NepalEarthquake and aftershocks, official says. <a href="http://t.co/hCyjO7YyS7">http://t.co/hCyjO7YyS7</a> Anyone with information about my son Joseph Patrick please help #NepalEarthquake #Pray4Joe <a href="http://t.co/X2Kn7mOtRO">http://t.co/X2Kn7mOtRO</a> <a href="http://t.co/2026">http://t.co/2026</a>
04/27/2015	surges, devastation, drone, thankyoupm, donations	Nepal #earthquake: Death toll surges to 3,218; four aftershocks felt in last 12 hours <a href="http://t.co/Njvru9k2kQ">http://t.co/Njvru9k2kQ</a> @cnni: New drone footage shows the extent of devastation from the #NepalEarthquake: <a href="http://t.co/7PiPjayQZ1">http://t.co/7PiPjayQZ1</a> <a href="https://t.co/phIGRkYoZQ">https://t.co/phIGRkYoZQ</a> #ThankYouPM for massive rescue and relief operation by India in Nepal after #earthquake Nepal Earthquake: Facebook to Match Donations Made for Victims <a href="http://t.co/aLooadYNxj">http://t.co/aLooadYNxj</a> Free Submission <a href="http://t.co/J90dT2qnXb">http://t.co/J90dT2qnXb</a>
04/28/2015	salute2indianforces, koirala, sanjay, humanitarian	Thank you very much Indian Forces for being with us.It means alot.... #Salute2IndianForces CNN's Dr. Sanjay Gupta performs surgery on girl in Nepal: CNN's Dr. Sanjay Gupta performed a life-saving... <a href="http://t.co/4EtmH28EwC">http://t.co/4EtmH28EwC</a> #tcot Live: Nepal earthquake kills 4,352, PM Sushil Koirala says death toll could reach 10,000: A high-intensity ear... <a href="http://t.co/A68VtR6hWK">http://t.co/A68VtR6hWK</a>
04/29/2015	survivor, hours, hospital, field, miracle	Nepal earthquake survivor drank urine while trapped for 82 hours <a href="http://t.co/v9DhM5Jhnf">http://t.co/v9DhM5Jhnf</a> #worldnews That is amazing, Nepal Army rescued a 4-month kid alive after 22 hours! <a href="http://t.co/KzJPJeZDCx">http://t.co/KzJPJeZDCx</a> <a href="https://t.co/HvTkvs0Ba0">https://t.co/HvTkvs0Ba0</a> via @sharethis RT @haaretzcom: Nepal earthquake updates / Israeli field hospital opens, to treat 200 people per day <a href="http://t.co/PMwRlRT6YO">http://t.co/PMwRlRT6YO</a> <a href="http://t.co/s9i">http://t.co/s9i</a>
04/30/2015	pakistan, serves, masala, teenage, lydia	Pakistan serves 'beef masala' to earthquake-hit Nepal via /r/worldnews <a href="http://t.co/GoFJO09mJP">http://t.co/GoFJO09mJP</a> Teenage boy pulled out of rubble alive five days after Nepal earthquake <a href="http://t.co/0kiAigYE7M">http://t.co/0kiAigYE7M</a> #telegraph #news Lydia Ko donating earnings to Nepal relief effort: The 18-year-old Ko, ranked No. 1 in the world, successfully... <a href="http://t.co/2nquCITqJa">http://t.co/2nquCITqJa</a>

From the table, we observe that in the beginning of the earthquake, media posts focused more on the earthquake itself using keywords such as “earthquake” and “death” in tweets. As the earthquake developed, people switched their attention to relief efforts, using keywords such as “donations” and

“humanitarian”. Later, the discussion focused on survivors, using keywords such as “survivor” and “hospital”. Neither the original occurrence of the event nor any of the above keyword pairs was known to our algorithms in advance. They were detected automatically and associated with the same event based on discussed distance metrics. The example shows the capability of our algorithm to tracking real-world events as they evolve.

## 4.5 Related Work

The idea of using social networks as sensor networks was discussed in recent literature [127, 131]. While much work focused on analysis of reliability of crowd-sourced observations, this work exploits social media (specifically, Twitter) to build an IoT service for event detection, demultiplexing, and tracking.

Event detection in social spaces is an active research topic in information retrieval. Some early work includes Allan et al. [13], in which they proposed an online event detection and tracking algorithm. Their algorithm exploits features based on term frequency (TF) and inverse document frequency (IDF), such that if the feature score for a new term is above a predefined threshold then a new event or topic is found. Some recent literature exploits TF-IDF-like features includes Shamma et al. [107] and Benharus et al. [17]. Shamma et al. [107] proposed a peakiness score to identify words that are salient in some time window that were used to detect new events. Since unigrams may not always be sufficient to describe complex events, Benharus et al. [17] proposed a different normalized frequency metric called the trending score for identifying event related n-grams instead of unigrams. These approaches are good at identifying event categories and topic. However, as shown in the evaluation, they are less efficient at separating individual event instances. Our work is also related to the text stream clustering literature [8]. An example is work utilizing optimizations of  $k$ -means algorithms to cluster data streams, as proposed by Ordonez [95] and Zhong [162]. However, theirs need prior knowledge (such as the  $k$ ), which is not always available in social streams for event detection and de-multiplexing. Our approach, in contrast, depends on detecting *co-occurrence surprise*; that is to say, new frequently co-occurring words in tweets that did not previously co-occur. Moreover, our calculations are conducted based on only two adjacent time windows, which is much more efficient than the TF-IDF approach that needs to consider the

whole (or a large portion of) corpus.

Topic modeling is another common approach for event detection [53, 74, 163]. Lau et al. [74] proposed an online variation of Latent Dirichlet Allocation (LDA) [18]. In LDA, each topic is modeled as a multinomial distribution of words in a vocabulary, and each document is modeled as a multinomial distribution of  $k$  topics, where  $k$  is a predefined parameter denoting the total amount of topics. And these two classes of multinomial distributions have two Dirichlet priors respectively. (Dirichlet prior is chosen due to the fact that it is the conjugate prior of the multinomial distribution.) The idea in Lau et al. [74] is incrementally updating the priors in each time window based on previous calculated parameters, and maintaining the one-to-one correspondence of the topics in the current time window and the last one. If there is a sudden change in the topic word distribution, then a new event is supposed to have occurred, where the distance of the distributions is measured by the Jensen-Shannon divergence. Hu et al. [53] proposed ET-LDA (joint Event and Tweets LDA) that exploits a search engine and aligns tweets with corresponding text of events provided by traditional media. They showed that results are greatly improved. Zhou et al. [163] further expand LDA with time and location of the tweets, and proposed a new graphical model called location-time constrained topic (LTT). In their approach, the tweet content, timestamps and geo-tags are all considered. As with TF-IDF based approaches, the topic modeling based approaches also suffer when multiple event instances occur in parallel. Furthermore, on Twitter (which is our focus), reliance on geotags is not sufficient to distinguish different event instances due to the relative scarcity of geotagged tweets.

Previous work also exploits the features (metadata) of tweets in event detection. Chierichetti et al. [26] proposed an event detection algorithm that is purely based on communication pattern analysis in the tweet stream. In their solution, events are detected based only on tweet and retweet counts, via logistic regression. They also provide a model of communication to explain the rationale behind event detection. Similarly, Aggarwal and Subbian [9] considered the social network topology and proposed a clustering solution for event detection. The rationale behind such techniques is based on distinguishing shared human interests; namely, clustering text with similar retweet/communication patterns will isolate events with shared community interest. However, in such approaches, events that trigger a similar community response (such as different terror attacks

in nearby locations, or different assaults on police in nearby towns) cannot be easily demultiplexed.

An entirely different line of event detection and demultiplexing techniques focus on location-based (or more generally, spatio-temporal) features [19, 80, 123]. These approaches use different forms of clustering by location metadata contained in tweets, which is indeed an effective means of separation of event instances if the location metadata is sufficiently fine-grained. Unfortunately, less than 2% of tweets are geotagged [115, 142]. While location of other tweets can be estimated from the registered account location of the source, the account metadata carries only city-level location information, which is not sufficiently fine-grained for demultiplexing events at sub-city scale, such as traffic accidents. An interesting approach in the category of location-based event detection techniques is Geoburst [158]. It floats a circle of a pre-specified radius and computes a measure of coherence of tweets originating within the circle. Coherence measures semantic distances between words used in these tweets. When coherence spikes (indicating shorter distances) an event is said to be detected. The rationale is that event occurrence focuses the discussion around fewer topics related to the event, leading to increased coherence of local tweets. Liang et al. [81] exploit a noise filtering approach for event detection and demultiplexing, where the temporal and spatial frequency of each token is treated as a signal and a band-pass filter is applied to filter out background noise as well as separate different event signals within a given locale. In contrast to the above approaches, ours does not depend on using location metadata.

Finally, like us, some recent papers indeed propose demultiplexing schemes that do not use location metadata [45, 55, 115, 142]. Instead, they use language-specific features to distinguish events. A common example of such processing is *isolation of location keywords within the text* of the tweets [45, 115, 142], then clustering by the extracted location information. To appreciate the disadvantage of these techniques, the reader is invited to extract the location information from each of the sentences in Figure 4.6. Our point is: an approach that does not depend on having language-specific extraction rules is much easier to port across languages, which is a big advantage when considering an international medium, such as Twitter.

Our technique, in fact, often finds location keywords automatically as part of the detected signature keyword pairs. Importantly, however, it does so based on statistical analysis alone, and not linguistic analysis of data. Unlike other event detection techniques that rely on clustering [9,

Το γαλλικό πλήρωμα αναγκάστηκε σε προσγείωση  
στην Αθήνα, στην πορεία τους προς τη Μόσχα  
フランスの乗組員は、モスクワへ向かう途中、アテ  
ネの緊急着陸を余儀なくされた  
واضطر الطاقم الفرنسية إلى الهبوط اضطراريا في أثينا في طريقها إلى  
موسكو  
फ्रेंच चालक दल मास्को के लिए अपने रास्ते पर एथेंस में एक  
आपात लैंडिंग करने के लिए मजबूर किया गया

Figure 4.6: Tweets with location information.

49,77,96,107,145], ours looks for frequent pairs that did not usually co-occur. In contrast, much of the prior work looks for burstiness of keywords that are *semantically related* or *frequently co-occur* in some context, as a way of detecting events that feature the indicated semantics or context. This distinction, as we have shown, makes our solution better at event demultiplexing, which is the main contribution of the work.

Finally, target detection and tracking with physical sensors have been extensively studied in other communities such as sensor networks [48, 82, 150]. A particularly relevant sensor model is that of binary sensors [15], since it closely corresponds to twitter posts that either indicate an event or not. We hope that such literature will inform event detection and demultiplexing algorithms on Twitter.

## 4.6 Conclusions

In this chapter, we presented a novel service for IoT applications that augments physical sensor data aggregation and fusion with social media data processing for purposes of physical event detection and demultiplexing. We argued that the social modality of sensing is not unlike other sensing modalities, such as magnetic, acoustic, or seismic. In each case, a useful practice is to transform the signal received from the environment into an appropriate feature domain, and then perform signal processing on that domain. This chapter described an exercise in applying the above approach to Twitter text. A specific contribution was the development of an event demultiplexing algorithm that allows separation of (text pertaining to) different instances of a given user-defined category of urban events (e.g., car accidents) in a manner that (i) is entirely unsupervised and (ii) needs no location information. In turn, this separation allows computing various statistics about the

events in question, such as their frequency over time. Evaluation results show that the approach is successful at detecting, demultiplexing, and tracking physical events. The success of the approach is analytically attributed to a sparsity argument that enables one to use a very simple feature space to demultiplex instances of events.

The chapter is an example of IoT services that go beyond physical sensing. Indeed, in future applications, such as smart cities, data from physical sensors will be fused with data from social media in order to better understand events in the city. Such physical and social fusion offers interesting directions for future work. The chapter is a first step towards the envisioned novel cyber-physical architectures. The authors are in the process of investigating follow-up ideas that jointly exploit combinations of physical sensors and social media.

## Chapter 5

# Event Tracking by Integrating Twitter and Instagram

In this chapter, we investigate the problem of tracking events in physical spaces with the help of data shared on social networks by users observing them. As per the recent statistics [3,5], Twitter has 317 million monthly active users and more than 500 million uploaded tweets per day. Instagram has 600 million monthly active users and more than 80 million uploaded images per day. With the constant increase in users and content, these social networks are becoming a great source of crowdsourced information.

This work contributes to literature on event detection from social media. Specifically, we investigate the degree to which event detection can be improved by fusing data from Twitter and Instagram. The fusion operation itself is separable from the individual event detection techniques used in each network. While, in this workshop publication, we demonstrate a proof of concept by fusing outputs of two specific detection techniques, in principle, we aim at a fusion algorithm that is independent of per-network detection specifics.

One robust observation across several Twitter-based detection techniques, compared by the authors [137], is that it is difficult to filter out false positives. This might be attributed to the vast number of posted tweets, which increases the likelihood of formation of spurious clusters, not representative of actual physical events. Instagram, on the other hand, has sparser content, which leads to a much higher precision (when clustered for event detection), although a lower recall [44]. That is to say, clusters of pictures detected from Instagram posts are generally indicative of real geo-events, although (due to the smaller number of posts) more events are missed. To improve Instagram recall, recent work proposed techniques for corroborating Instagram pictures using Twitter data to distinguish one-off irrelevant pictures from those related to actively discussed events.<sup>1</sup> The approach allows detection of events with smaller support in Instagram, but does not

---

<sup>1</sup>This work is currently under submission and can be accessed via the link <http://hdl.handle.net/2142/95127> [42].



detect events that have *no* Instagram presence.

In contrast, this chapter aims at fusing events from Twitter and Instagram in a manner that chooses the best of both sets. Importantly, it includes events with representation on only one of the two networks (i.e., those on Twitter only or Instagram only), as long as they are sufficiently supported. We show that the approach offers a better trade-off between precision and recall. Note that, events discussed on social media are not a flat structure. Rather, they can be split into sub-events or aggregated into larger events. For example, social media users might discuss specific demonstrations at different locations. They might also refer to the larger context that brings about these protests, and to different individual incidents that occurred during one demonstration. By combining clusters from both Instagram and Twitter one has a higher chance of uncovering such event linkages, hence consolidating valid threads (tracks) of events over time, and eliminating poorly supported events outside such threads.

The rest of this chapter is organized as follows. In Section 5.1 we present the problem formulation and the design of our approach. The evaluation is discussed in Section 5.2. Related work is described in Section 5.3. Finally, conclusions are presented in Section 5.4.

## 5.1 Problem Definition and System Design

In this section, we first define our problem of event tracking integrating Instagram and Twitter data, then we overview our system architecture and design, and introduce each of the system modules in detail.

### 5.1.1 Problem Formulation

We study the problem of integrating Twitter and Instagram to detect physical events and to track them as they evolve. Our goal is to do so online, using a language-agnostic, unsupervised approach.

Following previous work [137], time is discretized into slots called *windows*. Each event instance,  $E_i$ , has a detection (or start) time, and a termination (or finish) time. The event is said to be ongoing between these two times. Each event instance is further associated with a chronologically sorted list of timestamped tweets and Instagram photos that describe it up to the current time.

We restrict our attention to approaches that do not analyze content of tweets or images. This

is because we want our detection and tracking system to be generally applicable regardless of language. Also, due to the heavy cost of human labelling of text and pictures, an unsupervised approach is desired.

We define our problem as follows: given the stream of tweets on Twitter and photo posts on Instagram in window  $k$ , for each  $k$

- determine the set of ongoing event instances at time  $k$ , and
- identify the set of tweets and images related to each event instance.

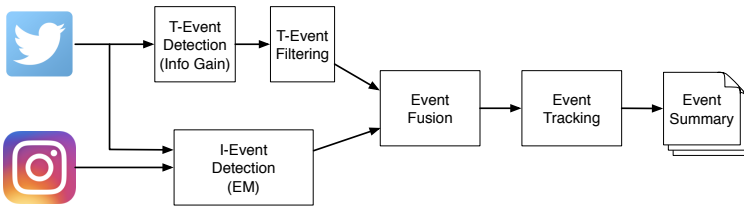


Figure 5.1: Event tracking system architecture

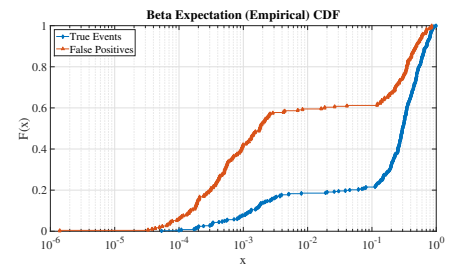


Figure 5.2: CDF of empirical Beta expectation comparison

### 5.1.2 System Architecture and Design

The system architecture is illustrated in Figure 5.1, where the boxes denote system components and arrows denote data flows. The input of the system comprises the data crawled from Twitter and Instagram, and the output comprises tweets and images associated with every event instance. We continuously crawl Twitter text and Instagram photo posts and feed the data to our system, say once per hour. In the system, we have two event detectors, the T-detector and I-detector, respectively. The T-detector detects events from Twitter data using a recently proposed detection algorithm [137]. The I-detector detects events from Instagram. We use an enhanced version [42] that corroborates detected potential Instagram events using tweets with similar tags. The events detected by the T-detector are called *T-events (or T-buckets)*, and those detected by the I-detector are called *I-events (or I-buckets)*. Both the T-events and I-events are fed into the fusion module, where we correlate their buckets. The fused events are then fed into the tracking module, where we track these events over time in an online fashion by consolidating buckets of same events over

time. Finally, we display a summary of current and past events on demand. Next, we describe each of the system modules in detail.

### 5.1.3 Event Detection

The T-detector, in our system, was first introduced in the Storyline paper [137]. The high-level idea is that, in each window, we detect patterns of keywords that occur with disproportionately high frequency compared to the previous window, which translates into high information gain. Evaluation of Storyline [137] demonstrates that this detector does better than other Twitter-based techniques at event demultiplexing. Furthermore, the Storyline detector can be applied online to streaming data, is unsupervised, and does not require analysis of text.

Since we aim to design a fusion technique that is independent of the specifics of individual detectors, we introduce an independent filtering module after the T-detector to remove false positives. The idea behind the filtering module is that real events tend to focus discussion around a narrower scope of topics, leading to a skew in the distribution of keywords in tweets related to the discussed event. A smaller number of keywords get more frequently mentioned, followed by a tail of infrequent keywords. In contrast, when discussion is not focused, the distribution of keywords is much more evenly spread.

To exploit the above intuition, for each token in the tweet cluster returned by the Twitter-based detector (the T-bucket), we calculate the fractions (empirical probability) of tweets in the cluster containing the token. We then fit the empirical probability distribution of tokens to a Beta distribution, and compute its parameters,  $\alpha$  and  $\beta$ . These parameters are then used to detect false positives based on an appropriate threshold.

In order to determine the best threshold, we manually labeled 700 detected events with true event or false positive. The mean values of the Beta parameters for true events were  $\alpha = 0.2104$  and  $\beta = 0.3987$ , and those for false positives were  $\alpha = 0.4025$  and  $\beta = 394.6$ . These values clearly indicate that for true events, there is a subset of tokens that appear in a large portion of tweets in the event cluster, but for false positives, the tokens are distributed much more “evenly” across tweets in each event cluster. To find a single threshold, we further plot the CDF of the *expectation* of the fitted Beta distributions for both the true events and false positives, shown in Figure 5.2.

From the figure, we observe that the expected value of the Beta distribution is a good feature to separate the true events from false positives, as they differ significantly in this feature. Accordingly, we set the threshold to 0.03; if the Beta expectation is greater than this threshold, the bucket is classified as a true event. Otherwise, it is classified as a false positive. Note that, this classifier can work with the output of any tweet clustering scheme.

Our I-detector uses Instagram to detect events, as described in recent work [42]. We use a detector that (1) clusters Instagram photo posts that are co-located in the same proximity and are within a short period of time, and (2) tries to find supporting tweets on Twitter by correlating their hashtags and locations (converting Instagram geotags into street addresses and looking for tweets with corresponding keywords). If we are able to find supporting tweets for a cluster, then we claim successful detection of one (Twitter-corroborated) Instagram event. Please note that the supporting tweets here are not necessarily from some T-bucket(s) that generated by the T-detector. In essence, this approach embeds its own false-positive elimination by seeking corroboration with Twitter. It can be used on top of any other I-detector that returns images related to potential detected events. Note that, the approach returns only those events that have representation in Instagram data. Hence, it returns considerably fewer events compared with the T-detector, although the precision is higher.

Next, we present a fusion algorithm that combines I-events and T-events in a manner that attains both high precision and high recall.

#### 5.1.4 Event Fusion

The fusion algorithm uses the I-buckets and T-buckets as input. Note that, we have both the tweets and photo metadata in the I-buckets. As mentioned above, if a different I-detector is used that returns only buckets of photo metadata, we can always use the approach in [42] to corroborate these buckets with tweets, thereby augmenting them with related tweets. The effect of such corroboration is to reduce false positives (such as “selfies” and other images not corresponding to events of interest).

Next, we create a bipartite graph  $BG = \{I, T, E\}$ , where each node  $i \in I$  is corresponding to an I-bucket, and each node  $t \in T$  is corresponding to a T-bucket. When the similarity in tweets

---

**Algorithm 4** Fusion I-bucket and T-bucket

---

**Input:** I-buckets and T-buckets in window  $k$ , threshold  $\tau_{bucket}$

**Output:** The fused buckets.

- 1: Build the bipartite graph  $BG$  with empty edge set  $E$
  - 2: **for** Each pair of I-bucket  $i$  and T-bucket  $t$  **do**
  - 3:     **if** The similarity score is beyond  $\tau_{bucket}$  **then**
  - 4:          $E \leftarrow E + (i, t)$
  - 5:     **end if**
  - 6: **end for**
  - 7: **for** Each node  $i \in I$  **do**
  - 8:     Merge each node  $t \in T$  s.t.  $(i, t) \in E$ , and denote the merged node as  $m_t$
  - 9:     Update  $E$  such that  $\forall (j, t) \in E, \forall t \in \{m_t\}, E \leftarrow E + (j, m_t)$
  - 10:     Remove every original T-nodes  $t$  in any merged node  $m_t$  and remove all edges incident with it, i.e.  $\forall t \in \{m_t\}, T \leftarrow T - t, E \leftarrow E \setminus \{(\cdot, t)\}$
  - 11: **end for**
  - 12: **for** Each node  $t \in T$  **do**
  - 13:     Merge each node  $i \in I$  s.t.  $(i, t) \in E$ , and denote the merged node as  $m_i$
  - 14:     Remove every original I-nodes  $i$  in any merged node  $m_i$  and remove all edges incident with it, i.e.  $\forall i \in \{m_i\}, I \leftarrow I - i, E \leftarrow E \setminus \{(i, \cdot)\}$
  - 15: **end for**    $\triangleright$  (Now bipartite graph  $BG$  becomes a *match*, that is  $\forall i_0 \in I$  at most one  $t \in T$  s.t.  $(i_0, t) \in E$  and  $\forall t_0 \in T$  at most one  $i \in I$  s.t.  $(i, t_0) \in E$ )
  - 16: Merge the matched pairs, and output the merged buckets and individual I- or T-buckets
- 

between an I-bucket,  $i$ , and a T-bucket,  $t$ , is beyond a similarity threshold  $\tau_{bucket}$ , we add an edge  $(i, t)$  to  $E$ . The similarity of two buckets is defined by the Jaccard distance between the tweet text tokens of them<sup>2</sup>. Here, the threshold  $\tau_{bucket}$  denotes whether we want to merge the I-bucket and T-bucket (if similarity above it) or not (otherwise). Hence, for any edge  $e_{i,t} \in E$ , we merge the corresponding I-bucket  $i$  and T-bucket  $t$ . We summarize the fusion procedure in Algorithm 4. The time complexity of the algorithm is determined by the size of the bipartite graph, therefore, it is  $O(MN + M + N) = O(MN)$  where  $M = |I|$  and  $N = |T|$ .

### 5.1.5 Event Tracking and Summary

The event fusion algorithm is then extended to event tracking in a straightforward manner; instead of correlating I-buckets and T-buckets in the same window  $k$  as done in the event fusion module, in tracking, we correlate buckets detected in window  $k$  and those detected in the previous window  $k - 1$ , merging those with a high Jaccard similarity. The unified buckets fusion algorithm for both the event fusion module and tracking module simplifies the system implementation and improves

---

<sup>2</sup>Note that I-buckets have associated tweets.

the system maintainability. Inspired by Storyline [137], we use a sliding window, such that windows  $k$  and  $k - 1$  overlap as illustrated in Figure 5.3. In a typical configuration, a window of size 6 hours slides 1 hour at a time.

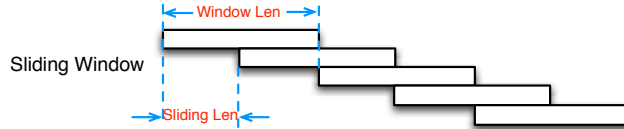


Figure 5.3: Sliding window

The output of merged buckets after fusion and tracking generates larger clusters. Each such cluster is associated with a unique ID. Buckets associated with a given ID can be displayed on demand, forming a chronological list of tweets and images that describe event evolution.

## 5.2 Evaluation

In this section, we first introduce the real-world dataset used in the evaluation and then discuss the evaluation methodology and results.

### 5.2.1 Datasets from Twitter and Instagram

In order to evaluate the performance of our algorithm, we collected real world datasets using Twitter and Instagram on *protest* events. We used the query word “protest” to crawl data from both social networks. Twitter provides an API to collect all tweets containing the query word. For Instagram, we used the web service `picodash.com` to collect all the Instagram images containing the query word as an image tag. The data was collected for a period of one month in February 2016, totalling 295,643 tweets from Twitter and 5,688 photo posts from Instagram. The weekly statistics of the data are shown in Table 5.1.

### 5.2.2 Methodology and Results

We evaluate two aspects of performance of our event tracking system that correspond to our two major contributions; (1) event detection by the fusion module and (2) event tracking, respectively. The baselines in the evaluation are underlying detectors used separately; that is to say, the event

Table 5.1: Statistics of collected datasets

<b>Week Index</b>	<b>#Tweets</b>	<b>#Instagram posts</b>
Feb 2016 Week 1	77001	1377
Feb 2016 Week 2	78334	1424
Feb 2016 Week 3	75639	1489
Feb 2016 Week 4	64669	1398

detector that exploits Twitter data (called “Twitter”) [137], and the detector that finds (Twitter-corroborated) Instagram events (called “Instagram”) [42].

### Event Detection by Fusion

Table 5.2 summarizes the precision and recall of all three algorithms at event detection during a randomly selected period of 7 days, where we manually labeled all ground truth. More precisely, since we do not know exactly how many events occurred that might have not been reflected in either data set, we abuse recall by referring to the absolute *number of true events* detected. From the table, as expected, we observe that the Instagram algorithm has the highest precision but lowest recall, whereas the Twitter algorithm has the lowest precision.

Table 5.2: Precision and recall

<b>Algorithm</b>	<b>Total# events</b>	<b>Precision</b>	<b>Recall</b>
Instagram	54	<b>87.037%</b>	47
Twitter	174	63.218%	110
Fusion	211	70.616%	<b>149</b>

We also investigate the F1 score for all algorithms, as shown in Figure 5.4. Since, we do not know the ground-truth total number of true events that occurred in each window (to properly compute recall), we plot the F1 score for different values of such total on the  $x$ -axis. From Figure 5.4, we observe that although the Instagram algorithm has the highest precision, its F1 score is consistently the lowest due to its poor recall. In contrast, our fusion-based solution has the highest F1 score, which means that it offers the best trade-off between precision and recall compared to the baselines.

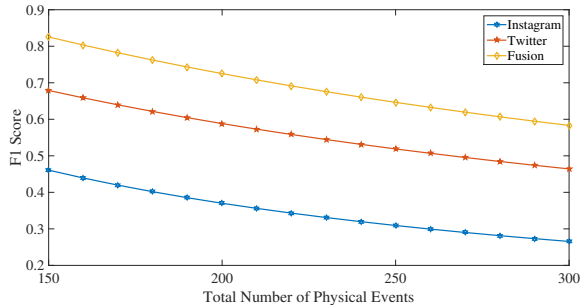


Figure 5.4: F1 score comparison with varied ground truth of total number of events

Next, we study the performance of demultiplexing (i.e., proper separation of different event instances). Ideal demultiplexing requires that an event detector output a separate event bucket for each event instance. Hence, the number of different events mentioned in tweets in any one bucket should be exactly 1. Accordingly, we use the average number of events (mentioned) per bucket as the metric to evaluate quality of demultiplexing.

We also prefer an event detector that outputs only one bucket for each true physical event. Such a detector has no redundant detections. We use the metric of the number of buckets per detected true event to evaluate redundancy. Note that, for both metrics, the closer to 1 the better in demultiplexing quality and detection redundancy.

Table 5.3: Demultiplexing quality and detection redundancy

Metric	Instagram	Twitter	Fusion
#events per bucket	<b>1.078</b>	1.290	1.194
#buckets per event	1.28	1.35	<b>1.24</b>

The results are summarized in Table 5.3. It shows that our solution has the best performance in terms of reducing redundancy. The intuition behind it is that our solution fuses I-buckets and T-buckets. It is possible that two T-buckets corresponding to the same physical event are correlated with the same I-bucket. They would thus be merged by our fusion algorithm thereby reducing redundancy in event detection. Same argument goes for the I-buckets as well. The Instagram approach remains the best at demultiplexing quality. This is attributed to the location-centric nature of Instagram clusters. A cluster of images from the same location is more likely to describe a single event.



## Event Tracking


For event tracking, we empirically observed that running the tracking algorithm on top of the I-buckets alone does not consolidate images of the same event over different windows. This might be because image tags on Instagram are chosen by users in a more independent fashion than the wording of tweets on Twitter. In order to observe the evolution of some event instance, we thus cannot exploit Instagram in isolation. Rather, we must also exploit Twitter.

In this section, we use a case study to showcase the performance of our event tracking. Table 5.4 shows a segment from a tracked event of Delhi protest that sabotaged water supplies for more than 10 million people. In the beginning of this event, people tended to tweet more about the fact, that is Delhi water supplies were sabotaged. The protest later became violent, and India sent soldiers to the area of protest. Next, the impact of this protest was estimated (that more than 10 million people in India were without water), and finally the protest terminated. There were also after-effects that we truncated due to page limits. Without the tracking capability, we could not be able to automatically stitch together the progression of this event. Table 5.4 also shows one I-bucket for this event and it was fused with a corresponding T-bucket. The case study demonstrates the effectiveness of our tracking solution and the feasibility of automatically merging posts from Twitter and Instagram about the same events.

## 5.3 Related Work

Event detection is an extremely popular research topic in the social network community. Much prior to the rise of social media, detection of objects with physical sensors has been studied extensively in the sensor network community [48, 82, 150]. The social sensing field in particular tries to tackle problems related to detection, localization, and tracking the events that occur in a physical space over a period of time. In the following subsections we briefly describe the contributions made in this field using the two popular social networks.

Table 5.4: Segment of a tracked event instance of Delhi protest

Window ID	Tweets Sample	Bucket Type
1	VIDEO: Delhi water supplies sabotaged by protest No water left in Delhi due to Jat protest, schools closed, rationing begins due to CASTE. . .	T-bucket
2	Water rationed as India caste protest toll rises	T-bucket
3	Caste violence . . . violent protest had briefly shut down the water supply in New Delhi. . . . that Jat started protest, they destroyed munak canal, no water for Delhi	T-bucket
4	Heaven help us all. Upper caste protest in Delhi leads to death and destruction. . .	T-bucket
5	India Sends Soldiers To Area Of Caste Protest, water cut off in New Delhi Caste Protests Near Delhi Close Roads and Restrict #Water Supply: Though the Indian Army	T-bucket
6	RT @fakingnews: Fed up Delhi youth start a protest against protests	T-bucket
7	More than 10 million people in #India’s capital are without water despite the army regaining control of its key water source after protest	T-bucket
8	 Haryana State in India Proposes New Caste Status in Bid to Quell Protests Jats protest leaves millions in New Delhi without water	Fusion bucket
9	Deal reached to end Jat protests in India’s Haryana state; roadblocks to be cleared, protest leader and police say	T-bucket

### 5.3.1 Event Detection using Twitter

Event detection is one category that has been widely explored with the help of *Twitter*. Topic modeling is a common approach for event detection [53, 74, 163]. Lau et al. [74] proposed an online variation of Latent Dirichlet Allocation (LDA) [18]. In LDA, each topic is modeled as a multinomial distribution of words in a vocabulary, and each document is modeled as a multinomial distribution of  $k$  topics, where  $k$  is a predefined parameter denoting the total amount of topics. And these two classes of multinomial distributions have two Dirichlet priors respectively. The idea in Lau et al. [74] is incrementally updating the priors in each time window based on the previous calculated

parameters, and maintaining the one-to-one correspondence of the topics in the current time window and the last one. If there is a sudden change in the topic word distribution, then a new event is supposed to be detected, where the distance of the distributions is measured by the Jensen-Shannon divergence. Hu et al. [53] proposed ET-LDA (joint Event and Tweets LDA) that exploits a search engine aligns tweets with the corresponding texts of events provided by traditional media, and they showed the results greatly improved. Zhou et al. [163] further expand LDA with time and location of the tweets, and proposed a new graphical model called location-time constrained topic (LTT). In their approach, the tweet content, timestamps and geo-tags are all considered. However, the topic modeling-based approaches usually suffer in the scenario that multiple event instances happen in parallel, even when they exploit meta data of the tweets like timestamps and geo-tags [11].

There have also been a few works in determining the reliability of the texts as well as the sources posting information on Twitter. In [127, 131] the authors have focused on the data reliability issue to find the true information from the noisy crowd data. The benefits of using humans directly as sensors include the capability of sensing information in high semantics in real time, which is not possible for physical sensors. However, due to the freedom of posting (almost) any content on social networks, the crowd data is usually very noisy containing rumors, partial information, or polarized viewpoints, which introduces the data reliability issue in social sensing. Wang et al. [127, 131] and a more recent work [140] proposed variant EM based algorithms to address the data reliability challenge by jointly estimating the data authenticity and source reliability. Tracking of events using Twitter was recently explored in [137] where real world datasets have been analyzed to find the evolution of subevents in time and space.

### **5.3.2 Event Detection using Instagram**

Instagram has emerged as a popular platform among researchers to analyze social networks from a crowdsensing point of view due to an explosive growth in the number of users. In [51], the authors have conducted a study to use Instagram as a social media visualization tool to identify cultural dynamics in major cities. The study particularly zoomed into the city of Tel Aviv, Israel, for a period of two weeks collecting images shared on important national event days. In [54], an analysis was presented to identify different types of users on Instagram and the categories of

pictures they take. The work characterized Instagram based on eight categories of pictures shared by five distinct types of users. Prior work [108] also described an approach capable of identifying important *tourist attractions* (POIs) with the help of Instagram. The focus of that work was to identify locations that are extensively visited by tourists. The authors of [52] described the implementation of a system capable of detecting events using geo-tagged data from networks that include Instagram. Their method determines a burst of keywords (tags) within a time interval, which is then modeled by Gaussians, and events are detected based on mapping the bursts. A very recent work [44] explores the techniques to detect and localize events in urban spaces. This work proposes an algorithm that focuses on using the distribution properties of the pictures related to an event in the time domain along with geo-coordinates to do an adaptive clustering followed by false positive elimination. There are a few other event detection techniques using Instagram but to our best knowledge no work has been done to track evolving events.

Contrary to all the related work, in this work we try to demonstrate the capability of jointly using Twitter and Instagram to not only detect events but also develop a deeper understanding on how these events evolve over a period of time. The combination of data from two different social networks results in better corroboration thus giving a higher precision over the baseline techniques.

## 5.4 Conclusions

We proposed an online event tracking system that integrates Twitter and Instagram data using an unsupervised approach that does not rely on language-specific features. Real-world data evaluation results demonstrate the effectiveness of the proposed system in event detection and tracking. Specifically, compared with two state-of-the-art baselines, our solution offers a better trade-off between precision and recall, lower instances of redundant event detection, and better monitoring of event evolution over time.

## Part III

# Information-Maximizing Delivery . . . .

## Chapter 6

# Coverage-based Information-Maximizing Data Delivery

This chapter introduces Minerva; a novel publish-subscribe-based programming system for optimizing information throughput in social sensing applications. Social sensing refers to the act of crowd-sourcing sensor data collection to volunteer participants in exchange for offering data services of interest. A common example is the collection and sharing of traffic speed data by drivers on different streets for purposes of computing speed maps that help plan individuals' commute.

We argue that development of social sensing applications calls for an *information-centric* programming paradigm in that the underlying run-time support is geared at maximizing *information flow*. This, as we show below, is not the same as maximizing *data throughput*. Social sensing applications fit a publish-subscribe model, where the sources involved in data collection are the publishers and the service that computes the quantities of interest is the subscriber.<sup>1</sup> Sources are typically mobile, such as phones or cars, and opportunistic WiFi offloading is used to reduce the cost of data upload (most data plans now charge for 3G/4G data upload, which makes it an unattractive choice for the sensing application). Hence, information propagates from one participant to another when they meet, and is uploaded to the subscriber when a participant has a free upload opportunity. Importantly, unlike the traditional publish-subscribe model, where publishers are independent, social sensing applications typically exhibit information overlap among sources. For example, vehicles waiting in the same traffic jam may collect very similar observations about traffic. Redundancy in data collection thus leads to inefficiency, which motivates a system that can recognize and eliminate the redundancy. Such a system would maximize information flow, as opposed to mere data throughput.

The main contribution of Minerva lies in its information-maximizing data prioritization scheme.

---

<sup>1</sup>The service also makes the computed results available, but this is done using standard dissemination techniques and is not the focus of this chapter.

It transmits publishers' data in an order that maximizes information flow. Hence, if data transfer is interrupted before all data are transmitted, a notion of information coverage is maximized for the given transfer size. The scheme is suitable for mobile environments where connectivity between nodes may be interrupted due to the nodes' mobility patterns and limited battery capacities. We show that without knowing the data transmission time in advance, which is the common case, no prioritization scheme can guarantee the optimal information throughput. Instead, an approximation bound is derived that is achieved by our prioritization algorithm, making it provably near-optimal.

From an API perspective, Minerva separates application-specific components from application-independent components. We recognize that *information* is a measure that may mean different things to different applications. To keep the information-maximization support in Minerva as application-independent as possible, we ask the programmer to define only one application-specific function per collected content type; namely, *a map function*, which takes a data object as operand and returns its position in a virtual space, called the *information space*, where objects that are closer to each other have more information overlap. When two Minerva nodes meet, they exchange content in an order that maximizes coverage in the information space.

An example map function could be one that places objects (that constitute sensor readings) in a space whose dimensions are the location and time of data capture. Hence, sensor readings at closer locations and times would be closer in the virtual space (which designates that such readings are more redundant). In general, other features may be considered as dimensions of information space. For example, in an application that measures temperature in campus buildings, a more meaningful set of features to consider might be time, building name, and room number. Hence, readings from the same time, the same building, and the same room number would be more redundant than readings from different times, different buildings or different room numbers. The map function would then map such measurements to a space where feature similarity translates into proximity. Once a map function is defined, information maximization, informally, becomes a problem of selecting points that are far enough apart in the information space, so that they are not redundant. The design of a good map function is an important application-specific problem. To keep the discussion in this chapter application-independent, we assume that a good map function,

for the application at hand, has already been designed and consider how to use it in order to implement an information-maximizing publish-subscribe service.

Minerva is implemented on top of the recently proposed Named Data Networking (NDN) framework [63]; a network paradigm where data objects are given unique names in a hierarchical name space (reminiscent of a UNIX directory structure), allowing the network to retrieve them by name. By giving collected data objects descriptive names (that encode the features of interest), we allow the map function to be a function of object names only. Hence, Minerva only needs to know data objects names in order to determine, with help of the map function, an information-maximizing transmission order.<sup>2</sup>

We evaluate Minerva using two smart-phone-based experiments as well as a large-scale simulation using the T-Drive dataset collected by Microsoft Research Asia (MSRA) [155]. Evaluation results demonstrate that our prioritization algorithm outperforms other candidates in terms of completeness of information delivered.

The remainder of this chapter is organized as follows. We compare our work with the state of the art in Section 6.1 and present the system design in Section 6.2. We formulate and solve our problem of maximizing information coverage in Section 6.3. The implementation and evaluation for our proposed solution are discussed in Section 6.4. Finally, we conclude the chapter in Section 6.5.

## 6.1 State of the Art

Social sensing attracted much attention in the research community since it was introduced in Burke *et al.* [22]. Examples of early services include CenWits [56], CarTel [57], BikeNet [33], PoolView [38], and GreenGPS [37]. *Application-specific* redundancy-eliminating sensing services, such as PhotoNet [118] and CARE [143], were proposed in earlier social sensing literature. In contrast, our work is the first to offer a *general application-independent* architecture that maximizes information *coverage*, while allowing customization (via the map function) to the specific application. We further design a novel data prioritization algorithm that is proven to maximize coverage subject to an approximation bound. The work is broadly related to the area of Information Centric

---

<sup>2</sup>The only element of NDN used by Minerva is that data has hierarchical names. In principle, Minerva can run on top of any system that offers a hierarchical, globally unique naming scheme.



Networking (ICN), investigated in recent years [40, 41, 117].

Liu et al. proposed a QoS-heterogeneous prioritization algorithm [84], to allow data packets with deadlines to be transmitted first in order to increase the possibility of offloading them faster. Previous efforts exist on redundancy elimination in networks including application-level [146] and packet level [14] techniques. Their work only try to eliminate redundant data, but do not consider the information carried in the data. For example, if two files have the very similar content, such as traffic speed measurements of the same street block at the same time, but different names, their work will consider these two files as different ones. However, these two files actually are redundant in information. Our work focus on eliminating redundancy in information.

We exploit the Named Data Networking (NDN) framework because it offers significant simplifications in the implementation of information-centric programming. NDN is recently proposed as a future Internet architecture, introduced by Van Jacobson [63, 160]. Since then, several papers investigated aspects of this framework such as suitability to ad hoc networking [91] and naming for mobile environments [133]. Our work is the first in proposing a programming API for social sensing applications that uses the NDN framework to simplify the maximization of information coverage.

## 6.2 System Design

This section describes the detailed system design. We first present the system model for social sensing applications, then explain the programming framework based on this model.

### 6.2.1 System Model

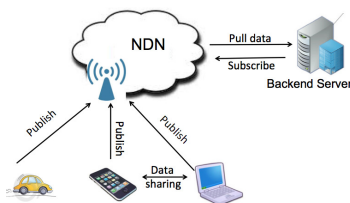


Figure 6.1: System model

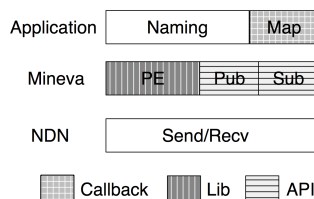


Figure 6.2: Programming framework

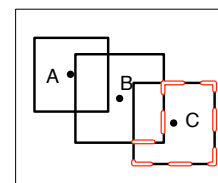


Figure 6.3: Illustration of coverage and marginal coverage with 2 features.

Figure 6.1 depicts the system model for our proposed social sensing applications. Mobile devices participate in these applications by generating and sharing sensory data, which are stored locally

and uploaded to a backend server via opportunistic WiFi offloading. Hence mobile devices serve as *publishers*, and the backend server acts as the *subscriber*. Opportunistic peer-to-peer communication might also be enabled to allow information to transparently propagate from one participant to another when they meet, in hopes of finding an offloading opportunity to the server faster. We adopt the NDN framework [63], thus data generated by users are identified by descriptive names.

## 6.2.2 Programming Framework

The programming support in Miverva is straightforward. Minerva provides a publish and a subscribe interface. Additionally, the application provides a callback function (one per content type), called `map()`, that takes as operand the name of a content object of a particular type and returns the corresponding position and coverage in a virtual information space. The position and coverage of a data point are used to compute its priority in transmission that maximizes information coverage in a resource-constrained environment. Objects are transmitted in the order of largest increase in marginal coverage as discussed in detail in Section 6.3.

As shown in Figure 6.2, an operational system would consist of three different layers: the application layer, the Minerva layer, and the network layer.

The application layer would take care of application-specific functions, such as content naming and publishing. Maintaining uniqueness of names is an application-specific concern not addressed in this chapter. A fully specified name refers to a unique item. Names can also be partially specified to designate a collection of items that share a common name prefix. In Minerva, publishers and subscribers refer to content collections by name when expressing availability of or interest in content.

In general, applications that use our publish-subscribe system own “subdirectories” in the global name space. For example, an application called *GreenGPS* might own the subdirectory “/root/GreenGPS”. The application might publish multiple types of content. Each part could start with “/root/GreenGPS/content-type”. Following the content type in the name comes a listing of content attributes of relevance to the map function. A type-specific map function can therefore parse the name to determine the attributes, and compute the coordinates of the object in virtual space accordingly. For example, an object might be called “/root/GreenGPS/content-type/location/time/filename”, where the location and time are the features of the data object.

In addition to the publish and subscribe functions, Minerva internally has a core function, *PE*, short for Prioritization Engine. *PE* reflects our optimization algorithm described in Section 6.3 to compute priorities for data objects such that they are transmitted in an order that contributes to maximum coverage.

The underlying network layer provides the communication functions across a network. In our implementation, we use NDN as the underlying network layer. Our solution does not require any changes to the standard NDN library (developed by PARC). Thus, it is general enough to be compatible with other existing NDN applications.

In the next section, we describe in detail the prioritization algorithm.

## 6.3 Information-maximizing Prioritization

In this section, we first introduce the definition of information coverage and formulate the information coverage maximizing problem. Then, we present the design and analysis of our algorithm.

### 6.3.1 Information Coverage

Data collected in social sensing applications are not independent; they exhibit correlations as discussed in the introduction. For the purpose of theoretical problem formulation, we assume that each data point covers a region in the *information space*, referred to as the *data coverage* region, defined below.

**Definition 2** (Data Coverage). *Suppose that there are  $k$  features of the data collected in a social sensing application. The Cartesian product<sup>3</sup> of domains of the  $k$  features forms a  $k$ -D information space. Any data point  $X$  with coordinates  $\langle x_1, x_2, \dots, x_k \rangle$  covers an interval  $I_j$  centered at  $x_j$  on the  $j$ -th dimension, where  $1 \leq j \leq k$ . The coverage of  $X$  is  $C_X = I_1 \times I_2 \times \dots \times I_k$ , where  $\times$  is the Cartesian product.*

Please note that data of different applications might have different coverage intervals. Given a particular application, the notion of coverage is usually clear. For example, when measuring temperature in a corn field, the “coverage” in the time dimension might be, say, 10-20 minutes,

---

<sup>3</sup>The Cartesian product of two sets  $A$  and  $B$  is a set  $C$ , such that  $C = \{\langle x, y \rangle \mid x \in A, y \in B\}$ . Similarly, we can define the Cartesian product of  $k$  sets.

since weather does not noticeably change in such a short time. Similarly, coverage in space might be 200-300 meters, since these distances are small enough not to dramatically affect temperature. Hence, given a temperature measurement at some time and location it can be assumed to remain valid for the entire coverage interval (in space and in time).

By Definition 2, the coverage of a data point is a  $k$ -D box as illustrated in Fig. 6.3. The coverage of a dataset  $\mathbb{S}$  is defined as  $\mathcal{C}_{\mathbb{S}} = \bigcup_{S \in \mathbb{S}} \mathcal{C}_S$ . The coverage of the intersection (*resp.* union) of two datasets  $\mathbb{S}_1, \mathbb{S}_2$  is defined as  $\mathcal{C}_{\mathbb{S}_1 \cap \mathbb{S}_2} = \mathcal{C}_{\mathbb{S}_1} \cap \mathcal{C}_{\mathbb{S}_2}$  (*resp.*  $\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{S}_2} = \mathcal{C}_{\mathbb{S}_1} \cup \mathcal{C}_{\mathbb{S}_2}$ ).

We define the *marginal coverage* of a data point  $X$  *w.r.t.* a dataset  $\mathbb{S}$  in Definition 3.

**Definition 3** (Marginal Coverage). *The marginal coverage of a data point  $X$  w.r.t. a dataset  $\mathbb{S}$  is the region in the information space covered by  $X$  but not covered by  $\mathbb{S}$ , i.e.,  $\mathcal{MC}_{X|\mathbb{S}} = \mathcal{C}_X - \mathcal{C}_{\{X\} \cap \mathbb{S}}$ .*

As illustrated in Fig. 6.3, the area surrounded by the dashed red line is the marginal coverage of data point  $C$  *w.r.t.* the dataset  $\{A, B\}$ . By definition,  $\mathcal{MC}_{X|\emptyset} = \mathcal{C}_X$ .

We define the value of the coverage of a data point in Definition 4.

**Definition 4** (Coverage Value). *The coverage value of a data point  $X$  in a  $k$ -D information space is the size of its  $k$ -D coverage region, defined as  $\mathcal{V}(\mathcal{C}_X) = \prod_{i=1}^k I_i$ , where  $I_i$  is the coverage interval in the  $i$ th dimension as in Definition 2.*

For example, if  $k = 2$ , the value of the coverage of a data point is simply the area of its coverage region in the information space. Similarly, definitions of the coverage value of a dataset and the marginal coverage value of a data point *w.r.t.* a dataset follow.

### 6.3.2 Problem Definition

A common goal of social sensing is to gather *information* that is as complete as possible. One trivial solution is that when a connection is established between two participants they sync all data, and when connecting to the backend server, a participant offloads its entire local data. However, due to the mobility and resource constraints (*e.g.*, energy), it is not always possible to sync or offload the entire dataset in a single transmission. Thus, in each transmission session, we aim to maximize

the marginal information coverage value of the subset of data that can be transmitted, referred to as the MAXINFO problem.

In the rest of this chapter, we shall assume that all data objects of the same type are of the same size. This is a common assumption in sensing applications. For example, in the context of a particular navigation application, all GPS readings have the same format and size. Similarly, in the context of a particular environmental sensing application, all temperature and humidity, measurements have the same format and size. In general, if the data format for sensory measurements is fixed, then all data records have the same size. This assumption simplifies terminology, allowing us to represent connection duration by a corresponding number of transmitted objects. It can be easily generalized to arbitrary object sizes simply by weighting each object by its size. Assuming same size objects, the the MAXINFO problem is formulated as follows:

**Problem 1 (MAXINFO).** *Suppose that there is a dataset  $\mathbb{S}_1$  (resp.  $\mathbb{S}_2$ ) on the data receiver (resp. the data provider). MAXINFO is to determine an order based on which the receiver should pull data from the provider such that for any data transmission size the receiver's information coverage is maximized. In other words, let  $\mathbb{R} \subset \mathbb{S}_2$  with cardinality  $n$  is the dataset pulled by the order, then  $\forall n, \forall \mathbb{T}, |\mathbb{R}| = |\mathbb{T}| = n, \mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{R}}) \geq \mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{T}})$ .*

Unfortunately, the unpredictability of the duration of each transmission session makes it *impossible* to find an order that is optimal for any  $n$ , which can be proved by a counter example as illustrated in Fig. 6.3. When  $n = 1$ , the optimal order is to select data object  $B$  first, since its coverage value is the highest. When  $n = 2$ , the optimal order is to select data objects  $A$  and  $C$  first, which conflicts with the optimal order when  $n = 1$ . Hence, we need to quantify the best one can do to approximate the optimal MAXINFO solution when one does not know connection duration in advance.

We first define the optimal solution  $OPT$  for MAXINFO to be an offline coverage-maximizing solution that assumes knowledge of the cardinality  $n$  in advance. It will constitute a theoretical upper bound, since such knowledge is generally not available online. Note that, as illustrated above,  $OPT$  may return different optimal orders for different values of  $n$ . Let us define the approximation ratio of an online solution  $A$  as follows:

**Definition 5** (Approximation Ratio). Consider a dataset  $\mathbb{S}_1$  (resp.  $\mathbb{S}_2$ ) on the data requester  $m_r$  (resp. the data provider  $m_p$ ). Let solution  $A$  of MAXINFO represent a fixed priority order for data object to pull from  $m_p$ . Let  $\mathbb{A}^n \subset \mathbb{S}_2$  denote the subset of data transmitted from  $m_p$  with cardinality  $n$  during the transmission session. Let  $\text{OPT}^n$  denote the subset output by OPT with  $n$  known in advance. The approximation ratio of  $A$  is

$$\tau = \min_{\forall n, 0 \leq n \leq |\mathbb{S}_2|} \frac{\mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{A}^n})}{\mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \text{OPT}^n})}.$$

Please note that for any fixed  $n$ , when  $\mathbb{S}_1 = \emptyset$ , the MAXINFO is exactly the weighted *Max  $n$ -Cover* problem [36].

**Theorem 2.** [36] If *Max  $n$ -Cover* can be constructively approximated in polynomial time within a ratio of  $(1 - 1/e + \epsilon)$  for some  $\epsilon > 0$ , then  $NP \subset \text{TIME}(p^{O(\log \log p)})$ , where  $p$  is the cardinality of the set (as  $|\mathbb{S}_2|$  in Definition 5).

Theorem 2 directly implies that achieving a better approximation ratio than  $(1 - 1/e)$  for MAXINFO is NP-hard. Thus, we have the following Corollary.

**Corollary 1** (Approximation Bound for MAXINFO). Achieving approximation ratio  $(1 - 1/e + \epsilon)$ ,  $\forall \epsilon > 0$  for MAXINFO is NP-hard.

### 6.3.3 Greedy Algorithm

In this section, we outline our prioritization algorithm. The idea of the algorithm is to give higher transmission priority to data with larger marginal coverage value *w.r.t.* the dataset at the receiver side.

We now prove that the approximation ratio of Algorithm 5 is  $(1 - \frac{1}{e})$ .

**Lemma 2.** For any  $n \leq |\mathbb{S}_2|$ , if  $\mathbb{R}$  is the set of the first  $n$  elements of the queue output by Algorithm 5, we have

$$\mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{R}}) \geq (1 - 1/e) \cdot \mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{R}'}) , \forall \mathbb{R}' , |\mathbb{R}'| = |\mathbb{R}| = n,$$

where  $\mathbb{S}_1$  (resp.  $\mathbb{S}_2$ ) is the dataset at the data receiver (resp. provider) side.

---

**Algorithm 5** Prioritization Algorithm

---

**Input:** Two sets  $\mathbb{S}_1$  and  $\mathbb{S}_2$ **Output:** An order of elements in  $\mathbb{S}_2$ 

- 1: Set  $\mathbb{T} \leftarrow \mathbb{S}_1$
  - 2: FIFO Queue  $\mathcal{Q} \leftarrow \{\}$
  - 3: **while**  $\mathbb{S}_2 \neq \emptyset$  **do**
  - 4:      $X \leftarrow \arg \max_{X \in \mathbb{S}_2} \mathcal{V}(\mathcal{MC}_{X|\mathbb{T}})$
  - 5:      $\mathbb{T} \leftarrow \mathbb{T} \cup \{X\}$ ,  $\mathbb{S}_2 \leftarrow \mathbb{S}_2 - \{X\}$ ,  $\mathcal{Q}.\text{inqueue}(X)$
  - 6: **end while**
  - 7: Return  $\mathcal{Q}$
- 

The proof of Lemma 2 is similar as that in [36], except that in [36]  $\mathbb{S}_1 = \emptyset$ . Hence, we do not repeat the proof here. Lemma 2 directly implies the following theorem.

**Theorem 3** (Approximation Ratio). *The coverage value of the transmitted set based on the order output by Algorithm 5 is a  $(1 - 1/e)$ -approximation of MAXINFO.*

Note that the approximation ratio matches the approximation bound in Corollary 1.

### 6.3.4 Transmission Protocol Design

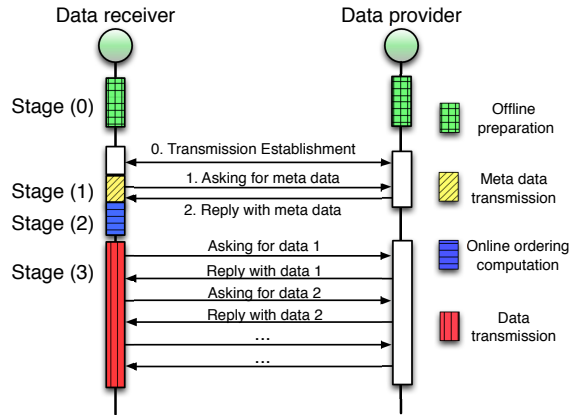


Figure 6.4: Transmission protocol illustration.

In this section, we present the transmission protocol, as illustrated in Fig. 6.4. Since we target mobile platforms, the transmission occurs in a disruption-tolerant (DTN) fashion; a device shares its data with a peer or offloads to a backend server when the corresponding connection is established. Thus, each transmission session (in which case we say the device is *online*) is followed by an idle session (when we say the device *offline*).

Each transmission session consists of three stages; (1) meta data transmission, (2) online ordering, and (3) data transmission. Stages (1) and (2) are the transmission overhead. Metadata, here, refers to the (information space) coordinates of data objects available at each node. In an NDN-based implementation, these coordinates can be computed from data names (using the map function). Hence, in our implementation, metadata refers to data object names. The idea being that data object names are generally much shorter than the data objects themselves. Hence, it makes sense to exchange the names first, then let each node specifically request from the other the named data objects it deems complementary (i.e., not redundant with) its own.

Before transmission, an offline preparation operation that generates the meta data needs to be carried out to reduce the overhead of the online ordering computation. We now present the offline preparation algorithm and online prioritization algorithm in detail as follows.

### Offline Preparation

The offline preparation stage outputs a meta data file which contains a list of data names as well as the overlap set of each *heavily informative* data point as described below. (The overlap set of data  $X$  contains any data  $Y$  s.t.  $C_Y \cap C_X \neq \emptyset$ .)

Consider two data points  $X$  and  $Y$ . If the coverage of  $X$  greatly overlaps with that of  $Y$ , then after  $X$  has been transmitted,  $Y$  carries little extra information. Thus, we introduce a constant threshold  $\beta > 1$ , such that, when the distance between  $X$  and  $Y$  is smaller than  $\frac{1}{\beta}$  we only need to consider one of them (*w.l.o.g.*, say  $X$ ) in the online prioritization algorithm. The other is put in the lowest priority bin for transmission. We apply this rule repeatedly until no more points can be assigned lowest priority. The surviving points (not assigned lowest priority) are called *heavily informative* data points. Note in particular that if  $X$  has no neighbors  $Y$  whose distance from  $X$  is smaller than  $\frac{1}{\beta}$ , then  $X$  will never be assigned lowest priority and is therefore a heavily informative data point. The heavily informative dataset contains all the data points like  $X$ .

Our offline preparation algorithm determines for each new data point whether or not it is heavily informative. If it is heavily informative, it adds the point to the metadata file to be exchanged on contact with another node. It is incremental in the sense that when new data points arrive, we do not need to redo the preparation for old data. The pseudocode for the offline preparation



algorithm is presented below.

---

**Algorithm 6** Preparation Algorithm

---

**Input:** Existing dataset  $\mathbb{S}$ , existing meta data file, newly arrived dataset  $\mathbb{T}$

**Output:** Updated meta data file

- 1: From meta data, get the heavily informative dataset  $\mathbb{H}$  of  $\mathbb{S}$
  - 2: Sort  $\mathbb{H}$  based on the lexicographical order of data coordinates in the  $k$ -D information space
  - 3:  $\mathbb{D} \leftarrow \emptyset, \mathbb{N} \leftarrow \emptyset$
  - 4: **for**  $\forall S \in \mathbb{T}$  **do**
  - 5:     Use binary search to find its overlap set  $\mathcal{O}_S \subseteq \mathbb{H}$
  - 6:     **if**  $\exists E \in \mathcal{O}_S, s.t. S \simeq E$  **then**
  - 7:          $\mathbb{D} \leftarrow \mathbb{D} \cup \{S\}, \mathbb{T} \leftarrow \mathbb{T} - \{S\}$ , **continue**
  - 8:     **end if**
  - 9:     Add  $S$  to the overlap set of any element in  $\mathcal{O}_S$
  - 10:     Insert  $S$  into  $\mathbb{H}$  *s.t.*  $\mathbb{H}$  remains sorted
  - 11:      $\mathbb{N} \leftarrow \mathbb{N} \cup \{S\}$
  - 12: **end for**
  - 13: Add the name following by the overlap set of each data in  $\mathbb{N}$  to the front of the meta data file
  - 14: Append names of data in  $\mathbb{D}$  to the end of meta data file
  - 15: Return the meta data file
- 

In our online ordering stage, we only need to consider the heavily informative dataset. The parameter  $\beta$  controls the cardinality of the set of highly informative data. The smaller  $\beta$  is, the smaller the cardinality of this set. In practice, we can use  $\beta$  as a knob to trade-off the accuracy and the time efficiency in the online prioritization as discussed below.

### Online Prioritization

Online prioritization is described in Algorithm 7. After metadata (i.e., names of heavily informative objects) have been exchanged, the receiver calculates the marginal coverage value of each data object in the highly informative set *obtained from the data provider*, and puts these values into a max heap. Then, it sends a request for the data object  $D$  popped from the max heap, and at the same time updates the marginal coverage value of each data object in the overlap set of  $D$  and do the standard heapify. This process continues until the max heap is empty, then, if the connection is still up, the receiver starts to pull data that not in the highly informative set in FIFO order.



### 6.4.1 Experimental Setup and Methodology

Minerva is designed for social sensing applications with resource constraints. Thus, we need to evaluate two aspects of the system: (i) the overhead of data prioritization, and (ii) the application performance, measured in terms of application-level metrics; namely, information coverage. The following methodology was used for evaluation:

- *Data prioritization overhead:* In order to measure overhead under a wide set of workloads, we generate synthetic load (i.e., synthetic data to be transmitted) that can be easily parameterized to represent a large set of relevant properties. These properties include the size of the data set, the dimensionality of the data, and the degree to which the data is redundant. We then test the overhead of prioritizing such data on real phones.
- *Application-level performance:* In order to evaluate application-level performance metrics (namely, coverage), we find an actual data trace of a participatory sensing application. We then compare coverage when Minerva is used and when other data transmission schemes are used, given a simplified network simulator.

To accomplish the above, we implemented Minerva on Google Galaxy Nexus smartphones [2], equipped with a 1.2 GHz dual-core CPU, 1GB RAM, and 802a/b/n Wifi radio with Android OS 4.1. Minerva is implemented using the Java language on top of PARC’s CCNx prototype software [1]. The overhead study is conducted in an outdoor environment on real phones. The application performance study uses a real-world taxi trace dataset, the T-Drive dataset [6,154,155], which contains the GPS trajectories of 10,357 taxi cabs during the period from February 2nd to February 8th, 2008 in Beijing. The total number of points in this dataset is about 15 million and the total distance of the trajectories is around 9 million kilometers. The trajectories covered are shown in Figure 6.5.

### 6.4.2 Overhead of Minerva

A key goal in the overhead study is to understand the overhead of data prioritization (which includes the overhead of metadata transmission for purposes of computing priorities) for a wide set of workloads. Hence, synthetic data is used. In this study, workload generation does not attempt

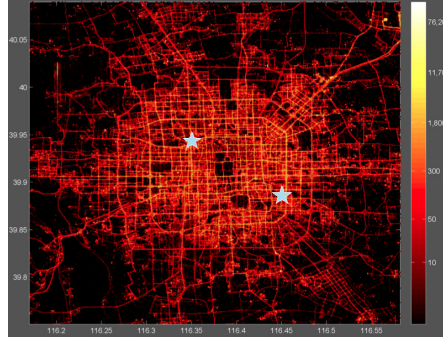


Figure 6.5: tdrive data used in simulation

to mimic characteristics of any specific application. Rather, it attempts to investigate overhead under a broad range of conditions that affect it. These include, the size of the data set (in terms of the number of objects), the dimensionality of data, and the degree of redundancy among data items.

To explore the effect of data dimensionality, we generate data by (uniformly) sampling from a  $k$ -dimensional box of unit size in the information space, where  $k$  is a configurable parameter that represents the number of features (i.e., information space dimensions) considered in the Minerva prioritization algorithm. We use a unit box and uniform sampling because it allows us to easily control the degree of data redundancy by tuning the value of coverage interval associated with individual data points. We focus on overhead only (as opposed to the time it takes to send the data objects). Hence, we measure the overhead of sending metadata and computing priorities only. At the of this overhead all objects are properly prioritized and ready for transmission. The results of the overhead study are shown in Table 6.1.

The data set parameters considered in the table are (1) the number of features, (2) the number of data points, (3) the coverage interval per point (and hence degree of redundancy of data), and (4) the value of  $1/\beta$  as defined in Section 6.3. The more data points are considered, the more computation is needed for data prioritization. Similarly, the larger the coverage interval of individual data points, the higher the redundancy (or the probability that two data points overlap in coverage), and hence the higher the computation overhead of redundancy-minimizing prioritization. The value of  $1/\beta$  is a parameter of our algorithm that indicates its tolerance of imprecision in redundancy minimization. The higher the  $1/\beta$ , the more approximate the prioritization, and the

lower the data prioritization overhead, as discussed in Section 6.3. Rows 1-12 of Table 6.1 show the total time in metadata exchange and prioritization between two android phones that have the same amount of data points on both phones. Rows 13-16 show the corresponding overhead when an android phone uploads data to a backend server with a 3.10GHz CPU and 8GB RAM.

Table 6.1: Overhead of Minerva

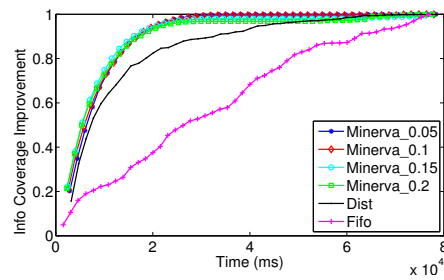
index	dataset features (# dim, # points, interval, $1/\beta$ )	overhead(s)
1	2, <b>250</b> , 0.05, 0.1	$0.321 \pm 0.165$
2	2, <b>500</b> , 0.05, 0.1	$0.837 \pm 0.208$
3	2, <b>750</b> , 0.05, 0.1	$3.070 \pm 1.240$
4	2, <b>1000</b> , 0.05, 0.1	$7.205 \pm 2.579$
5	2, 500, <b>0.01</b> , 0.1,	$0.339 \pm 0.071$
6	2, 500, <b>0.03</b> , 0.1	$0.582 \pm 0.104$
7	2, 500, <b>0.05</b> , 0.1	$0.837 \pm 0.208$
8	2, 500, <b>0.07</b> , 0.1	$1.667 \pm 0.320$
9	2, 500, 0.05, <b>0.15</b>	$0.700 \pm 0.140$
10	2, 500, 0.05, <b>0.2</b>	$0.626 \pm 0.145$
11	<b>3</b> , 500, 0.05, 0.1	$0.257 \pm 0.093$
12	<b>4</b> , 500, 0.05, 0.1	$0.204 \pm 0.040$
13	2, (10000, 500), 0.01, 0.1	$0.076 \pm 0.044$
14	2, (100000, 500), 0.01, 0.1	$1.152 \pm 0.093$
15	2, (1000000, 500), 0.01, 0.1	$4.773 \pm 0.537$
16	2, (1000000, 500), 0.01, 0.2	$0.727 \pm 0.104$
17	Wifi connection establish time	$2.002 \pm 0.106$

From the table, we observe that the overheads increase as the number of data points increase (rows 1-4) or as the coverage interval increases (rows 5-8), but decrease as  $1/\beta$  increases (rows 9-10), which corroborates expectations. When the number of data points is 1000 on both phones (row 4), Minerva takes about seven seconds to prioritize and all objects, which is unacceptable. The overhead drops off sharply with size of the data set (rows 1-3). With 500 objects (row 2), the overhead is less than one second, which is tolerable. Measurements reported in the next section show that one can send roughly 250K bytes during that time. Hence, if objects are 250K bytes long, the overhead of prioritizing 500 objects is roughly equal to the transmission time of one object. In other words, the prioritization overhead is acceptable as long as the individual objects are sufficiently long.

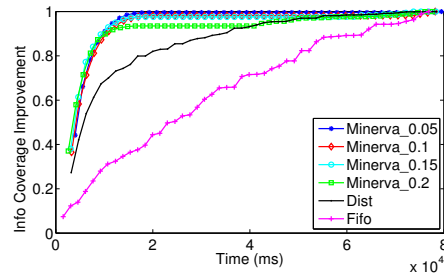
The effect of the number of features on overhead is shown in row 11 and 12 in the table. It can be seen that the overhead decreases in higher-dimensional spaces (all else being equal), because for the

same number of data points and the same coverage interval, higher dimensionality means a sparser space, and hence less redundancy, and less overhead for redundancy-minimizing prioritization.

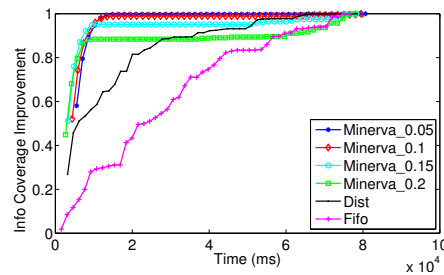
The overheads when data is offloaded from a mobile device to the backend server are shown in Row 13 to 16. The number of data points is set to 500 on the participant side, and the number of data points on server side is set to be 10,000, 100,000, and 1,000,000. We observe that as the number of data points increases on the server side, the overhead grows. We can also observe that the slope of overhead increase becomes smaller when the number of data points at the server side becomes larger. The reason is that the coverage improvement grows submodularly; when the server already got a large enough amount of data, the probability that a new data point is redundant is close to 1.



(a) Coverage interval 0.063



(b) Coverage interval 0.077



(c) Coverage interval 0.089

Figure 6.6: Performance of Minerva with different coverage intervals and  $1/\beta$  values in phone-based experiments with synthetic data.

Next, we study (in Fig. 6.6) the coverage achieved with Minerva transmissions between two smartphones using the synthetic data. The number of data points is set to 500 on both phones, and the coverage interval is set to be 0.063, 0.077, and 0.089 (thus, in expectation, one data point overlaps with 2, 3, and 4 other points respectively). For each interval value, we plot the coverage improvement using Minerva (with  $1/\beta \in \{0.05, 0.1, 0.15, 0.2\}$ ), Dist (a distance-based data selection algorithm used in PhotoNet [118]) and FIFO. The x-axis is the time in milliseconds that starts right after the connection is established. The y-axis denotes the normalized information coverage at the receiver side, where a coverage of 1 is equivalent to transmitting all sender data. Remember that the objective of prioritization with Minerva is maximize the coverage of transmitted data (i.e., achieve close to 1 coverage as early as possible during transmission).

From Fig. 6.6, we observe that Minerva outperforms the other algorithms in general in that it achieves higher coverage earlier on. The larger the coverage interval of individual objects, the better Minerva performs. From the figure, to get 80% coverage, Minerva uses 10 (8 and 5 *resp.*) seconds for a coverage interval of 0.063 (0.077 and 0.089 *resp.*). Dist uses around 20 seconds to achieve 80% coverage, while FIFO takes more than 50 seconds. Minerva coverage also decreases somewhat with increased  $1/\beta$  due to the approximation involved.

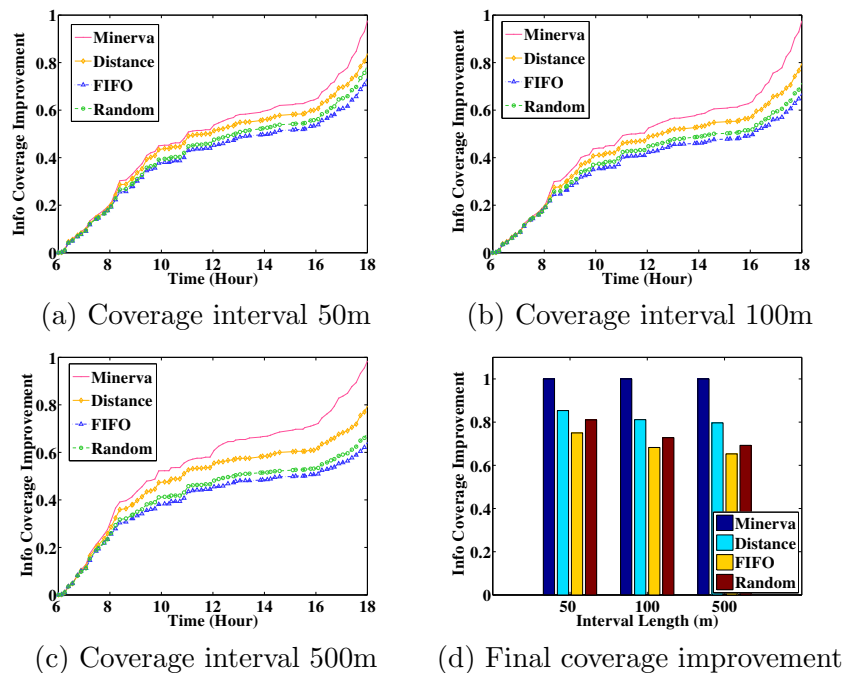


Figure 6.7: Performance of Minerva with different coverage intervals in large-scale real-world GPS trace simulations.

### 6.4.3 Large-scale Trace-based Evaluation Results

In order to test application-level performance with Minerva in large-scale applications, we emulate a hypothetical *real-time street view* application. This application applies social sensing in a future where vehicles are equipped with cameras. Participants are requested to send pictures from their car’s cameras to the base station when they encounter an access point. These pictures are then used on the server to provide a real-time view of city streets on demand.

In this evaluation, we run simulations on real taxi traces in Beijing (the T-Drive dataset). We consider data within the area from latitude  $39.5^{\circ}N$  to  $40.5^{\circ}N$  and from longitude  $116^{\circ}E$  to  $117^{\circ}E$ , where most data resides. In the simulation, we assume that there are two sinks that collect data for the backend server as indicated by the two stars in Fig. 6.5. They are located in two relatively busy roads, where cars can have a higher chance of offloading their data. Cars are assumed not to share data with each other. In our simulation, if the distance between a car and a sink is smaller than 200 meters, we assume that the car can offload data. We assume that each data point (a picture) is 1M bytes long.

We determine the transmission duration of each car by examining its speed when it enters the transmission range of a sink; the speed can be estimated from the time and location information of the latest GPS samples.

In order to obtain a realistic WiFi transmission rate, we conduct a small experiment where we use a smartphone to send a long file over Wifi in an outdoor environment to a desktop in our lab. The average data transmission rate is found to be approximately 250KB per second, and is set accordingly in the simulation.

In our simulations, we consider two features per data sample, the latitude and the longitude. The coverage interval for each data point is set to 50, 100, and 500 meters respectively in subsequent simulations. We simulate for 10 hours’ data (1,500,000 points) and assume at the very beginning the server does not have any data.

The results are shown in Fig. 6.7. From Fig. 6.7, we observe that at the beginning of data collection, the four algorithms compared yield similar performance. Minerva is slightly worse than the others due to its overhead. After collecting data for one hour, Minerva begins to outperform the others. The reason is that Minerva is designed for eliminating redundant data. At the beginning,



there is little redundancy since the server has only a limited amount of data. Hence, the overhead is not paying off in terms of application-level metrics. As more data is collected, more redundancy is exploited by Minerva. Eventually, Minerva outperforms all other algorithms in terms of attained coverage. Note that the slope of the coverage curves changes over the course of the day (where the x-axis is time of day). This is because more data is collected during rush hour, roughly between 6-8am and 4-6pm. In summary, evaluation shows that redundancy-minimizing data prioritization has promise in terms of improving coverage of the physical environment in social sensing. However, overhead remains an issue, which reduces applicability except where sensed objects are sufficiently large (e.g., multimedia objects).

## 6.5 Conclusions

In this chapter, we presented Minerva; an information-centric programming paradigm and toolkit for social sensing. Minerva is geared for social sensing applications, where different sources (participants sharing sensor data) often overlap in information they share, which distinguishes them from a regular publish-subscribe system where publishers are independent. One contribution in this chapter lies in an algorithm for maximizing information delivery from publishers to subscribers taking into account the *non-independent* nature of content. We analytically prove that our transmission prioritization scheme is within a constant approximation ratio from a “clairvoyant” optimal solution. We develop a programming toolkit to embody our prioritization scheme. The scheme is implemented over NDN. To the best of our knowledge, this is the first work that leverages the benefits of NDN in maximizing information coverage for social sensing applications. Evaluation results show that our algorithm outperforms other candidates in terms of information coverage for large data objects.

## Chapter 7

# Tree-based Information-Maximizing Data Delivery

The expanding proliferation of sensors available in social spaces (such as smartphone sensors, cameras, and GPS devices) and the exponential growth in digital data generated in recent years, far outstrip the human capacity to consume the resulting information. This trend suggests that an important category of future networked applications and services will focus around information sampling to bridge the widening gap between data generation rate and human consumption capacity.

Current transport abstractions, such as reliable transmission in TCP, offer pipes where each bit of input must be delivered at the output. In the future, driven by information overload, a new higher-level abstraction will become increasingly important: namely, one that offers at the output a *representative sampling* of information at the input, thereby reducing the large body of input to a readily consumable size. For wider applicability, this sub-sampling must be done in an application-independent manner. Nevertheless, it must do better than random selection. The *information funnel* implements such an abstraction.

The information funnel is targeted for scenarios, where resource constraints (e.g. limited transmission bandwidth or constrained power) or efficiency considerations prevent transmission of *all* collected data. In these scenarios, with limited number of data objects can be transmitted, the information funnel tries to sample a representative subset of the data objects that maximizes the information utility by minimizing data redundancy. Much prior work on data collection in sensor networks addressed the challenge of optimal data selection (based on different application-level metrics) (e.g. [85]), that judiciously chooses the best data objects to transmit when transmission of all data is impossible or undesirable. Unlike ours, such protocols are application specific, as they use application-specific information and optimize application-specific performance metrics. Therefore, their work is not general; it will have a poor performance or even not work in a different

application.

We also distinguish ourselves from work on sampling theory that determines how to sub-sample time series data in ways that generically minimize a measure of loss (e.g. [50] [83]). In contrast to these approaches, and in seeking a general service, we do not impose any specific requirements on the underlying data type. For example, the collected data may constitute images, text, or sound clips, as opposed to numeric data types.

The chapter complements the aforementioned literature by exploring the potential and limitations of *application-independent* maximization of delivered information utility (that is to minimize delivered redundant data). By application-independent, we mean that the solution does not use any application-specific knowledge or semantics. It only uses the hierarchical data names, treated as bit strings with no semantic interpretation. This is a major difference from sampling theory that requires understanding the application-specific semantics of data objects.

We exploit the *named-data networking* (NDN) paradigm [64] as an enabler for information utility maximization. NDN is originated because of the fact that people care more about *what* data they received but not *where* they get the data. Therefore, NDN names data objects, not hosts, which distinguishes it from the mainstream communication paradigm based on TCP/IP. In NDN, the information consumer (e.g., the data collection point) sends interests in information objects described by a given name prefix. Objects that match the specified prefix are returned in response to the respective interests. The information funnel is implemented on top of NDN as a thin layer that decides on the order of data transmission.

The chapter investigates (and confirms) the hypothesis that *by giving data objects hierarchical names, where the length of the common prefix between two names is a rough measure of similarity between the corresponding objects, information-maximizing ordering can be achieved using policies that diversify the transmitted names*. Here the similarity between two data objects refers to the possibility that the value of one can be approximated by the other's. Our service is geared for social sensing applications in which a receiver (such as a remote back-end server) acts as a collection point for a group of (typically mobile) nodes that report data from the physical environment. Often the nodes are disconnected and come only into sporadic contact with the collection point. The data collected usually carries much overlap. For example if data names encode location and time of

data collection, the more similar the names are, the more likely the overlap between the named measurements and the less is their aggregate utility. A data collection protocol that *diversifies the collected names* will tend to maximize information utility as well. It remains to show how exactly names should be diversified in the presence of resource constraints, which is the topic of this chapter.

The rest of the chapter is organized as follows. Section 7.1 presents our notion of optimality and suggests heuristics with near-optimal behavior. Section 7.2 presents evaluation results. Section 7.3 reviews related work. The chapter concludes with Section 7.4.

## 7.1 The Information Funnel

Social sensing applications share in common the fact that they (i) collect data objects from one or more (typically mobile) sources, (ii) do not need all sender data to operate correctly, and (iii) perceive a quality of information wherein receiving more data on the same “topic” has diminishing return. For example, to estimate the current speed of traffic on city streets, it is sufficient to obtain representative speed measurements from a subset of vehicles. More data will have diminishing return. Similarly, in a disaster-response application where first-responders pictorially document damage and report it to a rescue site, only a few pictures of each problem spot are needed to understand the situation. More pictures have diminishing return. This motivates the *information funnel* abstraction, described below. We discuss challenges in implementing it and define a notion of optimality. Finally, we present its design and implementation on top of a named-data-networking stack.

### 7.1.1 The Basic Abstraction

The information funnel targets applications that implement persistent data collection tasks. We require that data objects have a hierarchical name space. A funnel is associated with a name prefix in that space (analogous to a path prefix in a UNIX directory tree), defining the subtree in which data of interest to the application resides. Content that belongs to the subtree starting with that prefix is the target of collection. Senders publish content under appropriate names. If those names fall within the target subtree, the corresponding objects become targets for collection.

For example, in an urban traffic speed monitoring application, the name space might look something like this: `/ndn/app/city/street/block/speed`. The collection point creates a funnel (defined by the name prefix of the above tree, say `/ndn/app`) to populate this space with data from senders. The design of the name space is up to the application developer. In applications where mobile entities share a physical environment in which they measure some quantity, such as cars measuring traffic speed, the name space might associate names with parts of the physical environment (e.g., street blocks). In this case, mobile sensors will assign data names depending on what part of the environment they are sensing. In applications where sensors are fixed, such as security cameras in rooms, data names may be associated with sensor IDs.

An important goal of our design is to accommodate mobility and disconnected operation. Hence, we assume that the normal state of senders is “offline”. For example, mobile sensors may not have connectivity until they meet an access point. Smart phone users may disallow an application from using their 3G/4G data plan quota. First responders in a post-disaster scenario may communicate only using short-range radios, and thus be disconnected unless in close proximity, because other communication infrastructure is out of power or destroyed.

In general, at a given time, the receiver has a partially populated content tree. When a sender has a transmission opportunity (e.g., encounters an access point), the receiver needs to be updated on any new data the sender has, under that tree, that the receiver has not yet received. Two interesting questions arise: how to inform the sender efficiently of data gaps at the receiver, and in what order should such missing data objects be sent? Below, we first describe the underlying challenges, then define a notion of optimality and present our algorithm.

### 7.1.2 Data Ordering Challenges

To implement an update between a sender and the receiver of the funnel, the trivial solution is to have each sender collect data under `/ndn/app` and forward it when possible to the receiver. Once data is delivered, it can be discarded at the sender locally or marked as delivered. This solution works well when senders populate *non-overlapping* parts of the content tree, and when they do not exchange their collected data among themselves for uploading to the server. In this case, each sender can easily tell which of its data does not yet reside on the receiver. In social sensing

applications, many senders may report data on the same event. Hence, sending all of one’s newly collected data to the receiver may be suboptimal because the receiver may have already received that (or similar) content from another sender. The receiver needs to tell the sender (either exactly or approximately) what information it already has.

Summarizing the receiver’s information state to the sender is easy when data has a linear order (e.g., “I have all data up to time-stamp X”). This, unfortunately, is not true in our case. Two factors compound our problem. First, the receiver may have only partial data that populates the name space sparsely. Hence, many gaps exist in data coverage, making their exact enumeration hard. Second, the receiver may not know the totality of data generated under a given name subtree. For example, in our vehicular sensing application, a receiver can never tell that it has “all data from Main Street” because it does not know how many vehicles drove on Main Street that may not have uploaded their data yet. Hence, there is no easy way to prune subtrees from consideration on account of completeness.

Another question is regarding the order in which a sender should send the data that its receiver is missing. If the sender sends such data in the order it was collected, the receiver may receive a lot of data from one branch of the name space and no data from other branches, resulting in a very unbalanced coverage of content. Instead, it is better to diversify by sending a sampling of data from each branch. The need for diversification calls for a definition of optimal information utility to guide the data ordering algorithm, as formulated below.

### 7.1.3 Optimal Transmission Order

Consider the problem where a sender must order a set of data objects for transmission to a receiver (that fall within the content space of the funnel). An optimal transmission order is sought, where the utility of the transmitted data to the receiver is maximized. In this section, we formulate this problem more carefully and describe our solution. We initially assume that all objects have the same size and the same importance (weight). In subsequent sections, these assumptions will be removed.

A primary design objective is to keep the formulation *as simple as possible*, since quantities such as data utility are notoriously hard to compute exactly. Clearly, if utilities are set subjectively or

arbitrarily, then optimizing them does not make much sense. To render the problem of finding an optimal transmission order meaningful, we must seek an approach *that makes minimal assumptions about utility curves*. For example, we explicitly stay away from schemes that require computing absolute utility values for data objects, since those are subjective.

Instead, we assume that the following two properties hold regarding the marginal utility of data objects at different parts of the content tree:

**The hierarchical similarity property** The hierarchical name space is designed such that items that share a longer name prefix (measured in the number of tree levels) are more similar. By similarity, we mean that one can approximately be substituted by the other. For example, speeds at `/app/urbana/main/1200` and `/app/urbana/main/1000` are more similar than speeds at `/app/urbana/main/1200` and `/app/urbana/green/1100` since the former pair shares a longer common prefix. A corollary is that the marginal utility of a data object increases with the decreasing length of the longest common prefix between itself and any of the previously collected items. This is because the smaller that prefix, the less substitutable the item is by any of the ones already collected, thus the higher marginal utility it has.

**The diminishing return property** The marginal utility of adding a data object to a name subspace is diminishing; adding the first item to `/ndn/app/urbana/main` has a larger marginal utility than adding the second item, which in turn has a larger marginal utility than adding the third one, and so on.

The *hierarchical similarity assumption* implies that, at each step, the optimal transmission order must pick the data object whose maximum common prefix (with all previously collected ones) is shortest. If there is a tie between two (or more) such items, the *diminishing return assumption* implies that we pick the one that has the least populated prefix. In other words, we count how many data objects were collected under each prefix, then pick the one whose prefix has the smaller count. We call it the *occupancy count* of the prefix. If there is a tie again, we break the tie by picking the item on the left-most branch (which is the one with the most recent timestamp assuming branches are chronologically sorted).

More formally, let  $\ell_i(\mathcal{J})$  denote the longest common name prefix of a data object  $i$  with respect

to a data set  $\mathcal{J}$ ,  $|\ell_i(\mathcal{J})|$  denote its length, and  $\overline{\ell_i(\mathcal{J})}$  denote its occupancy count. We denote the marginal utility of a data object  $i$  with respect to a data set  $\mathcal{J}$  as  $\mathbb{U}(i|\mathcal{J})$ . Given any two data objects,  $i$  and  $j$ , and a data set  $\mathcal{J}$ , we can compare the marginal utilities of the two data objects as follows.

**Marginal utility comparison rules:**

- If  $|\ell_i(\mathcal{J})| < |\ell_j(\mathcal{J})|$ , then  $\mathbb{U}(i|\mathcal{J}) > \mathbb{U}(j|\mathcal{J})$ ,
- Otherwise, if  $|\ell_i(\mathcal{J})| = |\ell_j(\mathcal{J})|$  and  $\overline{\ell_i(\mathcal{J})} < \overline{\ell_j(\mathcal{J})}$ , then  $\mathbb{U}(i|\mathcal{J}) > \mathbb{U}(j|\mathcal{J})$ ,
- Else (*i.e.*, if  $|\ell_i(\mathcal{J})| = |\ell_j(\mathcal{J})|$  and  $\overline{\ell_i(\mathcal{J})} = \overline{\ell_j(\mathcal{J})}$ ),  $\mathbb{U}(i|\mathcal{J}) = \mathbb{U}(j|\mathcal{J})$ .

An optimal transmission order is one *that maximizes the marginal utility, for every count  $k$  of transmitted objects, over all ways of picking  $k$  objects for transmission*. A greedy solution is to transmit the object with the minimum  $|\ell_i(\mathcal{J})|$ . In case of a tie, transmit the object with the minimum  $\overline{\ell_i(\mathcal{J})}$ . In case of a second tie, break the tie arbitrarily (e.g., transmit the left-most child). In the following section, we illustrate our idea through an example and discuss optimality.

**7.1.4 An Example**

An example of the proposed algorithm is shown in Figure 7.1. The square boxes indicate data objects at the leaves of the content tree. The circles are intermediate nodes (directories) in the name space.

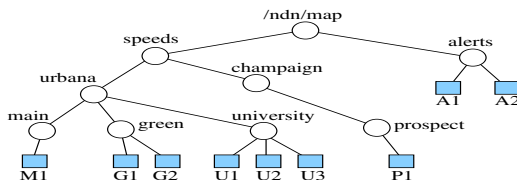


Figure 7.1: An example

Let the receiver have no data objects initially. There is a tie between all the items in that none share a common prefix with what the receiver has, and the name space has zero occupancy count. Picking the left-most branch, we send item M1 first. Next, the items that minimize the longest the common prefix with M1 are A1 and A2. Their longest common prefix with previous items (*i.e.*, M1) is the same; namely, `/ndn/map`, which has zero occupancy count. The tie is broken by following



the leftmost branch (i.e., send **A1**). The next items that minimize the longest common prefix with those transmitted earlier are **P1** and **A2**. Their respective longest common prefixes with earlier items are `/ndn/map/speeds` and `/ndn/map/alerts`. Tying on occupancy count (namely, one item was transmitted under each prefix), the leftmost item (i.e., **P1**) is transmitted next. Following that, **A2** minimizes the longest common prefix and is transmitted next. The next items that minimize the longest common prefix with those transmitted earlier are **G1**, **G2**, **U1**, **U2**, and **U3**. They tie on occupancy of their prefixes and so the leftmost one is transmitted (i.e., **G1**). Next, items **U1**, **U2**, and **U3** tie and **U1** is transmitted. It can be seen that, following the above logic, we then transmit **G2**, **U2**, and **U3** in that order.

### 7.1.5 Receiver Feedback

In the previous section, we have not addressed the case where the receiver already has a partially populated name space. To accommodate this scenario, when a sender comes in contact with the receiver, the receiver first sends the sender a packet that contains occupancy counts of all prefixes up to a configurable tree level  $n$ . The sender will initiate the occupancy number of the name tree based on the receiver's feedback. The initialization will cause transmission to favor data that resides in prefixes that the receiver has less (or no) data from. Consider again the example shown in Figure 7.1. Assume that the receiver already has items **G0** and **P0** that reside at the same prefixes as **G1** and **P1**, respectively. The transmission order will be **A1**, then **A2** (minimizing longest common prefix with previously collected items and favoring the prefix with lowest occupancy count), followed by **M1** and **P1** (since they are the next to minimize the longest common prefix with previously collected items), then **U1**, **G1**, **U2**, **G2** and **U3** (tie on longest common prefix, so round robin on occupancy count).

### 7.1.6 The Algorithm

In this section, we present the prioritization algorithm and prove its optimality. Its time complexity analysis is in the appendix. At the first look of the above tree traversal, it seems that our algorithm is just a simple breadth-first traversal, with round robin. However, after a careful examination, the breadth-first traversal does not always give the optimal prioritization by the marginal utility

comparison rules; it only guarantees optimality when all the data nodes (leaves) reside at the same tree level. We implement the algorithm in a recursive fashion, as shown in Algorithm 8.

---

**Algorithm 8** The prioritization algorithm

---

**Input:** The application root name prefix  $R$ , the named data set  $\mathcal{I}$  of the sender, the occupancy tree  $T$  of the receiver

**Output:** A prioritized order of object names

```

1: Return PRI( $R$ ,  $\mathcal{I}$ ,  $T$ )
2:
3: procedure PRI(name prefix  $P$ , data set  $\mathcal{I}$ , occupancy tree  $T$ )
4:   if  $P$  is leaf node then
5:     Return the corresponding data object  $\mathcal{I}.get(P)$ 
6:   end if
7:   order = [] ▷ Initiate an empty list of lists
8:   for Each branch  $b$  of  $P$  do
9:     order.append(PRI( $P/b$ ,  $\mathcal{I}$ ,  $T$ ))
10:  end for
11:  result = [] ▷ Initiate an empty list
12:  while order is not empty do:
13:     $S = \text{MINCHILD}(\text{order}, T)$ 
14:     $e = \text{LEASTOCCUPANCY}(S, T)$ 
15:    result.append( $e$ )
16:    UPDATEOCCUPANCY( $e$ ,  $T$ )
17:    if result[ $e.setIndex$ ] is empty then
18:      result.pop( $e.setIndex$ )
19:    end if
20:  end while
21:  Return result
22: end procedure

```

---

In this algorithm, the input parameters include the application root name prefix  $R$  under which all the application data resides in the name space, the data set at the sender side  $\mathcal{I}$ , and the occupancy tree  $T$  summarizing the name space occupancy at the receiver side. The algorithm calculates the prioritization order for each node at each level in the name tree in a bottom-up fashion. To compute the prioritization order of an inner node in the name tree, it “merges” the prioritization results of all its children nodes in a prioritized order. The merge process has three steps: (1) finding the data objects having the least common prefix with respect to the data set on the receiver side and assign highest priority to them (in the procedure `MINCHILD`), (2) balancing the occupancy tree at the receiver side by finding the data object residing at the name tree branch with least occupancy number (in the procedure `LEASTOCCUPANCY`), and (3) update the occupancy number of the occupancy tree  $T$  (by the procedure `UPDATEOCCUPANCY`). The return of Algorithm 8 is the prioritized order of the data objects at the sender side.

**Theorem 4.** *By marginal utility comparison rules in Section 7.1.3, Algorithm 8 returns the optimal prioritization order of the data objects at the sender side when data objects have the same size and weight.*

The proof of Theorem 4 is in the appendix. Please note that the optimality of Algorithm 8 holds *without assumption on the number of data objects transmitted in one transmission session*. In other words, for *any*  $k$  data objects transmitted in one transmission session Algorithm 8 is optimal, where  $k$  is no greater than the total number of data objects at the sender side.

### 7.1.7 Variable Object Length and Differentiated Service

In the above discussion, we assumed that all objects have the same length. In general, objects in some parts of the tree might be longer than others. For example, one branch might contain images with high quality (*i.e.* high resolution), whereas another contains images with low quality. To balance data collection from different branches, rather than maintaining a collected occupancy count for different prefixes, we maintain the number of collected bytes. Hence, when an object is selected, the occupancy count of its ancestor nodes in the name space is incremented by its length, as opposed to by one in UPDATEOCCUPANCY (see Algorithm 8). The approach will balance the bytes collected instead of objects. Besides the difference in balancing occupancy compared with the uniform data size, in this variable data object size case, we also want to transmit the data objects with the highest marginal utility “density”, which means that in the MINCHILD procedure, if two data objects under the same name prefix have the same marginal utility (*i.e.* the same length of the common name prefixes), the one with smaller size will be selected to transmit first. By modifying Algorithm 8, we can guarantee that the output prioritization order of the procedure PRI is in decreasing order of the data marginal utility density. However, due to the occupancy balancing of Algorithm 8, the property of decreasing marginal utility density is not guaranteed of the final returned prioritization order as shown in Theorem 6 in the appendix.

Finally, we can offer some prefixes preference over others by specifying a transmission weight,  $w_p$  for each prefix  $p$ . Accordingly, the function UPDATEOCCUPANCY in Algorithm 8 updates the occupancy count by the total bytes transmitted *divided by the weight of the prefix*. Hence, prefixes with higher weights will grow their (weighted) occupancy count at a slower rate resulting in an

amount of received content that is proportional to the weight of the prefix.

### 7.1.8 System Design and Implementation

The system contains three layers as shown in Fig. 7.2; (1) the application layer, (2) the information funnel layer, and (3) the NDN layer. The application layer contains all the application specific tasks, such as sensing and naming the data objects. The information funnel layer is designed for the generalized application-independent information-maximizing transmission. This novel design of separating application specific tasks from application independent tasks greatly simplifies the application development for both the sensing application on mobile devices (the clients) and the data collection application running on the backend server. In the rest of this section, we first introduce the APIs provided by the information funnel layer to the applications on both the mobile devices and the backend server respectively. Then we present how the information funnel layer interacts with the NDN layer.

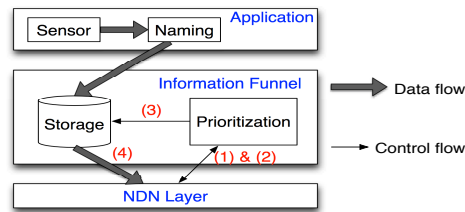


Figure 7.2: Information funnel structure

For mobile sensing applications, the funnel layer provides three APIs. The first one is `CREATEFUNNELSOURCE()` to start the *client funnel thread* for the information-maximizing data transmission. This API takes two parameters, the name prefix of the funnel and the device ID. The client funnel thread first allocates the *funnel repo* to cache the data objects under the name prefix of the funnel, and it actively probes the WiFi connection status. Once the connection is built, the thread initiates the data transmission in the prioritized order, which will be discussed later. Another API is `PUTTOFUNNEL()` to put data objects to the funnel repo. Its parameters are the funnel prefix, the data name, and the data object pointer. This function checks the name of the data object and only put the data with the funnel name prefix to the repo. The third API is `RELEASEFUNNELSOURCE()` for removing the funnel and recycling the resources to the OS. It takes only one parameter which is the name prefix of the funnel.

For the backend server application, the funnel layer also provides three APIs. All the three APIs only take one parameter, the name prefix of the funnel. The first API is `CREATEFUNNELSINK()` to allocate a local repo and start a thread called *server funnel thread* with a name prefix, the second one is `EXTRACTFROMFUNNEL()` to extract data objects from repo to the above server application, and the third one is `RELEASEFUNNELSINK()` to remove the funnel and recycle the OS resource.

After introducing the funnel APIs to the application layer, we present how the funnel interacts with the NDN layer for information-maximizing data transmission. The client funnel thread running on the mobile device actively probes the WiFi connection status. Once the WiFi connection is setup, it broadcasts an interest packet with the funnel name prefix. For example, `/ndn/uiuc/maps/[ID]/[timestamp]/summary`, where `ID` is the device ID and `timestamp` is the current local time. If the server funnel thread is created under the same name prefix, say `/ndn/uiuc/maps`, then the server responds with a data packet that contains the local occupancy tree of the name space (summarized to some level) as defined in Algorithm 8. Meanwhile, the sever funnel thread sends an interest packet with name `/ndn/uiuc/maps/[ID]/[new_timestamp]/list` to ask for the prioritized name list, where `ID` is the mobile device ID and `new_timestamp` is the server local time. After the client receives the occupancy summary of the server, it runs Algorithm 8 to prioritize the cached data objects in the repo and generate a name list. Upon receiving the interest packet `/ndn/uiuc/maps/[ID]/[new_timestamp]/list` from the server, the client funnel thread responds with a data packet of the name list. Then, the server fetches the data objects one-by-one according to the list.

Note that we add `timestamp` to the interests that are either sent by client asking for the server occupancy tree or sent by the server requesting the name list. The `timestamp` guarantees that those interests finally reach the end node rather than some intermediate cache. Although the in-net caching design of NDN accelerates the data transmission (for example, data dissemination from a content provider to content consumers), in our application we need those requests reach the end nodes because the state of either the server or the mobile device probably has already changed since last communication, thus the cached data probably be meaningless. However, the interests requesting data objects do not contain `timestamp`. With the assumption that the application will name different data objects differently, the funnel can use the NDN in-net caching to accelerate

the data transmission.

## 7.2 Evaluation

In this section, we study the performance of our algorithm to maximize the marginal information utility. We first introduce our methodology for the evaluation, and then evaluate the performance of the information funnel.

### 7.2.1 Methodology

We evaluate two aspects of the system: (1) the overhead of the prioritization, and (2) its performance by comparing with other state-of-the-art solutions. To measure the overhead in practical scenarios, we implement the information funnel on Google Galaxy Nexus phones [2]. Each phone is equipped with a 1.2 GHz dual-core CPU, 1GB RAM, and running Android OS 4.1. The information funnel is implemented using the Java programming language on top of the PARC’s CCNx prototype software [1]. The data set used in the evaluation is the T-Drive data set [155] collected by MSRA. We use the taxi traces in the urban area of Beijing, China, with GPS coordinates from latitude  $39.5^{\circ}N$  to  $40.5^{\circ}N$  and from longitude  $116^{\circ}E$  to  $117^{\circ}E$ , where most data points reside.

In the evaluation, we assume the social sensing application provides a hypothetical service called “city view everyday”, which is an improved version of the Google street view, where the user can see up-to-date street changes day-by-day as recorded by cameras in cars on street. This social sensing application needs to collect data objects (*i.e.* pictures) continuously from participants (*i.e.* cars).

To study the prioritization performance of the information funnel, we run a simulation on the T-Drive data set with assumptions that:

1. There are two WiFi sinks (gateways to one central server) to collect data that are located on two busy streets as shown in Fig 7.3(c),
2. The coverage range of each WiFi gateway is 100 meters,
3. The pictures are 100KB each,

4. The WiFi bandwidth is from 700Kbps to 1Mbps, which is estimated using the campus WiFi network, and
5. The speed of each cab is from 40km/h to 80km/h, which is estimated from the street speed limits of Beijing.

We simulate for 10 hours during which 50,000 data objects are collected by cabs (of which only 15% are uploaded to the server) and we assume that at the very beginning of the simulation the server does not have any data.

The area in the simulation is partitioned into 400 tiles, and each tile is further partitioned into 16 cells. The name of each data object (picture) is following the structure defined as `/citysense/tile_idx/cell_idx/filename`. So there are two possible levels of summary for the occupancy tree at the receiver (the central server) side.

We compare the performance of the Information Funnel with three baseline algorithms: (1) **FIFO**, which transmits the data objects in the fifo order of their time stamps, (2) **Distance-based** prioritization algorithm in PhotoNet [119] which always transmits the data object with the longest minimum distance from the data objects at the receiver side first, and (3) **Coverage-based** prioritization in Minerva [138], which always transmits the data object with the largest marginal coverage, where the side length of the coverage area of each data object defined to be 100 meters and we consider the information space is 2D. (Please refer to Minerva [138] for the detailed explanation of the configuration.) In the following section, we present the evaluation results of the information funnel, and we henceforth call the algorithm used in the information funnel as the “name-based” algorithm.

### 7.2.2 Evaluation Results

The computational overhead of data ordering results are shown in Table 7.1. The prioritization computation of the Information Funnel is on a Google Galaxy Nexus phone, because it is a client-side algorithm, while the computations of the distance-based and coverage-based algorithms are on a desktop with a 3.2GHz Intel i5 quad-core CPU, because they are designed to run on the data collection server. The average, maximum, and 80th percentile overheads are shown in the table, where the 80th percentile means that in 80% of the transmission sessions the computation

time is no more than this value. In the table, we also compare different levels of receiver feedback, denoted by  $X$  (as in Named-based( $X$ )), where  $X = 0$  means no feedback, and  $X = 1$  (*resp.*  $X = 2$ ) means the receiver summarizes the occupancy of the top one level (*resp.* two levels) of its name tree of all the local data objects under the funnel’s prefix. Note how the computational overhead introduced by data ordering in the named-based algorithm is much less than that introduced by distance-based and coverage-based algorithms. This is because the previous algorithms were quadratic in the number of items to prioritize, where ours is in the order of  $O(n \log n)$  (Theorem 5 in appendix). Considering that the WiFi connection time is around 2 seconds, the computational overhead introduced by the Information Funnel is negligible.

Table 7.1: Overhead study results

Algorithm	avg(ms)	max(ms)	80%(ms)
Name-based(0)	0.000	0.001	0.000
Name-based(1)	0.310	7.491	0.180
Name-based(2)	0.315	7.658	0.189
Distance-based	14.212	609.992	6.441
Coverage-based	14.418	369.931	7.286

Fig. 7.3 shows coverage performance of the algorithms, where we consider a cell of the map covered if at least one picture was uploaded from there. Please note that more uniform distribution of points in the figure implies a larger coverage. From Fig. 7.3, we clearly observe that FIFO is the worst algorithm, since the data collected by it covers the smallest area, and the distance-based and coverage-based algorithms performs better than FIFO, whereas our name-based algorithm is the best (*covers the largest area*).

Table 7.2: Coverage study results

Algorithm	tile cover.	cell cover.
Name-based(2)	100%	95.54%
Name-based(1)	100%	94.23%
FIFO	85.96%	68.90%
Distance-based	94.74%	78.74%
Coverage-based	98.25%	89.76%

The actual percentage of cells (and tiles) covered by the compared algorithms is shown in Table 7.2. Note that, our algorithm maximizes both metrics.

Fig. 7.3(f) illustrates the impact of our differentiated service extension, where we assign a higher



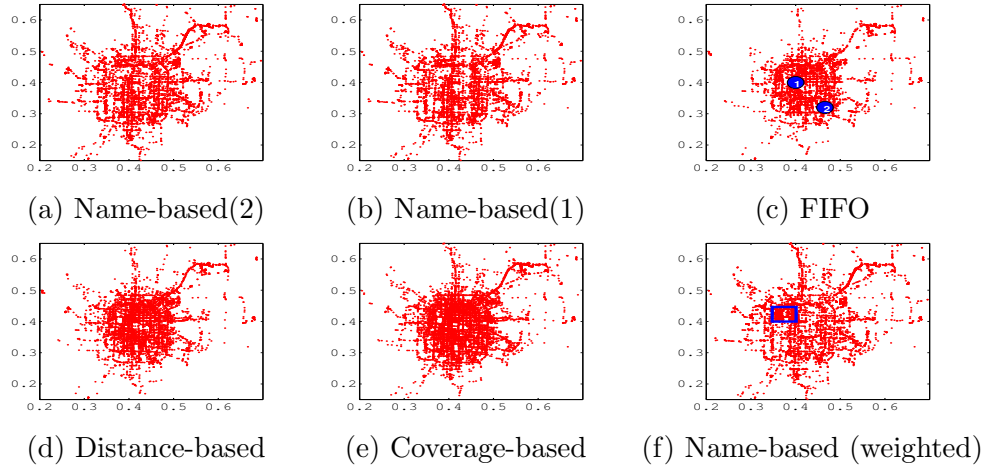


Figure 7.3: Performance of information funnel.

weight to the area  $[0.35, 0.4] \times [0.4, 0.45]$ , identified by the blue rectangle. Compared to Fig. 7.3(a) where the data has the same weight, we clearly observe that more data (more red) is collected within this rectangle than in Fig. 7.3(a).

Table 7.3: Coverage study with variable data size

Algorithm	tile cover.	cell cover.
Name-based(2)	100%	95.41%
Name-based(1)	100%	93.70%
FIFO	85.09%	68.50%
Distance-based	93.86%	78.22%
Coverage-based	97.37%	88.45%

Next, we study the coverage performance of the algorithms with variable data sizes. In this experiment, we randomly assign data objects sizes ranging from 100KB to 200KB. Please note that in our simulation, we first generate the random data size and then run the simulation, which guarantees that all the algorithms run on the same data set. The coverage performance is shown in Table 7.3. Note that, our algorithm maximizes both metrics.

The evaluation shows that named-data networking can be leveraged for efficient automatic coverage (and hence, information utility) maximization simply by giving data hierarchical names, where length of the common prefix grows with data similarity.

### 7.3 Related Work

Cyber-physical and sensing applications, where data objects are collected from the physical world, typically exhibit significant redundancy in collected data. This calls for prioritizing data collection in a way that reduces redundancy received. The problem was recently described by the authors in PhotoNet [119], where a picture-collection service was developed for disaster-response applications that maximize situation-awareness. Dron *et al.* [31] proposed a novel caching design aiming to maximize the information utility. Another illustration of such redundancy was covered in Car-Speak [72], where autonomous vehicles share cloud-point data for obstacle avoidance. In these papers, application-specific data prioritization policies were described that aim at improving an information utility metric.

Our work points out one aspect of application utility maximization (for data originating in the physical environment) that can be broadly generalized independently of other application details. Namely, within the scope of a query, received information is maximized when redundancy is minimized, since redundancy reduces information content. Hence, it makes sense to dedicate such information-maximization to a layer that is independent of and supports the specific application.

NDN enables the development of such a layer. One big benefit of NDN is that data has name, which makes the designing and implementing some kinds of applications easier, like the dataset synchronization [164]. Specifically, if data names can approximately denote similarity between objects, information maximizing data ordering (e.g., in data transmission across bottlenecks) can be done in a completely generic fashion by applying a breadth-first traversal algorithm to the subtree of an application's name space that falls within the scope of the query. This work focuses on persistent queries, where the relevant name-space subtree is expressed as a name prefix associated with the information funnel.

Our work is related to general efforts that attempt to handle network resource constraints in an efficient manner. Prior literature explores how to efficiently transfer (spatially, temporally or spatio-temporally) correlated data samples over multi-hop networks to a sink. Usually, one of two approaches is adopted: either compress samples to reduce redundancy or select a small subset of nodes to aggregate samples before sending them to the sink. A few notable examples of the first approach include Cristescu *et al.* [29], Pattem *et al.* [98] and Vuran *et al.* [122]. The sec-

ond approach includes schemes such as selecting an energy-efficient correlation-dominating set [50], clustering based on correlations [83], clustering based aggregation [87], routing through a set of mobile sinks [147], sensor data dissemination for mobile users [93] and distributed data collection by localized coding [156]. In [85], authors propose distance entropy as a metric to formalize communication cost for collecting correlated data. While the above work focused on time-series data, some more recent work [67, 119, 144] focuses on more complex data types (such as pictures). In contrast to the above literature, this work focuses on generic prioritization policies based on named-data networking.

Several approaches were considered for efficient data collection in intermittently connected networks and DTNs. These include interest profiles [39], cooperative sensing [161], vehicular data collection [132], publish/subscribe methods [78, 153], and subscription of channels [20, 76]. In [151], the authors studied caching, where nodes cache data based on popularity, such that future queries can be answered with less delay. Some research efforts [60, 99] improve data accessibility from infrastructure networks such as WiFi Access Points [60] or the Internet [99]. Our work is complementary in that we focus on maximizing information utility by minimizing redundancy in collected data. Specifically, we do so by designing an appropriate name space in the context of named-data networking.

## 7.4 Conclusions

In this paper, we introduced the information funnel, a data collection scheme that leverages the ability to name data (as in named-data networking) to offer information-maximizing content delivery for resource constrained social sensing applications. Our evaluation shows that our scheme increases the information coverage compared with the state-of-the-art solutions, while offering a very low overhead.

## 7.5 Math Proofs

**Lemma 3.** *Algorithm 8 always schedule the data objects with least common name prefix with respect to the data set on the receiver side plus the data set already scheduled on the sender side.*

*Proof.* In Algorithm 8, the procedure MINCHILD always returns the data objects with the shortest common name prefix with respect to the data objects at the receiver side in each iteration. The procedure UPDATEOCCUPANCY guarantees that the occupancy tree of the receiver side is updated by adding the data object currently is scheduled. Therefore, the lemma holds.  $\square$

**Lemma 4.** *When data objects to be scheduled share the same length of the common name prefix with respect to the data set on the receiver side plus the data set already scheduled on the sender side, Algorithm 8 always populates the name tree in a balanced fashion to schedule the data object whose name prefix has the least occupancy.*

*Proof.* This lemma is guaranteed to hold by the procedure LEASTOCCUPANCY in Algorithm 8. The input of this procedure is the output data set of MINCHILD such that data objects have the same length of common name prefix with respect to the data set on the receiver side union the data set already scheduled on the sender side. The output of this procedure is the data object residing at the least populated name prefix branch. Therefore, the lemma holds.  $\square$

*Theorem 1:* By marginal utility comparison rules in Section 7.1.3, Algorithm 8 returns the optimal prioritization order of the data objects at the sender side when data objects have the same size and weight. (This optimality holds *without assumption on the number of data objects transmitted in one transmission session*. In other words, for *any*  $k$  data objects transmitted in one transmission session Algorithm 8 is optimal, where  $k$  is no greater than the total number of data objects at the sender side.)

*Proof.* Since data objects are assumed to have the same size, in one transmission session, the number of data objects can be transmitted is the same for any prioritization. Let's denote this number  $n$ . Furthermore, the assumption that data objects have the same weight implies that the marginal information utility of the data objects only depends on the data names.

By Lemma 1, Lemma 2 and the marginal utility comparison rules in Section 7.1.3, we know that Algorithm 8 prioritizes the data objects in the non-increasing order of the marginal utilities. We claim that:

*Claim:* Given  $n$  numbers  $\{m_1, m_2, \dots, m_n\}$  and two permutations  $P$  and  $Q$ .  $P$  permutes those numbers in the non-increasing order and  $Q$  permutes in an arbitrary order. For any  $0 \leq k \leq n$ ,

$\sum_{i=P_0}^{P_k} m_i \geq \sum_{i=Q_0}^{Q_k} m_i$ , where  $P_i$  ( $Q_i$  resp.) means the  $i$ -th number based on the permutation  $P$  ( $Q$  resp.).

*Proof of Claim:* Suppose that  $\sum_{i=P_0}^{P_k} m_i < \sum_{i=Q_0}^{Q_k} m_i$ . Then there must exist some number  $m_t$  that in the first  $k$  elements by the  $Q$  permutation but not in that by the  $P$  permutation and  $m_t > m_{P_j}$  for some  $0 \leq j \leq k$ . Therefore, we got a contradiction with that  $P$  permutes the numbers in the non-increasing order. Therefore,  $\sum_{i=P_0}^{P_k} m_i \geq \sum_{i=Q_0}^{Q_k} m_i$ .

The above claim actually proves that our prioritization is optimal, since it permutes data objects in the non-increasing order of information utility. And the optimality is guaranteed for any number  $k$  of data objects transmitted in one transmission session, where  $k$  is no greater than the total number of data objects at the sender side by the above claim.  $\square$

**Theorem 5.** *The time complexity of Algorithm 8 is  $O(n \log n + H^2 n)$ , where  $H$  is the height of the name tree composed of the names of data objects to be transmitted. When  $H = O(1)$ , the time complexity is  $O(n \log n)$ , the best complexity bound for sorting based on comparison.*

*Proof.* The time complexity of MINCHILD and LEASTOCCUPANCY is  $O(\log(\Delta))$ , where  $\Delta$  is the maximum number of children of every node in the name tree, i.e.,  $\Delta := \max_{v \in \text{tree}} |v.\text{child}|$ . The time complexity of UPDATEOCCUPANCY is  $O(H)$ , where  $H$  is the height of the name tree. For each level in the name tree, the time complexity of the while loop is  $O(n(\log \Delta + H))$ , thus the total time complexity of Algorithm 8 is  $O(H \cdot n(\log \Delta + H)) = O(n \log n + H^2 n)$ , since  $n = O(\Delta^H)$ , which completes the proof.  $\square$

Consider the fact that  $H \ll n$  in practical, we have the time complexity of Algorithm 8 is  $O(n \log^2 n)$  if  $H = O(\log n)$ , or  $O(n \log n)$  if  $H = O(1)$ , which means the time complexity of our algorithm is almost the same as the sorting algorithm.

**Theorem 6.** *Compared with the optimal offline algorithm satisfying the hierarchical similarity property and diminishing return property, in one transmission session, the approximation ratio of Algorithm 8 in the general case is  $\frac{N}{N+\delta}$ , where  $N$  is the total number of packets transmitted in the transmission session by Algorithm 8 and  $\delta = \lfloor \frac{L_{\max}}{L_{\min}} \rfloor$ ,  $L_{\max}$  (resp.  $L_{\min}$ ) is the size of the largest (resp. smallest) data object.*

*Proof.* The optimality requires the two properties must be satisfied, which means the occupancy should be well balanced in the name tree. By Theorem 4, Algorithm 8 balances the occupancy in the name tree optimally in the uniform data size case, which can be generalized straightforward in the variable data size case. Thus, all the data objects transmitted by using Algorithm 8 should be transmitted using the optimal algorithm. Otherwise, the occupancy balancing is violated.

In the worst case, using Algorithm 8 we can waste  $L_{\max} - \epsilon$  transmittable bytes by scheduling a largest data object in the end of the transmission session. However, we can use those bytes to transmit several small data objects. So the number of packets transmitted by the optimal algorithm is at most  $N + \lfloor L_{\max}/L_{\min} \rfloor = N + \delta$ , which proves the theorem.  $\square$

# Chapter 8

## Conclusion and Future Work

### 8.1 Summary

The thesis proposes an information filtering framework for social sensing on text-based data that delivers the most informative data to the users in a consumable volume. Due to the high costs of data labeling, we believe the supervised approach that requires huge amount of labels would be non-scalable for the massive data of social sensing, therefore, we chose unsupervised approaches. Furthermore, we do not intend to confine the system only being able to process texts of some specific language but rather design a general system applicable to a wide range of languages, therefore our framework is language-agnostic that is it does not exploit language semantics. With the above objectives, we design a three-level information filtering framework, as follows.

#### 8.1.1 Untruthful Information Removal Module

The first module targets on untruthful information removal, a practice also known as fact-finding. We extend the previous fact-finding approaches by considering two constraints that widely hold in certain types of social sensing applications, especially in human-in-the-loop Cyber-Physical Systems. The first constraint is time-varying system state that the ground truth of the system state (represented by the binary values of all variables in the system) is not static which the previous work assumed. The second constraint is inter-dependent variables where, unlike previous work that assumed the variables are independent, we assume their dependency can be modeled as a Bayesian network. For both constraints, we propose Expectation-Maximization algorithms respectively that simultaneously learn the reliability of individual human sensors and the binary value of each variable (i.e. true or false), and demonstrate the efficiency of the proposed solutions with real-world data evaluations.

### 8.1.2 Event-level Information Summarization

Instead of feeding social sensing users with individual text messages, it makes more sense to cluster them for each physical event. Therefore, our information filtering system requires a summarization module that can automatically cluster individual text messages into event-based clusters, a practice also known as event detection. Being different from prior work, ours focuses on de-multiplexing event instances. We propose an unsupervised and time-window-based approach that extracts a bunch of potential signatures of different event instances based on information gain, and cluster the text messages with the detected signatures, then it consolidates the clusters of the same physical event on a normal text distance metric. Experiments with real data crawled from Twitter verifies that our work performs better in event de-multiplexing compared with the state-of-the-art. We also extend our consolidation algorithm with a sliding-window for tracking one event instance across time, and show the efficiency of event tracking on a bunch of case studies.

### 8.1.3 Information-Maximizing Delivery

The third filtering module applies when we are ready to feed the event-level summaries to the users. Here we define information maximization as redundancy minimization, with the rationale that the users would not like to read redundant or similar feeds. Uddin et al. [120] proposed a distance-based diversifying scheme that always prioritizes the data item with largest distance of the delivered items in some information space, which we can also apply in our system. However, the proposed approach, although handy to use, is merely heuristic and does not have any analytical performance guarantee. We propose two prioritization schemes that under some assumptions are theoretically optimal.

## 8.2 Future Research Directions

One direction would be social noise removal. Besides physical events, a large body of the social network posts are about people's opinions, such as expressing one's happiness/sadness or showing the viewpoints about something, etc. Those opinion posts are not of interest in social sensing because we only care about physical events here. We term those opinion posts in social networks as



the *social noises*. Intuitively, the social noises are all from some individual's mind but not observable by others before posting whereas the physical events are observable by multiple individual people at the same time. Therefore, in hypothesis, there would be some underlying post-repost structural differences between the social noises and physical event posts, for example, the post-repost structure is tree-like for social noises whereas forest-like for physical event posts. A good future work is to explore this structural difference and propose some unsupervised solution to remove social noises.

Another direction would be filtering for other types of posts in social media, like photos and videos. The thesis only filters text posts, while leaves other types for future work. Deep learning might be a good tool here for photo and video content abstraction or description. One potential solution is to apply deep learning technique to describe the contents in photos/videos in texts, then apply the system framework proposed in the thesis for information filtering.

# References

- [1] CCNx prototype software. <http://www.ccnx.org>.
- [2] Google galaxy nexus. <http://www.google.com/nexus>.
- [3] Instagram statistics. <http://expandedramblings.com/index.php/important-instagram-stats/>.
- [4] Twitter inc. <http://www.twitter.com>.
- [5] Twitter statistics. <https://www.omnicoreagency.com/twitter-statistics/>.
- [6] User guide of t-drive data. [http://research.microsoft.com/pubs/152883/User\\_guide\\_T-drive.pdf](http://research.microsoft.com/pubs/152883/User_guide_T-drive.pdf).
- [7] T. Abdelzaher, Y. Anokwa, P. Boda, J. A. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *Center for Embedded Network Sensing*, 2007.
- [8] C. C. Aggarwal. A survey of stream clustering algorithms., 2013.
- [9] C. C. Aggarwal and K. Subbian. Event detection in social streams. In *SDM*, volume 12, pages 624–635. SIAM, 2012.
- [10] A. Agresti. *An introduction to categorical data analysis*, volume 135. Wiley New York, 1996.
- [11] L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Goker, I. Kompatsiaris, and A. Jaimes. Sensing trending topics in twitter. *Multimedia, IEEE Transactions on*, 15(6):1268–1282, 2013.
- [12] All hazards consortium. <http://www.ahcusa.org/>.
- [13] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR*. ACM, 1998.
- [14] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker. Packet caches on routers: the implications of universal redundant traffic elimination. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 219–230. ACM, 2008.
- [15] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *SenSys*. ACM, 2003.
- [16] F. Atefeh and W. Khreich. A survey of techniques for event detection in twitter. *Comput. Intell.*, 31(1):132–164, Feb. 2015.

- [17] J. Benhardus and J. Kalita. Streaming trend detection in twitter. *International Journal of Web Based Communities*, 9(1):122–139, 2013.
- [18] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
- [19] A. Boettcher and D. Lee. Eventradar: A real-time local event detection scheme using twitter stream. In *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications, GREENCOM '12*, pages 358–367, Washington, DC, USA, 2012. IEEE Computer Society.
- [20] C. Boldrini, M. Conti, and A. Passarella. ContentPlace: social-aware data dissemination in opportunistic networks. In *Proc. of MSWiM*, 2008.
- [21] M. Brenner and E. Izquierdo. Social event detection and retrieval in collaborative photo collections. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 21. ACM, 2012.
- [22] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. Srivastava. Participatory sensing. 2006.
- [23] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. 2006.
- [24] G. Casella and R. L. Berger. *Statistical inference*, volume 70. Duxbury Press Belmont, CA, 1990.
- [25] D. M. Chickering. Learning bayesian networks is np-complete. In *Learning from data*, pages 121–130. Springer, 1996.
- [26] F. Chierichetti, J. M. Kleinberg, R. Kumar, M. Mahdian, and S. Pandey. Event detection via communication pattern analysis. In *ICWSM*, 2014.
- [27] E. M. Clarke, B. Krogh, A. Platzer, and R. Rajkumar. Analysis and verification challenges for cyber-physical transportation systems. In *National Workshop for Research on High-confidence Transportation Cyber-Physical Systems: Automotive, Aviation & Rail*, 2008.
- [28] H. Cramér. *Mathematical methods of statistics*, volume 9. Princeton university press, 1999.
- [29] R. Cristescu, B. Beferull-Lozano, M. Vetterli, and R. Wattenhofer. Network correlated data gathering with explicit communication: Np-completeness and algorithms. *IEEE/ACM Trans. Netw.*, 14:41–54, 2006.
- [30] A. P. Dempster, N. M. Laird, D. B. Rubin, et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.
- [31] W. Dron, A. Leung, M. Uddin, S. Wang, T. Abdelzaher, R. Govindan, and J. Hancock. Information-maximizing caching in ad hoc networks with named data networking. In *Network Science Workshop (NSW), 2013 IEEE 2nd*, pages 90–93. IEEE, 2013.
- [32] J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou. Distributed real-time software for cyber-physical systems. *Proceedings of the IEEE*, 100(1):45–59, 2012.

- [33] S. Eisenman, E. Miluzzo, N. Lane, R. Peterson, G. Ahn, and A. Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):6, 2009.
- [34] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):6, 2009.
- [35] A. Faza, S. Sedigh, and B. McMillin. Integrated cyber-physical fault injection for reliability analysis of the smart grid. In *Computer Safety, Reliability, and Security*, pages 277–290. Springer, 2010.
- [36] U. Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
- [37] R. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. Abdelzaher. Greengps: A participatory sensing fuel-efficient maps application. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 151–164. ACM, 2010.
- [38] R. Ganti, N. Pham, Y. Tsai, and T. Abdelzaher. Poolview: stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 281–294. ACM, 2008.
- [39] W. Gao and G. Cao. User-centric data dissemination in disruption tolerant networks. In *Proc. of IEEE Infocom*, 2011.
- [40] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *Proc of SIGCOMM Workshop on ICN*, 2011.
- [41] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.
- [42] P. Giridhar, T. Abdelzaher, and L. Kaplan. Social fusion: Integrating twitter and instagram for event monitoring. In *UIUC tech report*, 2017.
- [43] P. Giridhar, M. T. Amin, T. Abdelzaher, L. Kaplan, J. George, and R. Ganti. Clarisense: Clarifying sensor anomalies using social network feeds. In *PERCOM Workshops*. IEEE, 2014.
- [44] P. Giridhar, S. Wang, T. Abdelzaher, R. Ganti, L. Kaplan, and J. George. On localizing urban events with instagram. In *IEEE Infocom, Atlanta, GA, May 2017*, 2017.
- [45] P. Giridhar, S. Wang, T. F. Abdelzaher, J. George, L. Kaplan, and R. Ganti. Joint localization of events and sources in social networks. In *Proceedings of the 2015 International Conference on Distributed Computing in Sensor Systems, DCOSS '15*, pages 179–188, Washington, DC, USA, 2015. IEEE Computer Society.
- [46] A. Goswami and A. Kumar. A survey of event detection techniques in online social networks. *Social Network Analysis and Mining*, 6(1):107, 2016.

- [47] S. Gu, C. Pan, H. Liu, S. Li, S. Hu, L. Su, S. Wang, D. Wang, T. Amin, R. Govindan, G. Agarwal, R. Ganti, M. Srivatsa, A. Barnoy, P. Terlecky, and T. Abdelzaher. Data extrapolation in social sensing for disaster response. In *Proceedings of the 10th IEEE International Conference on Distributed Computing in Sensor Systems*. IEEE Press, 2014.
- [48] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *MobiCom*. ACM, 2004.
- [49] A. Guille and C. Favre. Mention-anomaly-based event detection and tracking in twitter. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 375–382. IEEE, 2014.
- [50] H. Gupta, V. Navda, S. Das, and V. Chowdhary. Efficient gathering of correlated data in sensor networks. In *Proc. of MobiHoc*, 2005.
- [51] N. Hochman and L. Manovich. Zooming into an instagram city: Reading the local through social media. *First Monday*, 18(7), 2013.
- [52] P. Houdyer, A. Zimmerman, M. Kaytoue, M. Plantevit, J. Mitchell, and C. Robardet. Gazouille: Detecting and illustrating local events from geolocalized social media streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 276–280. Springer, 2015.
- [53] Y. Hu, A. John, D. D. Seligmann, and F. Wang. What were the tweets about? topical associations between public events and twitter feeds. In *ICWSM*, 2012.
- [54] Y. Hu, L. Manikonda, S. Kambhampati, et al. What we instagram: A first analysis of instagram photo content and user types. *Proceedings of ICWSM. AAAI*, 2014.
- [55] T. Hua, F. Chen, L. Zhao, C.-T. Lu, and N. Ramakrishnan. Automatic targeted-domain spatiotemporal event detection in twitter. *Geoinformatica*, 20(4):765–795, Oct. 2016.
- [56] J. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 180–191. ACM, 2005.
- [57] J. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 180–191. ACM, 2005.
- [58] J.-H. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *SenSys*, 2005.
- [59] M. Huang, J. Li, X. Song, and H. Guo. Modeling impulsive injections of insulin: Towards artificial pancreas. *SIAM Journal on Applied Mathematics*, 72(5):1524–1548, 2012.
- [60] Y. Huang, Y. Gao, K. Nahrstedt, and W. He. Optimizing file retrieval in delay-tolerant content distribution community. In *Proc. of ICDCS*, pages 308–316, 2009.
- [61] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *SenSys*, 2006.

- [62] P. Ishwar, R. Puri, K. Ramchandran, and S. S. Pradhan. On rate-constrained distributed estimation in unreliable sensor networks. *Selected Areas in Communications, IEEE Journal on*, 23(4):765–775, 2005.
- [63] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.
- [64] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proc. of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, New York, NY, USA, 2009. ACM.
- [65] M. Jäger, C. Knoll, F. Hamprecht, et al. Weakly supervised learning of a classifier for unusual event detection. *Image Processing, IEEE Transactions on*, 17(9):1700–1708, 2008.
- [66] R. I. Jennrich. An asymptotic  $\chi^2$  test for the equality of two correlation matrices. *Journal of the American Statistical Association*, 65(330):904–912, 1970.
- [67] Y. Jiang, X. Xu, P. Terlecky, T. Abdelzaher, A. Bar-Noy, and R. Govindan. Mediascope: selective on-demand media retrieval from mobile devices. In *Proceedings of the 12th international conference on Information processing in sensor networks*, pages 289–300. ACM, 2013.
- [68] A. R. Jonckheere. A distribution-free k-sample test against ordered alternatives. *Biometrika*, pages 133–145, 1954.
- [69] Kevin Murphy. Bayes Net Toolbox for Matlab. <https://code.google.com/p/bnt/>.
- [70] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.
- [71] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [72] S. Kumar, L. Shi, N. Ahmed, S. Gil, D. Katabi, and D. Rus. Carspeak: a content-centric network for autonomous driving. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '12, pages 259–270, New York, NY, USA, 2012. ACM.
- [73] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo, and A. T. Campbell. Urban sensing systems: opportunistic or participatory? In *HotMobile*, 2008.
- [74] J. H. Lau, N. Collier, and T. Baldwin. On-line trend analysis with topic models: \# twitter trends detection topic model online. In *COLING*, pages 1519–1534, 2012.
- [75] E. A. Lee. Cyber physical systems: Design challenges. In *ISORC*, 2008.
- [76] V. Lenders, G. Karlsson, and M. May. Wireless ad hoc podcasting. In *Proc. of IEEE SECON*, pages 273–283, 2007.
- [77] C. Li, A. Sun, and A. Datta. Twevent: segment-based event detection from tweets. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 155–164. ACM, 2012.

- [78] F. Li and J. Wu. MOPS: Providing content-based service in disruption-tolerant networks. In *Proc. of ICDCS*, 2009.
- [79] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198. ACM, 2014.
- [80] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang. Tedas: A twitter-based event detection and analysis system. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, ICDE '12*, pages 1273–1276, Washington, DC, USA, 2012. IEEE Computer Society.
- [81] Y. Liang, J. Caverlee, and C. Cao. A noise-filtering approach for spatio-temporal event detection in social media. In *Advances in Information Retrieval*, pages 233–244. Springer, 2015.
- [82] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE TMC*, 5(8):1044–1056, 2006.
- [83] C. Liu, K. Wu, , and J. Pei. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Trans. Parallel Distrib. Syst.*, 18:1011–1023, 2007.
- [84] H. Liu, A. Srinivasan, K. Whitehouse, and J. Stankovic. Mélange: Supporting heterogeneous qos requirements in delay tolerant sensor networks. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pages 93–96. IEEE, 2010.
- [85] J. Liu, M. Adler, D. Towsley, and C. Zhang. On optimal communication cost for gathering correlated data through wireless sensor networks. In *Proc. of MobiCom*, 2006.
- [86] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The smart thermostat: using occupancy sensors to save energy in homes. In *SenSys*, 2010.
- [87] Y. Ma, Y. Guo, X. Tian, and M. Ghanem. Distributed clustering-based aggregation algorithm for spatial correlated sensor networks. *Sensors Journal, IEEE*, 11(3):641–648, 2011.
- [88] Mark Paskin. A short course on graphical models. <http://ai.stanford.edu/paskin/gm-short-course/>.
- [89] E. Masazade, R. Niu, P. K. Varshney, and M. Keskinöz. A probabilistic transmission scheme for distributed estimation in wireless sensor networks. In *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, pages 1–6. IEEE, 2010.
- [90] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe. Parknet: drive-by sensing of road-side parking statistics. In *MobiSys*, 2010.
- [91] M. Meisel, V. Pappas, and L. Zhang. Ad hoc networking via named data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, pages 3–8. ACM, 2010.
- [92] S. Nawaz, C. Efstratiou, and C. Mascolo. Parksense: a smartphone based sensing system for on-street parking. In *MobiCom*, 2013.

- [93] E. Ngai, M. B. Srivastava, and J. Liu. Context-aware sensor data dissemination for mobile users in remote areas. In *INFOCOM, 2012 Proceedings IEEE*, pages 2711–2715. IEEE, 2012.
- [94] T. D. Nielsen and F. V. Jensen. *Bayesian networks and decision graphs*. Springer, 2009.
- [95] C. Ordonez. Clustering binary data streams with k-means. In *ACM SIGMOD workshop*. ACM, 2003.
- [96] R. Parikh and K. Karlapalem. Et: events from tweets. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 613–620. International World Wide Web Conferences Steering Committee, 2013.
- [97] J. Pasternack and D. Roth. Knowing what to believe (when you already know something). In *COLING*, 2010.
- [98] S. Patten, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Trans. Sensor Networks*, 4:24–33, 2008.
- [99] M. J. Pitkanen and J. Ott. Redundancy and distributed caching in mobile DTNs. In *MobiArch*, 2007.
- [100] C. S. Raghavendra, K. M. Sivalingam, and T. Znati. *Wireless sensor networks*. Springer, 2004.
- [101] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference (DAC)*, pages 731–736. ACM, 2010.
- [102] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: Real-time event detection by social sensors. In *WWW*, 2010.
- [103] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95(1):163–187, 2007.
- [104] G. Schirner, D. Erdogmus, K. Chowdhury, and T. Padir. The future of human-in-the-loop cyber-physical systems. *Computer*, 46(1):36–45, 2013.
- [105] Sense Networks. Cab Sense. <http://www.cabsense.com>.
- [106] L. Sha and J. Meseguer. Design of complex cyber physical systems with formalized architectural patterns. In *Software-Intensive Systems and New Computing Paradigms*, pages 92–100. Springer, 2008.
- [107] D. A. Shamma, L. Kennedy, and E. F. Churchill. Peaks and persistence: modeling the shape of microblog conversations. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 355–358. ACM, 2011.
- [108] T. Silva, P. de Melo, J. Almeida, J. Salles, and A. Loureiro. A picture of instagram is worth more than a thousand words: Workload characterization and application. In *2013 IEEE DCOSS*, pages 123–132, May 2013.
- [109] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry. Kalman filtering with intermittent observations. *Automatic Control, IEEE Transactions on*, 49(9):1453–1464, 2004.



- [110] J. A. Stankovic, I. Lee, A. Mok, and R. Rajkumar. Opportunities and obligations for physical computing systems. *Computer*, 38(11):23–31, 2005.
- [111] L. Su, J. Gao, Y. Yang, T. F. Abdelzaher, B. Ding, and J. Han. Hierarchical aggregate classification with limited supervision for data reduction in wireless sensor networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 40–53. ACM, 2011.
- [112] L. Su, S. Hu, S. Li, F. Liang, J. Gao, T. F. Abdelzaher, and J. Han. Quality of information based data selection and transmission in wireless sensor networks. In *RTSS*, pages 327–338, 2012.
- [113] L. Su, Q. Li, S. Hu, S. Wang, J. Gao, H. Liu, T. Abdelzaher, J. Han, X. Liu, Y. Gao, and L. Kaplan. Generalized decision aggregation in distributed sensing systems. In *Real-Time Systems Symposium (RTSS), 2014 IEEE 35th*. IEEE, 2014.
- [114] L. Su, Q. Li, S. Hu, S. Wang, J. Gao, H. Liu, T. Abdelzaher, J. Han, X. Liu, Y. Gao, and L. Kaplan. Generalized decision aggregation in distributed sensing systems. In *Real-Time Systems Symposium (RTSS)*, 2014.
- [115] I. Tien, A. Musaev, D. Benas, and C. Pu. Detection of damage and failure events of critical public infrastructure using social sensor big data. In *Proceedings of International Conference on Internet of Things and Big Data*, April 2016.
- [116] I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.
- [117] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, and A. Mauthe. A survey of mobility in information-centric networks: challenges and research directions. In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*, pages 1–6. ACM, 2012.
- [118] M. Uddin, H. Wang, F. Saremi, G. Qi, T. Abdelzaher, and T. Huang. Photonet: A similarity-aware picture delivery service for situation awareness. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pages 317–326. IEEE, 2011.
- [119] M. Uddin, H. Wang, F. Saremi, G.-J. Qi, T. Abdelzaher, and T. Huang. PhotoNet: a similarity-aware picture delivery service for situation awareness. In *Proc. of IEEE RTSS*, 2011.
- [120] M. Y. S. Uddin, H. Wang, F. Saremi, G.-J. Qi, T. Abdelzaher, and T. Huang. Photonet: a similarity-aware picture delivery service for situation awareness. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pages 317–326. IEEE, 2011.
- [121] K. N. Vavliakis, A. L. Symeonidis, and P. A. Mitkas. Event identification in web social media through named entity recognition and topic modeling. *Data & Knowledge Engineering*, 88:1–24, 2013.
- [122] M. Vuran, O. Akan, and I. Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45:245–259, 2004.

- [123] M. Walther and M. Kaisser. Geo-spatial event detection in the twitter stream. In *Proceedings of the 35th European Conference on Advances in Information Retrieval, ECIR'13*, pages 356–367, Berlin, Heidelberg, 2013. Springer-Verlag.
- [124] D. Wang, T. Abdelzaher, H. Ahmadi, J. Pasternack, D. Roth, M. Gupta, J. Han, O. Fatemieh, H. Le, and C. C. Aggarwal. On bayesian interpretation of fact-finding in information networks. In *FUSION*, 2011.
- [125] D. Wang, T. Abdelzaher, L. Kaplan, and C. C. Aggarwal. Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications. In *ICDCS*, 2013.
- [126] D. Wang, T. Abdelzaher, L. Kaplan, R. Ganti, S. Hu, and H. Liu. Exploitation of physical constraints for reliable social sensing. In *RTSS*, 2013.
- [127] D. Wang, T. Amin, S. Li, T. A. L. Kaplan, S. G. C. Pan, H. Liu, C. Aggrawal, R. Ganti, X. Wang, P. Mohapatra, B. Szymanski, and H. Le. Humans as sensors: An estimation theoretic perspective. In *IPSN*, 2014.
- [128] D. Wang, L. Kaplan, T. Abdelzaher, and C. C. Aggarwal. On scalability and robustness limitations of real and asymptotic confidence bounds in social sensing. In *SECON*, 2012.
- [129] D. Wang, L. Kaplan, and T. F. Abdelzaher. Maximum likelihood analysis of conflicting observations in social sensing. *ACM Transactions on Sensor Networks (TOSN)*, 10(2):30, 2014.
- [130] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. On truth discovery in social sensing: a maximum likelihood estimation approach. In *IPSN*, pages 233–244. ACM, 2012.
- [131] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. On truth discovery in social sensing: a maximum likelihood estimation approach. In *IPSN*, 2012.
- [132] J. Wang, R. Wakikawa, and L. Zhang. Dmnd: Collecting data from mobiles using named data. In *Vehicular Networking Conference (VNC), 2010 IEEE*, pages 49–56. IEEE, 2010.
- [133] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang. Data naming in vehicle-to-vehicle communications.
- [134] S. Wang, T. Abdelzaher, S. Gajendran, A. Herga, S. Kulkarni, S. Li, H. Liu, C. Suresh, A. Sreenath, W. Dron, et al. Poster abstract: information-maximizing data collection in social sensing using named-data. In *Proceedings of the 13th international symposium on Information processing in sensor networks*, pages 303–304. IEEE Press, 2014.
- [135] S. Wang, T. Abdelzaher, S. Gajendran, A. Herga, S. Kulkarni, S. Li, H. Liu, C. Suresh, A. Sreenath, H. Wang, W. Dron, A. Leung, R. Govindan, and J. Hancock. The information funnel: Exploiting named data for information-maximizing data collection. In *Proceedings of the 10th IEEE International Conference on Distributed Computing in Sensor Systems*. IEEE Press, 2014.
- [136] S. Wang, P. Giridhar, L. Kaplan, and T. Abdelzaher. Unsupervised event tracking by integrating twitter and instagram. In *Proceedings of the 2nd International Workshop on Social Sensing*. ACM, 2017.

- [137] S. Wang, P. Giridhar, H. Wang, L. Kaplan, T. Pham, A. Yener, and T. Abdelzaher. Storyline: On physical event demultiplexing and tracking in social spaces. In *IoTDI*, 2017.
- [138] S. Wang, S. Hu, S. Li, H. Liu, M. Uddin, and T. Abdelzaher. Minerva: Information-centric programming for social sensing. In *Proc. of IEEE ICCCN*, 2013.
- [139] S. Wang, S. Hu, S. Li, H. Liu, M. Y. S. Uddin, and T. Abdelzaher. Minerva: Information-centric programming for social sensing. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pages 1–9. IEEE, 2013.
- [140] S. Wang, L. Su, S. Li, S. Hu, T. Amin, H. Wang, S. Yao, L. Kaplan, and T. Abdelzaher. Scalable social sensing of interdependent phenomena. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 202–213. ACM, 2015.
- [141] S. Wang, D. Wang, L. Su, L. Kaplan, and T. Abdelzaher. Towards cyber-physical systems in social spaces: The data reliability challenge. In *Real-Time Systems Symposium (RTSS)*, 2014.
- [142] K. Watanabe, M. Ochi, M. Okabe, and R. Onai. Jasmine: A real-time local-event detection system based on geolocation information propagated to microblogs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11*, pages 2541–2544, New York, NY, USA, 2011. ACM.
- [143] U. Weinsberg, A. Balachandran, N. Taft, G. Iannaccone, V. Sekar, and S. Seshan. Care: Content aware redundancy elimination for disaster communications on damaged networks. *Arxiv preprint arXiv:1206.1815*, 2012.
- [144] U. Weinsberg, Q. Li, N. Taft, A. Balachandran, V. Sekar, G. Iannaccone, and S. Seshan. Care: content aware redundancy elimination for challenged networks. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 127–132. ACM, 2012.
- [145] J. Weng and B.-S. Lee. Event detection in twitter. *ICWSM*, 11:401–408, 2011.
- [146] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy. On the scale and performance of cooperative web proxy caching. *ACM SIGOPS Operating Systems Review*, 33(5):16–31, 1999.
- [147] X. Xu, J. Luo, and Q. Zhang. Delay tolerant event collection in sensor networks with mobile sink. In *Proc. of INFOCOM*, 2010.
- [148] D. Yang, G. Xue, X. Fang, and J. Tang. Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking (MobiCom)*, pages 173–184. ACM, 2012.
- [149] Y. Yang, J. G. Carbonell, R. D. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, (4):32–43, 1999.
- [150] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod Record*, 31(3):9–18, 2002.
- [151] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *IEEE Trans. on Mobile Computing*, 5, 2011.

- [152] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *Knowledge and Data Engineering, IEEE Transactions on*, 20(6):796–808, 2008.
- [153] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *Proc. of MSWiM*, 2007.
- [154] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2011.
- [155] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108. ACM, 2010.
- [156] K. Yuan, B. Li, and B. Liang. A distributed framework for correlated data gathering in sensor networks. *IEEE Trans. Veh. Technol.*, 57:578–593, 2008.
- [157] C. Zhai and S. Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Association for Computing Machinery and Morgan & #38; Claypool, New York, NY, USA, 2016.
- [158] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. M. Kaplan, S. Wang, and J. Han. Geoburst: Real-time local event detection in geo-tagged tweet streams. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 513–522, 2016.
- [159] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Semi-supervised adapted hmms for unusual event detection. In *IEEE CVPR*. IEEE, 2005.
- [160] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. Technical report, PARC, Tech. report ndn-0001, 2010.
- [161] D. Zhao, H. Ma, and S. Tang. Coupon: Cooperatively building sensing maps in mobile opportunistic networks. In *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pages 295–303. IEEE, 2013.
- [162] S. Zhong. Efficient streaming text clustering. *Neural Networks*, 18(5):790–798, 2005.
- [163] X. Zhou and L. Chen. Event detection over twitter social media streams. *The VLDB journal*, 23(3):381–400, 2014.
- [164] Z. Zhu and A. Afanasyev. Let’s chronosync: Decentralized dataset state synchronization in named data networking. In *Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP 2013)*, 2013.