TRACKING PHYSICAL EVENTS ON SOCIAL MEDIA

BY

PRASANNA GIRIDHAR

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, 2018

Urbana, Illinois

Doctoral Committee:

Professor Tarek Abdelzaher, Chair
Professor Jiawei Han
Associate Professor Hari Sundaram
Dr. Lance Kaplan, U.S. Army Research Laboratory

ABSTRACT

Social media platforms have emerged as the widely accessed form of communication channel on the world wide web in the modern day. The first ever social networking website came into existence in the year 2002 and currently there are about 2.08 billion social media users around the globe. The participation of users within a social network can be considered as an act of sensing where they are interacting with the physical world and recording the corresponding observations in the form of texts, pictures, videos, etc. This phenomenon is termed as *Social Sensing* and motivates us to develop robust techniques which can estimate the physical state from the human observations.

This dissertation addresses a set of problems related to detection and tracking of real-world events. The term 'event' refers to an entity that can be characterized by spatial and temporal properties. With the help of these properties we design novel mathematical models that help us with our goals. We first focus on a simple event detection technique using 'Twitter' as the source of information. The method described in this work allow us to perform detection in a completely language independent and unsupervised fashion. We next extend the event detection problem to a different type of social media, 'Instagram', which allows users to share pictorial information of nearby observations. With the availability of geotagged data we solve two different subproblems - the first one is to detect and geolocalize the instance of an event and the second one is to estimate the path taken by an event during its course. The next problem we look at is related to improving the quality of event localization with the help of text and metadata information. Twitter, in general, has less volume of geotagged data available in comparison to Instagram, which demands us to design methods that explore the supplementary information available from the detected events. Finally, we take a look at both the social networks at the same time in order to utilize the complementary advantages and perform better than the methods designed for the individual networks.

To my parents, for their love and support.

ACKNOWLEDGMENTS

My journey as a Doctoral student has been extremely memorable due to the vast amount of experiences associated with it. I am really thankful to my advisor, Tarek Abdelzaher, who motivated and inspired me to develop the qualities of a researcher. His critics constantly helped me in improving my skills in different areas such as high quality writing and presentation style. Without his help this thesis would not have been possible. He also provided me with the opportunity to develop a software toolkit that incorporates the algorithms described in this thesis. The development of this toolkit allowed me to present my research to a broader audience. The demo of my toolkit also won the "Best Runner-Up" award at an international conference attended by several researchers actively working in my domain.

I would also like to thank my committee members: Jiawei Han, Hari Sundaram, and Lance Kaplan for the numerous help, suggestions, and support provided for the completion of my thesis. Thanks to the University of Illinois Department of Computer Science and the support staff for helping me carry out the tasks associated with administration and conference travels. Research published in this dissertation has been funded by grants from U.S. Army Research Laboratory. My collaborator, Lance Kaplan, from the U.S. Army Research Laboratory deserves a big thanks for his guidance and advice during my visits. I am really grateful for the numerous brainstorming sessions with him. I would like to thank Tanvir Amin who helped me during the initial phase of my graduate program. I would also like to thank Shiguang Wang for collaborating on a few projects and helping me with the experiments. I am also thankful to other labmates and collaborators, who helped shape up my research. I cannot possibly include all the names here, but I would specially thank Jongdeog Lee, Shuochao Yao, Shaohan Hu, and Huajie Shao.

I was born and raised in India and completed my high school during the era of technical boom which led to the emergence of many big computer related industries. I completed my Bachelor's degree from BITS-Pilani, India with a major in Computer Science. It was my professors there who taught me the fundamentals of Computer Science and Engineering, inspired me to pursue a Ph.D. in Computer Science, and made me aware of the social and ethical responsibilities a scientist should hold. I want to take some time to thank my undergraduate advisor, Professor Chittaranjan Hota, who provided me with the opportunity to work on various research projects under his guidance which inculcated the desire in me to pursue a graduate program.

This journey to doctoral thesis is a long one. Without the love, inspiration, and support from my parents it would have been difficult to complete this journey thousands of miles

away from home. I dedicate this dissertation to my father Giridhar Srinivasan and my mother Sandhya Giridhar. I would also like to appreciate my friends Bhargava Reddy, Virajith Jalaparti, Ashish Khetan, Vivek Madan, Paritosh Katyal, Parimal Singh, Naman Agarwal, Tushar Kaushik, Ankit Aggarwal, and other colleagues who helped me sail through the journey of a graduate student life at Urbana-Champaign.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

According to the United Nations, today, 54% of the world population lives in cities. Arguably, humans are some of the most versatile and widely deployed "sensors" in smart cities. They are often the owners and users of "smart things" on the Internet of Things. They are witnesses of suspicious activity in national security scenarios, survivors and first-responders in post-disaster operations, friendly locals in peacekeeping and stabilization missions, and commuters in intelligent transportation applications. These individuals may relay important information such as real-time state of traffic around recent accidents, locations of spreading post-disaster damage such as flooding or fires, real-time progress of unfolding unpredictable gatherings (e.g., demonstrations and protests), and unusual events that impact safety, such as escalating confrontations or high-speed pursuits. Today, observations of such events are voluntarily shared on social media, leading to the prospect of exploiting social media as state observers of the physical world. It would be very benefiticial for many cyber-physical applications to harvest the collective power of these observations to gain better situation awareness.

The main contribution of this dissertation lies in establishing a signal processing approach to social sensing. Specifically, we ask the question of whether social networks, such as Twitter and Instagram, can be explored as a novel *sensing modality* in cyber-physical systems (CPS). The data provided by human observers is more rich in context than the traditional physical sensors. However, this advantage comes with a set of challenges and we try to address them using basic principles that have a theory to explain why things work. By far, we are not the first to use social media from a sensing point of view which has already been discussed in quite a few past works [1–3]. In this disseration we explore a new sensing approach to reconstruct the aspects of the state of the physical world from social network feeds. Our work is new and unique in the way that (i) we do not perform any supervision of historical data, such as within a geogrpahical region or a given period of time to construct a model that can make prediction on new data, and (ii) we take a completely language agnostic route to avoid the limitations posed by structural differences in text on system design. Using these two underpinnings we build on a system (or service) to enable event detection and tracking using social media. The theme of this dissertation can be described as the following:

"*A language agnostic event detection service using social media streams without supervision.*"

One of the first question we need to answer before moving further is that whether social media platforms are actually feasible for event detection and, if so, which specific platforms should be used to build this service? To answer this we took a look at few Twitter feeds related to the most recent gun shooting incident that took place on March 18, 2018 by two police officers in Sacramento, California leading to the death of Stephon Clark. This incident sparked a few controversies resulting in protest marches around different cities. Table 1.1 shows a random sample of tweets associated with these protest events and provides a strong evidence of a real-world event discussion in social media. With respect to the other question, we consider the platforms that are more open, provide a better opportunity to retrieve publicly shared data, and also fit well with our big goal.

Table 1.1: Example tweets for a protest event

Protestors at Sacramento City Hall #StephonClark protest `https://t.co/Sk2tgwR1BV`
At #StephonClark protest outside Sacramento Co. DAs office, his older brother Stevante leads demonstrators in chan `https://t.co/ZEzENdYqon`
#StephonClark march in NYC marching fearlessly at NYPD tries to repress the protest. #BlackLivesMatter `https://t.co/d3DLxzEd5S`
Protest just ended on G and 9 Street. There will be another demonstration for #StephonClark tomorrow starting at 3 `https://t.co/XA6soXP7sm`

Now that we have indicated the feasibility of social media for event detection, we also need to define the term "*event*". For all practical purposes, we consider events as incidents that are observed by multiple human sensors within a limited time and space. This means that events have a start time, an end time, and also spatial signatures associated with them. Some events can be concentrated to a specific point in the space, some might move across the space, and some might be widespread at different locations within a space. For example, the protest events shown in table 1.1 are widespread in the sense that they are taking place in different cities but the underlying reason is the same. Based on this, we next take a look at few initial challenges that one might meet with while using data from social media platforms.

1.1 CHALLENGES

The following are the main technical challenges associated with using social media platforms for event detection:

- Unlike physical sensors, which are usually distributed identically over a geographical space, the human sensors are mobile and exhibit non-uniform behavior within the same media. For example, the number of observations recorded for the same type of event in two different locations depends on the popularity among the local crowd and the severity of the event. A traffic accident on a major freeway will generate more attention when compared to an accident on a country road.

- The quantity of data generated is not always an indication of an interesting activity. One key part for analyzing the data generated on a social network is to identify and eliminate the noise and redudancy. A single user posting a hundred tweets is more often a noise as opposed to five users posting one tweet each describing the same event. We need to be careful with modeling the noise elimination as each event is not equally popular.

- The information available within a social network platform can often be not sufficient to support the real-world facts. We can take the advantage of analyzing multiple social networks with the help of techniques that strike a better balance between the tradeoffs of the individual networks and find solutions that corroborate the events detected.

1.2 CONTRIBUTIONS

This dissertation addresses the challenges mentioned in section 1.1. We derive these algorithms using basic principles of math and probability theorems that rely on distributions obtained from the underlying data. The final outcome is a service that allows tracking the physical events using Twitter and Instagram. In the following sections we describe the proposed contributions:

1.2.1 Event Detection and Tracking with Twitter

This work leverages the emergence of crowd sensing services, where humans collect data about their environment (using phones), allowing us to exploit the content shared on the social media to detect and track physical urban events. We use Twitter feeds to build the service that embodies novel algorithms for real-time detection, demultiplexing, and tracking of physical events.

1.2.2 Event Detection and Tracking with Instagram

This work develops an algorithm that exploits picture-oriented social networks to detect urban events. We choose picture-oriented networks because taking a picture requires physical proximity, thereby revealing the location of the photographed event. Furthermore, most modern cell phones are equipped with GPS, making picture location, and time metadata commonly available. We consider Instagram as the social network of choice and limit ourselves to urban events (noting that the majority of the world population lives in cities).

1.2.3 Fusing Twitter and Instagram

This work describes the implementation of a service to identify and geo-locate real world events that may be present as social activity signals in two different social networks. Specifically, we focus on content shared by users on Twitter and Instagram in order to design a system capable of fusing data across multiple networks. Our previous works has demonstrated that it is indeed possible to detect physical events using the two social networks. However, many of these signals need corroboration in order to handle events that lack proper support within a single network. We leverage this insight to design an unsupervised approach that can correlate event signals across multiple social networks. Our algorithm can detect events and identify the location of the event occurrence. We evaluate our algorithm using both simulations and real world datasets collected using Twitter and Instagram.

1.3 THESIS ORGANIZATION

The rest of the thesis is organized as follows. Chapter 2 introduces the event detection and tracking technique for Twitter. We use feeds collected from the Twitter API service for specific query keywords to identify clusters of tweets that represent the same event entity in the real-world and also track the progress of these events over time. Chapter 3 analyzes the detected events from the previous chapter to measure the quality of information available from the texts and metadata that can help localize the events to a specific point. Chapter 4 extends the localization of Twitter-events by using a joint localization approach between the the sources and the event clusters. A maximum-likelihood estimation algorithm is described that improves the fraction of localized events in comparison to the baseline method. Chapter 5 presents a new event detection method that uses data retrieved using the Instagram API followed by an estimation technique to approximate the trajectory of

mobile events in Chapter 6. Chapter 7 introduces a new fusion based technique to utilize the benefits of Twitter and Instagram together in order to perform better than the individual social networks. Finally, we conclude the dissertation in Chapter 8, and explore avenues for future research.

CHAPTER 2: EVENT DETECTION AND TRACKING WITH TWITTER

Some of the most widely deployed IoT devices in urban areas are smart phones in the possession of urban individuals. Their proliferation has led to the emergence of crowd-sensing/crowdsourcing services, where humans collect data about their environment (using phones), and servers aggregate the data for various application purposes of interest. With the emergence of social media, a common alternative form of human data entry has become media posts (e.g., on Twitter). This leads to the prospect of building crowdsensing services on top of social media content, exploiting humans as "sensors". In this chapter, we present one such service, called *StoryLine*. 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 algorithms for real-time detection, demultiplexing, 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.

2.1 OVERVIEW

The proliferation of smart phones in urban spaces makes them some of the most commonly deployed devices in the emerging era of IoT. A common use of smart phones has been in data collection by exploiting phone sensors for various city-wide measurement tasks. This use is often termed *crowd-sensing*. While sensors offer a great opportunity for data collection on smart phones, a common alternative form of data entry is by the user via social media applications. This observation leads to the idea of building data collection/crowd-sensing services on top of real-time social media content. We shall henceforth call the idea of estimating physical state from social media posts, *social sensing* [4].

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

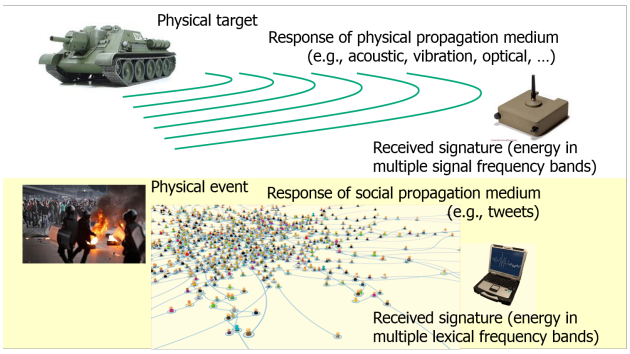


Figure 2.1: The Social Sensing Modality and its Analogy with Physical Sensing

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 2.1. 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. This chapter presents 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.

Our service makes a fundamental contribution to the state of the art in event detection from social media. Namely, we identify separate instances of a given event type (which we call, *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 offer event demultiplexing that has both of the above properties.

The idea that social media posts collectively constitute a form of sensing dates back several years. In their pioneering work, Sakaki et al. [5] proposed an algorithm to detect and track natural disasters, such as earthquakes and hurricanes, using Twitter feeds. The work exploited the spatio-temporal footprint of media posts together with a trained text classifier to detect and localize the events. Since then, work on event detection generally fell into three categories.

First, some papers do detection but not demultiplexing [6–9]. That's to say, they can detect that a major traffic accident occurred, and can separate traffic accidents from floods, but cannot not easily differentiate between two traffic accidents. Many papers in this category do a form of burst detection and text-similarity-based clustering on tweets. Hence, for

example, tweets containing words related to traffic, accidents, and death end up in the same cluster (but can include descriptions of multiple similar events). A second category of work does demultiplexing (separation of concurrent events of the same type) by clustering similar tweets based on location (and time) [10–12]. They often use some notion of coherence (or higher frequency of keywords that are *semantically related*) at a given location as an indicator that an event occured at that location [13]. Unfortunately, on Twitter, less than 2% of tweets are geotagged [14, 15], so this approach can easily miss small events. While user account registration information is more commonly available (about 25% of accounts have it), it is course-grained (city-scale only), and hence cannot distinguish different local events. Finally, some papers indeed do demultiplexing without location information [14, 15]. However, they use different degrees of natural-language processing or machine learning, and thus are language-specific and/or need prior training. For example, some papers use shallow analysis of text to identify location keywords (e.g., references to specific streets, cities, or landmarks) [14–17], and cluster tweets based on locations referred to in their text. This paper thus opens up a new category of event detection methods that can *demultiplex* events, *without use of location* infocmation, in an entirely *unsupervised* NLP-free fashion.

Demultiplexing is essential to our IoT service, where a city planner, for example, might want to know accurate statistics of events occurrence over time, which implies knowing how many events (say, car accidents) occured. Not relying on location metadata means we can identify more events. Not using language features means the service can be deployed internationally at little or no additional cost, regardless of local language. Finally, our approach is unsupervised and hence does not require classifier training [18, 19], bootstrapping [20], or significant pre-processing [21, 22]. 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 and propose our solution to unsupervised event detection, demultiplexing, and tracking in Section 2.2. The evaluation is presented in Section 2.3. We discuss the related work in Section 2.4 and conclude the chapter in Section 2.5.

2.2 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".[1] 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 for each query to better filter the desired event category. 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.

2.2.1 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. The separation of posts by event instance occurs in an entirely unsupervised manner (i.e., without prior training or prior knowledge of event-specific keywords/tags) and is language agnostic (i.e., does not rely on language-specific knowledge). 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

[1]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.

observations, below which an event is of no interest for the purposes of this paper. 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. As mentioned earlier, approaches that detect events by looking for spikes in some coherence metric (e.g., a spike in keywords that commonly co-occur) do not do well on demultiplexing different concurrent *instances* of the event. In contrast, we look at spikes in keywords that *do not* commonly co-occur. An information gain metric is derived to measure such spikes. For example, in description of car accidents, a particular car accident involving a drunk driver who ran over a dog on Bay 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.

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 paper is to determine for each time slot, k:

- The set of ongoing event instances, $\mathcal{E}(k)$.
- The observation summary, $Summary_i[k]$, for each event instance, $E_i \in \mathcal{E}(k)$.

2.2.2 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.[2] 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 $P(W_2|W_1)$ 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 keywords 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

[2]In a prior literature [23], 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.

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 apprear 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. Accorrdingly, 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.

2.2.3 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 be the variable associated with the keyword pair and Y be the variable associated with time slot. 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 straighforward physical meaning respectively.

$H(W)$ dentoes 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_j (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:

$$\begin{aligned} H(Y) &= -p(k)log_2p(k) - p(k-1)log_2p(k-1) \\ &= -\frac{w_k}{(w_k + w_{k-1})}log_2\frac{w_k}{(w_k + w_{k-1})} \\ &\quad -\frac{w_{k-1}}{(w_k + w_{k-1})}log_2\frac{w_{k-1}}{(w_k + w_{k-1})} \end{aligned} \tag{2.1}$$

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:

$$\begin{aligned} H(Y|s_i) &= -p_i(k)log_2p_i(k) - p_i(k-1)log_2p_i(k-1) \\ &= -\frac{w_k^i}{(w_k^i + w_{k-1}^i)}log_2\frac{w_k^i}{(w_k^i + w_{k-1}^i)} \\ &\quad -\frac{w_{k-1}^i}{(w_k^i + w_{k-1}^i)}log_2\frac{w_{k-1}^i}{(w_k^i + w_{k-1}^i)} \end{aligned} \tag{2.2}$$

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

$$IG_j = H(Y) - H(Y|s_j) \tag{2.3}$$

Equation (2.3) 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.

2.2.4 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*.

2.2.5 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 2.2. 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 2.2.) 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.

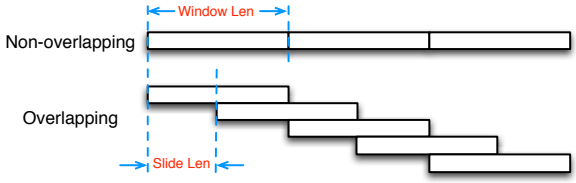


Figure 2.2: Illustration of the non-overlapping sliding window and the overlapping sliding window (with 50% overlapping).

2.2.6 Final Architecture

The architecture of our social event tracking system is shown in Figure 2.3. The targeted social medium of our system is Twitter [24]. The system is implemented in Python27 and integrated into an existing social sensing tool, Apollo [3]. 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.

[3] http://apollo3.cs.illinois.edu

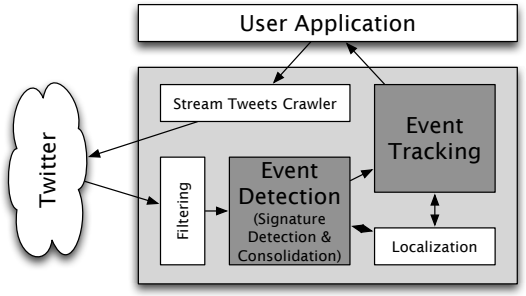


Figure 2.3: Event Tracking System Architecture

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. [17]), but it is not relevant to this chapter. Please note that unless the user calls KILL, the tracking task keeps working.

2.3 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.

2.3.1 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.

2.3.2 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 formal mathematical definitions of these distance metrics is provided below.

Let S_i be the set of key words of the tweet cluster C_i. For each word, $w \in S_i$, let $f_i(w)$ denote its frequency. For notation simplicity, if $w \notin S_i$, we define $f_i(w) = 0$. We investigate four broadly applied distance metrics described below for defining the lexical frequency domain distance between clusters.

- **Jaccard Distance:** The Jaccard similarity (JS) between clusters C_i and C_j is defined by

$$JS(i,j) = |S_i \cap S_j|/|S_i \cup S_j|,$$

where $|S|$ denotes the cardinality of the set S. The Jaccard distance is defined by $1 - JS(i, j)$.

- **Term Frequency Difference Ratio:** The term frequency difference (FD) between clusters C_i and C_j is defined by:

$$FD(i, j) = \sum_{w \in S_i \cup S_j} |f_i(w) - f_j(w)|,$$

where $abs(X)$ denotes the absolute value of X. The term frequency difference ratio is the normalized term frequency difference, i.e. $\frac{FD(i,j)}{\sum_{w \in S_i} f_i(w) + \sum_{w \in S_j} f_j(w)}$.

- **Cosine Distance:** Cosine Similarity (CS) is defined by

$$CS(i, j) = \frac{\sum_{w \in S_i \cap S_j} f_i(w) \times f_j(w)}{\sqrt{\sum_{w \in S_i} (f_i(w))^2} \times \sqrt{\sum_{w \in S_j} (f_j(w))^2}}.$$

CS measures the cosine of the angle between two vectors whose elements are f_i and f_j. The more similar the two vectors, the smaller the angle between them. The Cosine distance is defined by $1 - CS(i, j)$.

- **KL Divergence:** The KL divergence, KL, is a non-symmetric measure of the difference between two probability distributions, defined as follows in our case:

$$KL(i, j) = \sum_{w \in S_i \cup S_j} p_i(w) \ln \frac{p_i(w)}{p_j(w)},$$

$$p_i(w) = \frac{f_i(w)}{\sum_{w \in S_i \cup S_j} f_i(w)}, p_j(w) = \frac{f_j(w)}{\sum_{w \in S_i \cup S_j} f_j(w)}.$$

Note that, when $f_i(w) = 0$ or $f_j(w) = 0$, $KL(i, j)$ is malformed. To avoid this problem, we add 1 to $f_i(w)$ and to $f_j(w)$ for all w.

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 2.4 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 2.2. The error rate

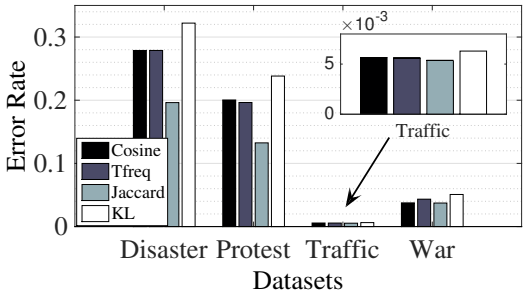


Figure 2.4: The consolidation error rate.

of signature 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.

2.3.3 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 2.1 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 geotagged 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.

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

1. *ET* [9]: In this work, an event is detected using common bi-grams, where the bi-grams

Table 2.1: 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%	0.1%	0	0

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-occurence (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* [25]: This work proposes an online variation of LDA (Latent Dirichlet Allocation) [26], 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 2.2: Event Detection Precision Comparison

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

Table 2.2 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, 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 2.5 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.

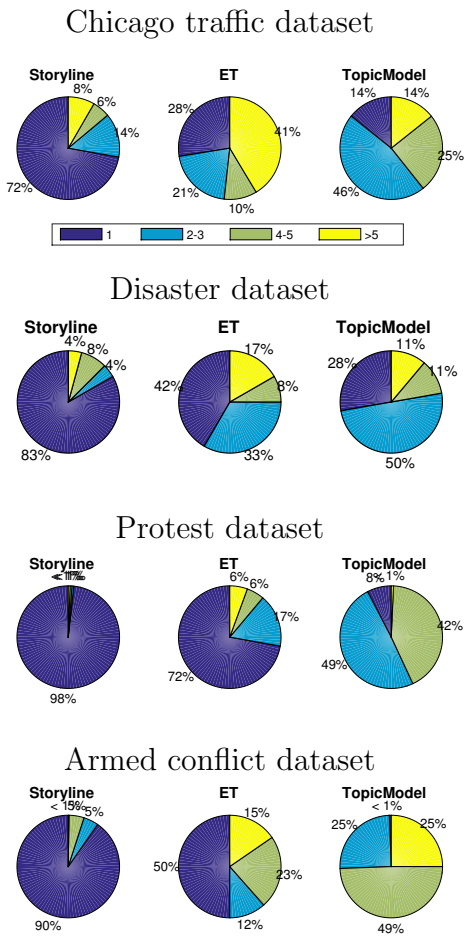


Figure 2.5: The purity pie charts.

2.3.4 Language Agnostic Analysis

We also collected Twitter datasets in non-English languages to show the performance of our component when new languages are involved. The query keywords used for this collection are shown in figure 2.6. All the keywords mean the same in different languages, which is "protest". We again evaluate the demultiplexing quality for these datasets which is reported in table 2.3.

Language	Keyword
English	protest
Korean	시위 폭동
Arabic	مظاهرة
Russian	протест
Hindi	विरोध

Figure 2.6: Twitter query keywords used for non-English dataset collection

Table 2.3: Demultiplexing Quality

Instances	**Korean**	**Hindi**	**Russian**	**Arabic**
1	52.83%	63.75%	93.34%	83.33%
2-3	41.5%	27.5%	6.67%	6.67%
4-5	3.77%	5%	0%	6.67%
5+	1.88%	3.75%	0%	3.34%

There is clearly a variation in the performance when different languages are used. One reason for this is the structure of sentences in each language. For example, *Hindi* has around twice the number of characters as *English* and *Korean* has some special characters that are appended to normal words and change the expression. These are just a few examples and since our research is not really about understanding the language we leave this work for future direction.

2.3.5 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 2.4: Real-time Earthquake Detection Summary

Earthquake Location	Earthquake 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 2.4 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.

2.3.6 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 2.5.

An earthquake occurred 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.

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.

2.4 RELATED WORK

The idea of using social networks as sensor networks was discussed in recent literature [27, 28]. While much work focused on analysis of reliability of crowd-sourced observations, this paper 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. [20], 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. [29] and Benharus et al. [30]. Shamma et al. [29] 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. [30] 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 [31]. An example is work utilizing optimizations of k-means algorithms to cluster data streams, as proposed by Ordonez [32] and Zhong [33]. 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 [25,34,35]. Lau et al. [25] proposed an online variation of Latent Dirichlet Allocation (LDA). In LDA, each topic is modeled as a multinomial distribution of words in a volcabulary, and each document is modeled as a multinomial distribution of k topics, where k is a predifined 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. [25] is incrementally updating the priors in each time window based on previous calculated parameters, and maintaining the one-to-one correpondence 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. [34] 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. [35] 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. Futhermore, 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.

Similarly, Aggarwal and Subbian [8] 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 [10–12]. 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 [14, 15]. 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 [13]. 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.

Finally, like us, some recent papers indeed propose demultiplexing schemes that do not use location metadata [14–17]. 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 [14, 15, 17], then clustering by the extracted location information. 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. Imortantly, however, it does so based on statistical analysis alone, and not linguistic analysis of data. Unlike other event detection techniques that rely on clustering, 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* is 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 this chapter.

2.5 SUMMARY

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 paper 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 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.

Table 2.5: 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 http://t.co/MULEuWhx3Q http://t.co/QeRKg8QgYp
		RT @BBCBreaking: At least 876 killed in Nepal #earthquake; deaths also reported in India, Tibet & Bangladesh http://t.co/3BTo9l1QZ4 http://ǎ026
04/26/2015	help, nepalearthquake	RT @cnnbrk: At least 2,263 people have died in Nepal from massive #NepalEarthquake and aftershocks, official says. http://t.co/hCyjO7YyS7
		Anyone with information about my son Joseph Patrick please help #NepalEarthquake #Pray4Joe http://t.co/X2Kn7mOtRO http://t.coǎ026
04/27/2015	surges, devastation, drone, thankyoupm, donations	Nepal #earthquake: Death toll surges to 3,218; four aftershocks felt in last 12 hours http://t.co/Njvru9k2kQ
		@cnni: New drone footage shows the extent of devastation from the #NepalEarthquake: http://t.co/7PiPjayQZ1https://t.co/phIGRkYoZQ
		#ThankYouPM for massive rescue and relief operation by India in Nepal after #earthquake
		Nepal Earthquake: Facebook to Match Donations Made for Victims http://t.co/aLooadYNxj Free Submission http://t.co/J90dT2qnXb
04/28/2015	salute2indianforces, koirala, sanjay, humanitarian	Thank you very much Indin 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... http://t.co/4EtmH28EwC #tcot
		Live: Nepal earthquake kills 4,352, PM Sushil Koirala says death toll could reach 10,000: A high-intensity ear... http://t.co/A68VtR6hWK
04/29/2015	survivor,hours, hospital, field, miracle	Nepal earthquake survivor drank urine while trapped for 82 hours http://t.co/v9DHM5Jhnf #worldnews
		That is amazing, Nepal Army rescued a 4-month kid alive after 22 hours! ::http://t.co/KzJPJeZDCx https://t.co/HvTkvS0Ba0 via @sharethis
		RT @haaretzcom: Nepal earthquake updates / Israeli field hospital opens, to treat 200 people per day http://t.co/PMwRlRT6YO http://t.co/s9i
04/30/2015	pakistan, serves, masala, teenage, lydia	Pakistan serves 'beef masala' to earthquake-hit Nepal via /r/worldnews http://t.co/GoFJO09mJP
		Teenage boy pulled out of rubble alive five days after Nepal earthquake http://t.co/0kiAigYE7M #telegraph #news
		Lydia Ko donating earnings to Nepal relief effort: The 18-year-old Ko, ranked No. 1 in the world, successfully... http://t.co/2nquCITqJa

CHAPTER 3: EVENT LOCALIZATION WITH TWITTER

Social networks, such as Twitter, carry important information on ongoing events and as such can be viewed as networks of sensors that monitor and report events in the physical world. In this chapter, we concern ourselves with the challenge of event localization from Twitter feeds. We explore the quality of information that can be derived either directly or indirectly from microblog entries regarding locations of ongoing events. Contrary to prior work that used Twitter to map regions of large-footprint events, or derived coarse-grained location information, in this chapter, we are interested in point-events, such as building fires or car accidents, and aim to pin-point them down to a street address. An algorithm is presented that identifies distinct event signatures in the blogosphere, clusters microblogs based on events they describe, and analyzes the resulting clusters for fine-grained location indicators. An exact event location is then derived by fusing these indicators. To evaluate the quality of derived location information, we use road-traffic-related Twitter feeds from 3 major cities in California and compare automatic event localization within our service to manually obtained ground truth data. Results show a great correspondence between our automatically determined locations and ground-truth.

3.1 OVERVIEW

The proliferation of social networks, where real-time information about ongoing events is broadcast, suggests that the classical sensor network model may be extended to include social sensing, where humans (either intentionally or unwittingly) act as sensors.

Prior work suggests a methodology for identifying events in social network feeds [36]. In contrast, this chapter focuses on the challenge of *localizing* observed events. Specifically, we address the question: what is the quality of information within social network feeds, contained either directly or indirectly, about the locations of events they describe? We focus on Twitter [24] feeds and point-events (such as car accidents), where the event is associated with a small time and location footprint. While identifying the overall footprint of widely-spread events, such as a heat wave in Europe, is also interesting, it is beyond the scope of this chapter.

Localizing events from Twitter feeds is challenging because only 2-3% of Twitter data is geotagged. Hence, a direct attempt at identifying locations of events through tweet geo-tags is not effective. Moreover, individuals who are present at one location will often discuss events that transpire at a different location. In the context of event localization, one is

interested in locations of events, not sources. Accordingly, the geo-tag associated with the source, even when present, may not necessarily correctly reflect the sought event location.

Instead of relying only on geo-tags, our service resorts to a complementary technique that collectively improves the localization accuracy. The events detected as per the previous chapter are analyzed for direct location references in tweet text. These references are translated into coordinates using the Google Maps geotagging API [37]. Finally, the resulting coordinates are themselves averaged (after outlier elimination) to obtain an approximate event location.

We compare the quality of localization obtained using the above insights by considering Twitter feeds related to road traffic incidents from areas in and around three different cities in California, namely San Francisco, San Diego, and Los Angeles, collected over a period of three weeks. These feeds are first processed to recognize events and then analyzed to determine possible event locations using the methodologies discussed above. A comparison between the determined locations and manually collected ground truth is then made. The comparison suggests that sufficient location information is available (either directly or indirectly) in tweets that makes high quality event localization possible.

The work is distinguished from recent efforts that attempt to localize individual sources from their tweets [38–40], in that the goal in this paper is to localize physical *events*, not sources. It is also different from prior coarse-grained attempts to localize events that focused on identifying the city where the events took place [41], or the general region of wide-spread events, such as Earthquakes [5]. In contrast, we focus on highly localized (i.e., point) events, and seek to identify their street addresses.

The remainder of this chapter is organized as follows. We present our data collection and event identification framework in Section 3.2. The evaluation of localization accuracy is discussed in Section 3.3. We describe related work in Section 3.4. Finally, conclusions are presented in Section 3.5.

3.2 SYSTEM DESIGN

There has been a big increase in the use of social networks to report various events observed in day-to-day life. Since we aim to identify locations of events reported on social networks, we are inherently concerned with those events that attract people's attention, leading to the propagation of information about them in the blogosphere. An example of such an event is a building fire, a bombing, or a car accident. The event will likely cause comments that will make their way to the social network.

Most Twitter users do not attach their current GPS information to their tweets. Many

event descriptions, however, are accompanied by keywords that act as location indicators. Such indicators may have different levels of precision. Some may be fairly precise, such as tweets indicating a specific freeway exit. Others may be incomplete. For example, they might only specify a street address, but not the city or state, or might indicate a local landmark that is not a globally unique name. By clustering together tweets refering to the same event, two benefits are achieved. First, redundant information may be obtained, hence increasing confidence in the location estimate via corroboration. Second, by combining different partial indicators from different tweets in the same cluster one may pinpoint the event. For example, a reference to a "Stadium" in one tweet and "Fenway Park" in another may increase the confidence that the location in question is the Red Sox stadium. Niether "Stadium", nor "Fenway Park" are individually unambiguous location indicators. Furthermore, an advantage of combining location indicators from tweet text together with source-based location indicators is that local spatial references often omit coarser-grained location information such as the city and state, restricting themselves to a *street address* or landmark name. For example, "Illini-Alert: Suspicious package at Education Bldg, 1310 S Sixth. Evacuate 2, 3, 4 floors to north side of bldg. Bomb unit on scene" (an actual tweet). Hence, such location references are often ambiguous outside their local context. Conversely, Twitter user accounts are more likely to contain *coarse-grained* location information, such as city and state, because it presents the least threat to user privacy. Combining the two, a complete and an unambiguous address can often be obtained.

For this chapter, we use the California highway system as a running example. Below we describe our system and investigate the degree of accuracy to which traffic events identified in this dataset can be localized. We then analyze the effect of different factors that contribute to the quality of location information.

3.2.1 Data Collection and Event Detection

In order to evaluate the quality of location information in microblog data, we use the StoryLine service to collect tweets followed by detection of event clusters. In our case, we used the word "Traffic" as query.

Table 3.1 shows a few examples of tweet clusters associated with different events identified by the StoryLine system. The first column is the high-information-gain keyword pair that identifies the event. The second column is the set of all tweets that contains the pair and hence is said to describe the event. The table produced for all events in the dataset is then input to the localization step.

Table 3.1: Examples of Events and Tweet Clusters

Keyword Pair	**Event**
crash, rancho	(1) #BREAKING: Massive crash has traffic down to a trickle on SB15 at Via Rancho Parkway in Escondido. (2) #BreakingNews #SigAlert Major traffic crash on the 15-southbound at Via Rancho Parkway. traffic backed up for miles. some lanes open now.
collision, north	(1) Traffic collision on SB I-5 just north of Encinitas Blvd. Vehicle hit center median. (2) One lane blocked on SB I-5 just north of the San Diego-Coronado Bridge due to traffic collision.
monica, santa	(1) Avoid Santa Monica Blvd. in WestHollywood between Fairfax and Orlando. #roadworks #traffic (2) Training from 830-1115. And then sitting in traffic from 12-2 to get up to Santa Monica is not recommended #fortheloveoffooty (3) SMFD responding to a Motorcyclist Down in the intersection of 23RD ST / SANTA MONICA BLVD. Possible traffic congestion. Inc.#14009277
lanes, pasadena	(1) All lanes were closed on the westbound 210 Freeway in Pasadena because of the crash and rush hour traffic (2) NBCLA: TRAFFIC ALERT: Big rig crash shuts down lanes of WB 210 Fwy in Pasadena. (3) SIGALERT Pasadena - 210 W before Rosemead: The carpool & 2 left lanes are closed due to a crash. Traffic bad from Myrtle. E heavy from Lake.
water, break	(1) Avoid Sunset Blvd around UCLA campus. Traffic really piling up from major water main break. En route to the scene. (2) If your commute takes you on #Sunset Blvd be aware that a water-main break near #UCLA has shut down Sunset from Hilgard to Veteran. #Traffic (3) Reports saying water main break has created a 50-ft wide sinkhole, flooding parts of #ucla. Traffic in area at standstill.
malibu, canyon	(1) JUST IN: #Malibu Canyon Road is closed between Seaver Drive & #PCH due to police activity per CityMalibu. SB traffic is being diverted (2) Traffic Accident Involving Robbery Suspect Closes Malibu Canyon Road (3) UPDATE: #PCH WB closed at #Malibu Canyon Road for police activity. Also, a traffic accident is blocking WB PCH near Webb Way

3.2.2 Content Analysis to Extract Location Indicators

The task of identifying locations associated with an event is challenging due to sparsity of tweets that actually contain exact location (e.g., GPS) information. Tweets describing physical events, however, usually mention spatial landmarks or partial addresses that can be used as implicit tags in order to determine possible locations. The task requires some preprocessing so that spatial information can be extracted. The raw tweets are first tokenized to delimit individual keywords forming event descriptions. These tokens are then tagged using a part-of-speech (POS) tagger. We found that phrases describing locations were always composed of Nouns (NN), Determiners (DT), Adjectives (JJ), Cardinal Numbers (CD),

Conjunctions (CC), and Possessive Endings (PE). There is a special case in which freeway numbers (such as I-91, etc) were classified as prepositions (PRP) because they started with "*I*". This case was handled separately.

The examples provided in Table 3.2 show all possible POS tags for phrases that constitute location identifiers, obtained by manual inspection of events over the collected Twitter dataset. Each row in the table indicates a rule number, the POS tags contained in the rule, and an identifier example for the rule. The "+" sign indicates the presence of a tag at least one or more times, the "?" sign indicates the presence of a tag zero or one time, and the "|" sign indicates the presence of one of the two tags. Except for the first rule, all other rules are composed of one or more previous rules in the table. All the POS tags that matched any of the indicated rules are marked as "LOC" suggesting a possible location identifier for the event.

Table 3.2: POS Tags for Location Identifiers

Rule	**POS Tags**	**Identifier Examples**
1	<NN>$^{+}$	tiburon blvd, san manteo
2	<DT>?<JJ>?<1>	the golden gate bridge
3	<CD>?<2>	third street
4	<2> <CD>?	freeway 91
5	(3|4) <CC | PE> (3|4)	sunset blvd and market street, levi's stadium

Besides actual location indicators, several phrases were identified by these rules that did not contain location information. They simply happenned to match. In order to avoid such inappropiate phrases from being picked up, we observed a common pattern for the majority of actual location identifiers. Namely, true location-identifiers (LOC) were commonly preceded by Prepositions (IN) such as *in*, *around*, *between*, and *after*. Thus an additional grammar-based rule was implemented to extract all such phrases providing chunks of actual possible location identifiers. This grammar only looked at the LOC tags that followed any type of IN tags in the description. The examples in table 3.3 show examples of chunks produced by such a grammar.

Thus, for each event, we identify a set of location indicators (or references) represented as chunks correspoding to each of the tweets within the cluster. Since a cluster may consist of several tweets, we can get multiple location indicators for the same event. Independently from extracting those indicators, we extract location metadata for all sources who tweeted about the event (i.e., all sources who issued a tweet in the cluster). We use majority voting to decide on the predominant coarse-grained location, given by city, state, and country. This location is then combined with each of the fine-grained indicators extracted from tweet text,

Table 3.3: Extracted Location Identifiers

Tweet	Chunks
this overturned tanker in marin has created a huge jam on wb 580 clear across the richmond san rafael bridge & Four	(marin),(wb 580),(richmond san rafael bridge)
attn srv in and around the presidio area may experience delays from traffic due to on going traffic incident on golden gate bridge	(presidio),(golden gate bridge)
southbound mission blvd between stevenson blvd and las palmas ave is closed due to a traffic collision	(stevenson blvd)
castro valley car fire slowing traffic at 580 and crow canyon rd wildfire averted	(580), (crow canyon rd)
a big rig truck hit a muni light rail train at third street and innes avenue	(third street),(innes avenue)

and the combination is geotagged using the Google maps API. Once geotagged, the results form a point cluster. Outliers are eliminated. The coordinates of non-outliers are averaged to pin-point the event centroid.

3.3 EVALUATION

To evaluate the degree to which location information contained in tweets helps identify event locations, we needed to find a data set, where ground truth information is available. We opted for data on traffic accidents in California. In other to limit the amount of data handled so that accuracy of our location estimation can be manually verified, we restricted the study to Los Angeles (LA), San Francisco (SF), and San Diego (SD). Tweets were collected from each of these cities for a period starting from July 13, 2014 to August 2, 2014.

Table 3.4 shows the number of events referring to traffic incidents, among those returned by the information-gain-based event detection algorithm, along with the total number of original tweets (excluding retweets) referring to those events.

Table 3.4: Tweets and Event Counts

City	Events	Original Tweets	Tweets/Event
San Francisco (SF)	66	181	2.742
San Diego (SD)	24	68	2.833
Los Angeles (LA)	142	411	2.894

Note that, by construction of our experiment, only those tweets were collected that originated from sources within the indicated geographic circles. Hence, only those tweets were collected whose sources published location metadata (otherwise they would not have matched

our crawler's query). Our purpose, however, is to experiment with realistic conditions, where only a subset of source profiles offer location metadata. The authors of [42] report that for a random sample of over 1 million Twitter users only 26% have listed a user location as granular as a city name. Hence, in the first part of this evaluation, we consider the ideal case where the location metadata of all sources is present, while in the second part we use 50%, 25%, and 10% of the metadata, respectively (while randomly deleting the rest). This allows us to explore the degree to which availability of source location metadata affects localization accuracy.

We first present how the accuracy of our service varies with the information gain (IG) values of the keyword pair signatures used to determine the events. In order to determine the correctness of our algorithm, we manually compared the output of the algorithm to the location of the accident as reported by the California Department of Transportation. Table 3.5 shows the percentage of events that were correctly localized for different intervals of the IG values when the complete metadata from sources was used to determine the city and state. In Los Angeles and San Diego, 100% of event clusters are correctly localized for events with information gain of more than 0.01, whereas in San Francisco the percentage is 85.7%. The percentages tend to drop for events with a lower information gain.

Table 3.5: Percentage of Correctly Localized Events for Different Information Gain Values & Full Source Metadata

IG Range	**San Francisco**	**San Diego**	**Los Angeles**
[0 - 0.005)	76.7%	80.9%	75.36%
[0.005 - 0.01)	68.75%	100%	100%
[>0.01]	85.7%	100%	100%

The lower accuracy values for San Francisco compared to Los Angeles and San Diego can be attributed to the fact that more tweets were collected on an average for the latter cities. Hence, the number of tweets per event was lower for San Francisco, resulting in lower accuracy. Overall, we correctly localized for Los Angeles 108 events out of 142, for San Diego 20 events out of 24, and for San Francisco 50 events out of 66. Table 3.6 shows examples of events that were did not corectly localize by our algorithm.

For the second part of the evaluation, we randomly retained 50%, 25%, and 10% of the source metadata respectively, which was used to identify the city and state part for an event. As before, source-based coarse-grained location information was decided by majority vote using the available metadata (for both the original tweets and retweets) in a cluster. We again evaluated the localization accuracy for different IG ranges. The results are presented in Table 3.7, showing a modest drop in accuracy with reduced prevalence of location

Table 3.6: Examples of Incorrectly Localized Events

Event	**Cluster of tweets**
murray, alert	(1) traffic alert westbound i 8 closed near lake murray boulevard while crew investigates suspicious device (2) hmmm, law enforcement activity rt traffic alert law enforcement activity forces closure of lake murray blvd on ramps to i 8 (3) traffic alert law enforcement activity forces closure of lake murray blvd on ramps to i 8
lanes, heads	(1) heads up wb 80 before ashby motorcycle down lanes 2 3 blocked traffic solid from central ave (2) heads up nb 101 before ellis st ax involving an o t pick up truck blocks the rt 2 lanes traffic b u to fair oaks wb 237 also affected

Table 3.7: Percentage of Correctly Localized Events for Different Information Gain Values & Partial Metadata

IG Range	**Metadata %**								
	50%			**25%**			**10%**		
	SF	SD	LA	SF	SD	LA	SF	SD	LA
[0 - 0.005)	51.16%	71.4%	52.89%	48.83%	66.67%	43.47%	44.18%	66.67%	39.13%
[0.005 - 0.01)	56.25%	100%	100%	50%	100%	100%	43.75%	100%	100%
[>0.01]	85.7%	100%	100%	57.14%	100%	100%	57.14%	100%	100%

information.

Table 3.8: Percentage of Non Localized Events with varying Metadata Prevalence

City	**Prevalence of Source Location Metadata %**			
	100%	**50%**	**25%**	**10%**
San Francisco	1.25%	6.27%	7.53%	10.8%
San Diego	0.6%	1.9%	2.62%	2.62%
Los Angeles	1.08%	8.07%	9.78%	11.8%

We also examined the events which were not localized by our algorithm. Table 3.8 shows the percentage of physical events that could not be localized when the source metadata was varied from 100% to 10%. Inability to localize an event is attributed to insufficient location information. It is clear that, as the source metadata decreased, more events were not localized. Table 3.9 shows examples of few events that did not get localized by our algorithm even with full source location metadata. The location specific tags in these events did not get classified correctly by the grammar rule as described in the design section of this paper.

We also computed the effect of clustering of tweets corresponding to the same event, by comparing the percentage of correctly localized events in the presence of clustering to the

Table 3.9: Examples of Non-Localized Events

Event	Cluster of tweets
police, chase	(1) awesome police chase in heavy traffic as some guy tried to get away from police on a very crowded i 80 just minutes ago didn't work (2) oh shit i'm in the middle of a police chase on 80 not smart to go through traffic when police are after you
gorge, mission-trailsfire	(1) mission gorge rd missiontrailsfire rt traffic is stopped traveling east but not west can't say why (2) rt traffic on mission gorge rd is now stopped traveling east at golfcrest missiontrailsfire
pursuit, elcaminoreal	(1) pursuit sb5 elcaminoreal 90 120mph traffic getting heavy (2) pursuit sb5 elcaminoreal 90 120mph traffic getting heavy and i'm sure somewhere in his dense head, he thinks he'll get away

corresponding percentage when localization was done based on individual tweets. Table 3.10 shows the results when clustering was used (C) and when single tweets were localized separately (NC). Localization was performed with source metadata prevalence ranging from 100% to 10%. This table allows shows that clustering is an important component in improving localization accuracy, especially when source metadata is not prevalent.

Table 3.10: Clustering versus No Clustering

	Metadata %							
	100%		**50%**		**25%**		**10%**	
City	C	NC	C	NC	C	NC	C	NC
SF	75.75%	72.72%	71.21%	68.78%	50%	46.96%	45.45%	40.9%
SD	83.33%	79.16%	75%	70.83%	70.83%	58.33%	70.83%	58.33%
LA	76.05%	76.05%	54.22%	52.81%	45.07%	42.55%	40.84%	39.43%

3.4 RELATED WORK

This chapter is related to research on localization and visualization of events described in social networks. There have been several past efforts in the domain of localization for social networks. Early work [43] tries to identify places of interest in tweets by building a unigram language model. The approach uses web pages to improve localization accuracy. It is geared for localizing landmarks, as opposed to fleeting events. In other interesting work [44], images from the Flickr image sharing network are placed on a map by exploiting textual tags associated with them, described by users. An initial language model is built over a geo-coordinate grid by placing images along with tags for which locations are already known and later used for prediction of other images.

Most prior work focused on identifying coarse-grained location information of sources or

events. For example, the authors in [40] build a probabilistic localization framework and a language model by looking at conversations between users in Twitter. Their work tries to find city-level information for users, without relying on any other inputs besides the content of their conversations. A work that comes close to ours describe the approach to localize events detected on Twitter [41]. However, it does so at the granularity of a city, not a street address. Similarly to ours, their approach is based on clustering tweets having the same text and using metadata available such as user profile location information, content of tweets, and GPS coordinates of tweets to build a Bayesian inference model that predicts the city within a confidence interval.

Our work complements that past works by localizing point-events on Twitter. These events are first detected by using an information-gain approach to find signatures which are used to form a cluster of tweets about each event. We do not aim to localize individual tweets or users in a social network, nor do we stop at city level granularity. Unlike building a language specific model for each city, as shown in some of the previous research, we rely on simple language processing techniques to predict location specific tags at a very fine-grained level. The locations identified by our algorithm are highly accurate to street level information for most of the events.

3.5 SUMMARY

In this chapter, we illustrate a localization tool capable of automatically identifying the locations of physical events of interest from social network data. Preliminary evaluation shows that the tool correctly identifies event locations most of the time. Not surprizingly, more tweeted events (or more precisely, events with a higher information gain) are localized more accurately. The correctness of localization depends on the number of original tweets and retweets about the event. More original tweets correlates with a higher likelihood of finding fine-grained location information, whereas more retweets correlates with finding coarse-grained (city, state) information. This is because retweets do not offer new text to extract fine-grained locations from, but do offer additional sources that are typically local when considering local events. Overall, the fraction of events not localized successfully remains low, even when most source do not have location metadata.

CHAPTER 4: IMPROVING LOCALIZATION QUALITY FOR TWITTER

In the previous chapter we formulated a physical event localization problem from social network data. This chapter improves on the above results by formulating a *joint* localization problem of events and sources, leveraging the fact that sources on social networks often have a location affinity: They tend to comment more on events in their locations of interest. While social networks, such as Twitter, do not offer source location information for the majority of sources, we show that our algorithms for jointly inferring source and event location significantly improve localization quality by mutually enhancing location estimation of both events and sources. We evaluate the performance of our algorithm both in simulation and using Twitter data about current events. The results show that joint inference of source and event location allows us to localize many more of the events identified in real-world datasets.

4.1 OVERVIEW

This chapter presents and evaluates the first algorithm for *joint* localization of sources and events in social networks acting as sensor networks. The intuition behind our joint event and source localization approach lies in that sources often have a location affinity. (We verify this intuition in Figure 4.2 of Section 4.2.) Hence, in the common case, they will tend to report local events or events at a limited number of locations that are of most interest to them. Only 26% of Twitter users have location information in their profile. This information tends to be coarse-grained (e.g., city level). Fine-grained location information (e.g., current GPS coordinates) is available for less than 3% of the tweets. However, aggregating a sufficient number of tweets about an event, it becomes possible to both localize the event with a higher probability and learn about the locations or location interests of sources. Doing so iteratively over time eventually produces an accurate estimate of both source and event locations; an insight we explore in this chapter.

We evaluate our framework using three Twitter datasets that we collected. Evaluation results show that given a small fraction of ground truth labels associated with events and/or sources, we are able to jointly localize a much larger fraction of both events and sources with high accuracy. On comparing with a baseline approach, our framework performs better for accurately determining the location labels associated with the events for all the test cases.

The rest of this chapter or organized as follows. We present the design of our framework in Section 4.2. The evaluation of localization accuracy is discussed in Section 4.3. Finally,

conclusions are presented in Section 4.4.

4.2 SYSTEM DESIGN

The objective of our localization system is to detect events and localize them to the finest degree of granularity. We begin by describing the concept of an event. We are interested in geographically contiguous events and thus aim to associate them with a *single* location label at an appropriate level of spatial granularity. For a counter-example, a terror act that results in simultaneous bombing of different capitals around the world is not an event that we consider in this chapter. Such an event will be associated with multiple different legitimate locations at the same time. In contrast, a political rally in a city, the outbreak of an epidemic in a country, or the collapse of a bridge across a river constitute events that are appropriate for localization using techniques developed in this chapter. Some of these events can be localized more precisely than others. For example, the collapsed bridge could be associated with a specific street address, whereas a rally may be localized to the level of a city, and the epidemic to the level of a country. Our system attempts to find the finest-granularity *single* location associated with each automatically-identified event.

Our work is novel in jointly localizing sources and events in the context of social (as opposed to physical) sensing. In doing so, we exploit the concept of *location-affinity* of sources. Informally, the location affinity of a source refers to its tendency for reporting observations about a specific location. It may not be the same as the actual location of the source, although many sources will naturally have an affinity for their home location. For example, a source who lives in France after immigrating from North Africa might have an affinity to both French and North African locations. We exploit this property to probabilistically infer event locations. The system is composed of four functional components as shown in Figure 4.1.

- *Event detection:* Use the StoryLine system to detect events from the incoming set of feeds.

- *Location initialization:* For each event, we identify the sources contributing to the reporting and compute initial (noisy) estimates for both source location affinity and event locations. Our initial estimate of source location affinity simply postulates that sources have affinity to the location mentioned in their profile. Our initial estimate of event location is based on any location or landmark references contained in tweets describing the event.

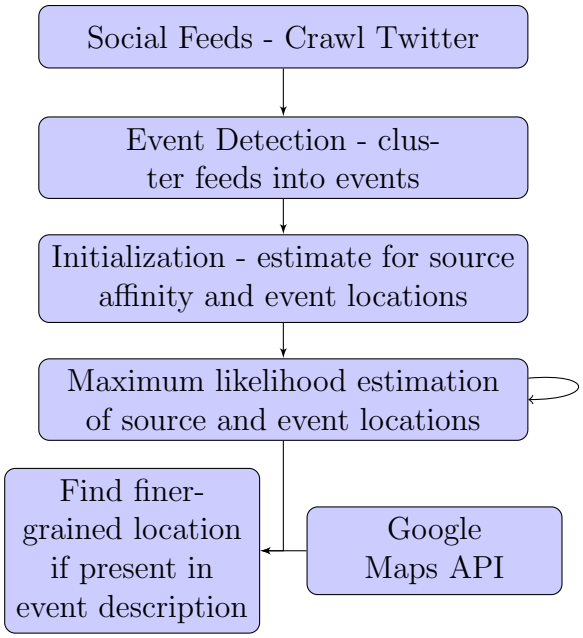


Figure 4.1: Overall architecture of the joint event and source localization system

- *Joint location estimation:* We iteratively perform a joint maximum likelihood estimation of locations associated with events and sources. Starting with the initial (noisy) location estimates, the algorithm iteratively attempts to infer source location affinities from estimated locations of events they report, as well as event locations from estimated location affinities of reporting sources.

- *Location refinement:* Once a coarse-grained location estimate is obtained for an event (e.g., at the level of a city and a country), we obtain a finer-grained estimate by considering street address data, if found in event descriptions. This step considers the approach described in the previous chapter.

4.2.1 Location Initialization

Our joint source and event location estimation requires that some source location affinities be approximately known, and some events be approximately localized in the beginning. It uses this initial knowledge to complete and enhance location estimates.

Given the tweet clusters computed by StoryLine, we first initialize source location affinities to the advertised source location, where known. This approach assumes that sources tend to report local events (possibly among other events). For example, a person who resides in San Francisco is more likely to tweet about a road accident causing congestion in the Bay

Area compared to a person in New York city. Hence, we identify sources of tweets in the clusters and search source profiles for location information.

To support our assumption of the source location affinity, we study the frequency that a source tweets an event with the same location as the source states in his/her profile using the datasets crawled from Twitter. We randomly select 20 sources with locations indicated in their profile, and manually detect the locations of their tweeted events. We then classify their tweets into three categories based on the locations of their corresponding events: 1. the event's location is the same as the source's profile location (tweets fall into this category are called the *Matched Tweets*), 2. the location is different from the source's profile location (tweets here are called the *Unmatched Tweets*), and 3. the location is unidentifiable (tweets here are called *Unidentified Tweets*). We plot the number of categorized tweets for each source in Figure 4.2. From the figure, we observe that 14 (of the 20) sources only tweet events in their near proximity. Only 2 sources have tweeted about events with locations not matching the ones in their profile. The inner plot in Figure 4.2 shows the overall percentage that a tweet falls into each category. In this case, 82% of the tweets are co-located with their sources. The results shown in Figure 4.2 support our assumption that sources tend to tweet something in their near proximity.

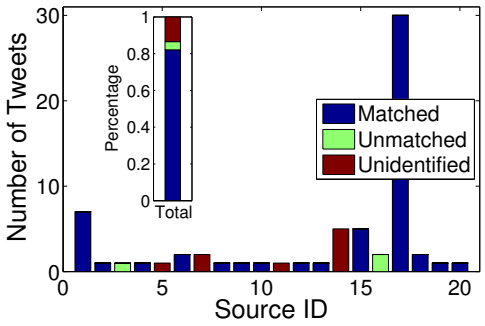


Figure 4.2: The number of tweets in each of the three categories for the 20 sources. The inner figure shows the overall percentage that a tweet falls into each category.

In our dataset, around 90% of the sources have location information in their profiles, which is usually at the city level. We propagate this location label to all tweets of this source. As a second source of location information, we scan tweet text for mention of known geographic landmarks (e.g., the "Golden Gate Bridge") and associate a location label with these tweets accordingly. (Hence, a tweet can have multiple location labels from different indicators.) We use labels formatted hierarchically from the more general to the more specific. That is to say, each label is an ordered tuple containing the following fields: Country, State, City, Street, Number, in that order. Not all elements may be present in the tuple. For example,

a tweet might refer to a street address but omit the city and state. During evaluation using real-world data sets we randomly remove some of the source profiles to see the effect on localization accuracy for both the events and sources.

To initialize event location labels, we tally all labels of all tweets in each tweet cluster and compute the majority label at each level of the address hierarchy (by voting on the majority label in each field of the address tuple). Since data are noisy, inconsistencies may arise. For example, we may obtain "USA", "NY", "New York", and "Santa Monica Blvd" as majority labels for country, state, city, and street, respectively, in some cluster. To remove noise, starting from the first (most general) field, we keep the majority label as long as it is consistent with its ancestors. To test that a label is consistent with its ancestors, we use GoogleMaps API [37] to geotag the location based the subset of fields traversed so far. If GoogleMaps succeeds, we move to the next field. We stop once GoogleMaps fails. In the above example, since there is no "Santa Monica Blvd" in New York, the most detailed address that GoogleMaps succeed at geotagging will be "USA", "NY", "New York". This address will be used as the initial event address. We call it the *longest consistent hierarchical location label*. Hence, different events may be localized to a different degree of granularity during initialization.

4.2.2 Joint Location Estimation

In order to estimate location labels associated with events we make use of an expectation maximization approach. Consider a Twitter data set, where M sources have generated tweets that fall into N clusters and collectively mention P landmarks. We compose a directed graph, as shown in Figure 4.3, where the source nodes, denoted by $S_1, ..., S_M$, and landmark nodes, denoted by $L_1, ..., L_P$, are connected to the event nodes, denoted by $E_1, ..., E_N$. A source node, S_i, is connected to an event node, E_j (denoted as $S_i \rightarrow E_j$), if source S_i generates a tweet that falls in the cluster of event E_j. A landmark node, L_i, is connected to event node, E_j (denoted as $L_i \rightarrow E_j$), if the landmark, L_i, was mentioned in a tweet that falls in the cluster of event E_j. The location of event E_j is denoted as $Loc(E_j)$.

Our problem is to estimate (or refine) the longest consistent hierarchical location label for each event. To do so, we would like to devise a maximum likelihood estimation algorithm to jointly estimate the field values at each level of the address hierarchy for both sources and events, subject to the consistency constraints with ancestor fields. In practice, checking field consistency constraints (i.e., that the fields, taken together, form a valid address) entails a lot of queries to GoogleMaps during the iterations, which is subject to rate limiting. Hence, as an approximation, we relax these constraints during the iterations of the maximum likelihood

algorithm and check them only upon completion. This allows is to estimate the fields at each level of the address hierarchy separately and independently from estimation of fields at other levels.

Consider the estimation of a single field in the hierarchy (e.g., "city"). Let each event be associated with a latent variable corresponding to the city being identified. The vector of all such latent variables for all events is called Z. These variables are initialized as described in the preceding subsection. Moreover, let us define $a_{i,k}$ as the probability that source S_i tweets about an event, given that the event is at location k. Conversely, the probability that an event is at location k, given it was mentioned by source S_i is denoted as $t_{i,k}$. Let us also define $l_{i,k}$ as the probability that a landmark L_i be mentioned in a tweet about an event, given that the event is at location k. The probability that an event is at location k, given that landmark L_i was mentioned in a tweet about this event is $f_{i,k}$.

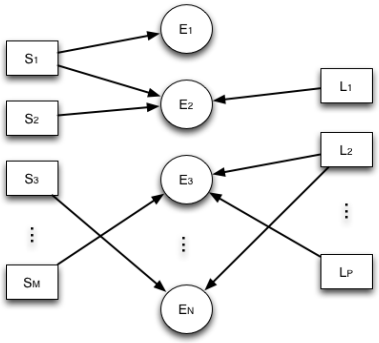


Figure 4.3: Graph network of sources, events and landmarks

As described in [45]

$$a_{i,k} = P(S_i \rightarrow E_j | Loc(E_j) = k) \tag{4.1a}$$

$$t_{i,k} = P(Loc(E_j) = k | S_i \rightarrow E_j) \tag{4.1b}$$

$$l_{i,k} = P(L_i \rightarrow E_j | Loc(E_j) = k) \tag{4.1c}$$

$$f_{i,k} = P(Loc(E_j) = k | L_i \rightarrow E_j) \tag{4.1d}$$

With help of approach mentioned in [45] we jointly estimate (i) the latent variable vector Z for the city level location labels of events, and (ii) the affinity of input sources using the parameters represented by vectors $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$, where $\theta_k = (a_{1,k}, a_{2,k}, \ldots, a_{M,k}, f_{1,k}, f_{2,k}, \ldots, f_{P,k})$. The values of $t_{i,k}$, $l_{i,k}$ can be computed using the estimated parameters $a_{i,k}$, $f_{i,k}$ by applying the Bayes' theorem. The maximum likelihood estimator finds the values of the unknown parameters θ that maximize the probability of observed input X, the graph network as

illustrated in Figure 4.3. The requirement is to maximize $P(X|\theta)$ using the Expectation-Maximization (EM) algorithm that starts with some initial random values for θ and iteratively updates the parameters.

There are in total three steps derived to calculate the labels and input source parameters as follows. We only give the result of each step. For the details of derivation, please refer to [45]. The time efficiency convergence property of the EM algorithm is the same as described in [45]:

- Defining the log-likelihood function, $\ell(\theta; X, z)$

$$\begin{aligned}
&\ell(\theta; X, z) = P(X, z|\theta) \\
&= \prod_{j=1}^{N} \left\{ \sum_{k=1}^{K} \left(\prod_{i=1}^{M} a_{i,k}^{X_{i,j}} (1 - a_{i,k})^{(1-X_{i,j})} \times d_k \times z_j^k \right) \times \right. \\
&\left. \left(\prod_{i=M+1}^{M+P} f_{i,k}^{X_{i,j}} (1 - f_{i,k})^{(1-X_{i,j})} \times d_k \times z_j^k \right) \right\}
\end{aligned} \tag{4.2}$$

  where $d_k = P(z_k = k)$.

- Deriving the E-step, $Q(\theta|\theta^{(t)}) = E_{z|X,\theta_t}\{\log P(X, z|\theta)\}$

$$\begin{aligned}
&Z(t, j, k) = p(z_j = k|X_j, \theta^t) \\
&= \frac{A(t,j,k) \times d_k}{A(\sum_{k=1}^{K} t,j,k) \times d_k}
\end{aligned} \tag{4.3}$$

  where $A(t, j, k)$ is defined as:

$$\begin{aligned}
A(t, j, k) \quad &= \left(\prod_{i=1}^{M} a_{i,k}^{(t)X_{i,j}} (1 - a_{i,k}^{(t)})^{(1-X_{i,j})} \right) \times \\
&\left(\prod_{i=M+1}^{M+P} f_{i,k}^{(t)X_{i,j}} (1 - f_{i,k}^{(t)})^{(1-X_{i,j})} \right)
\end{aligned} \tag{4.4}$$

- The M-step, $\theta_{t+1} = \arg\max_\theta Q(\theta|\theta^{(t)})$

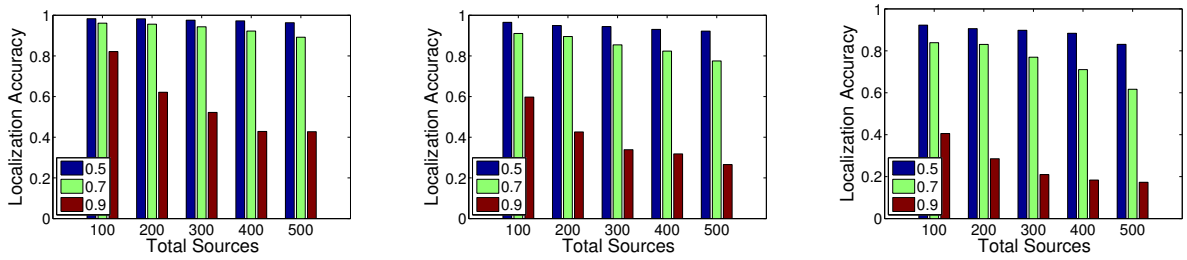
$$a_{i,k}^{(t+1)} = \frac{\sum_{j \in SE_j} Z(t, j, k)}{\sum_{j=1}^{N} Z(t, j, k)}, f_{i,k}^{(t+1)} = \frac{\sum_{j \in LE_j} Z(t, j, k)}{\sum_{j=1}^{N} Z(t, j, k)} \tag{4.5}$$

  where SE_j denotes all the events that the source connects, and similarly we define LE_j.

The algorithm terminates when the difference of estimation parameter between consecutive iterations becomes insignificant and the city level class label for E_j is assigned on the basis of largest value of $Z(t, j, k)$ for $k = 1, 2, \ldots, K$.

The above algorithm is repeated for each field in the address. Note that, it also generates the probability of different values of each field, $P(z_j = k)$, in each iteration. Values associated with a probability that are below a configurable reliability threshold are removed. The remaining values are deemed reliable. Once all reliable values have been computed, we repeat the estimation of the longest consistent hierarchical location label as was done in the initialization. Specifically, starting from the most general field (i.e., country), we keep the computed field value as long as it is consistent with its ancestors. We use GoogleMaps API to test if a field is consistent with its ancestors (i.e., generates a valid address). We stop once GoogleMaps fails. The algorithm generates a location estimate at different degrees of granularity for different events. For example, street-level events such as car accidents might be localized to the nearest intersection, whereas city-level events (such as a carnival) might end with the city they are in (since conflicting street-level information will likely be reported, if any, thereby reducing the reliability of street-level estimates computed by the maximum likelihood estimator).

4.3 EVALUATION



(a) 4 city location labels (b) 5 city location labels (c) 6 city location labels

Figure 4.4: Varying input sources from 100 to 500 and missing fraction of location labels from 0.5 to 0.9 for constant number of events.

In this section we evaluate the performance of our localization framework by considering real-world datasets collected during various events. We also carry out extensive simulation experiments to see the variation in performance as we change different parameters for the input. For comparing the accuracy of localization of events and sources we select two baseline methods. The Baseline 1 method is the one described in [46] where the city level location label for an event is determined by using the profile information of the sources linked to

the event. The final location label is determined by taking a maximum vote among all the available city location labels from the sources. The Baseline 2 method is the approach as mentioned in [42]. This method as described by the authors makes use of a probabilistic distribution over the city location labels for the words used in the content of tweets. The basic assumption for building this model is the independence of the words occuring in the events. Thus if there are K possible location labels then the probability of an event E_j belonging to a particular location can be formulated as:

$$Location(E_j) = argmax_{k \in K} P(k|E_j) \tag{4.6a}$$

$$= argmax_{k \in K} \frac{P(E_j|k)P(k)}{P(E_j)} \tag{4.6b}$$

$$= argmax_{k \in K} P(E_j|k)P(k) \tag{4.6c}$$

We can reformulate the above described equation as follows:

$$Location(E_j) = argmax_{k \in K} P(k) \prod\nolimits_{w_i \in V} P(w_i|k) \tag{4.7a}$$

$$P(w_i|k) = \frac{count(w_i, k) + 1}{\sum_{w_i \in V} (count(w_i, k) + 1)} \tag{4.7b}$$

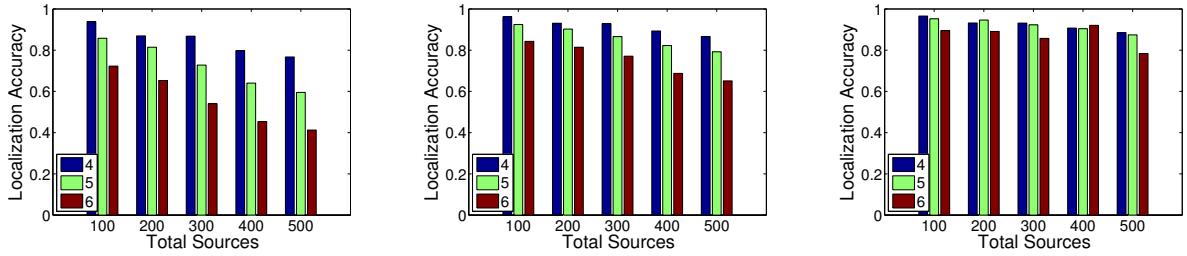
In this equation w_i are the set of words present in the event descriptions (tweets) and k is a particular location type from K. The final location label is assigned to an event based on the location which generates the maximum probability over the word distribution. For the simulation study we use only the first baseline method for comparison since the generation of word tokens following a distribution over locations was not supported by the simulator. On the other hand we use both the baseline methods for real-world datasets.

4.3.1 Simulation Study

We conducted experiments using simulated data to see the variations in performance when number of city location labels and number of input sources are varied. The simulator was built using Matlab to generate a random number of sources and events with ground truth labels for city locations. For the evaluation metric we determine the average accuracy of localization in two scenarios: (i) when input sources and missing fraction of location labels are varied with constant number of events, and (ii) when input sources and number of events

are varied with constant number of missing fraction of location labels.

- **Constant number of events** - In this part we ran the simulator to generate 1000 events and varied the number of input sources from 100 to 500. The number of location labels was set as $K = 4, 5, 6$. The links between the input sources and events were generated using a uniform random distribution. The ground truth labels of the events is also generated during the process and for the purpose of evaluation we randomly remove 50%, 70%. and 90% of the ground truth location labels. The input source parameters are randomly initialized and the events for which ground truth location label was not removed are not updated during the estimation step. When the algorithm converges, we compare the estimated location labels with the original ground truth location labels and obtain the accuracy for localization. This process is repeated 30 times for each number of source and missing fraction to smooth the noise present during random generation of the links between input sources and events. Finally the average accuracy over all 30 iterations is calculated. Figure 4.4 shows the bar plot for each location label when number of input sources vary in x-axis. For each source we report the average accuracy for three missing location label fractions shown in different colors. It can be clearly seen from the figure that as the fraction of missing location labels increased the average accuracy for localization dropped by a large percentage margin. This can be easily attributed to the fact that discovery of new location labels is dependent on input source parameters that are computed using initial available location labels.

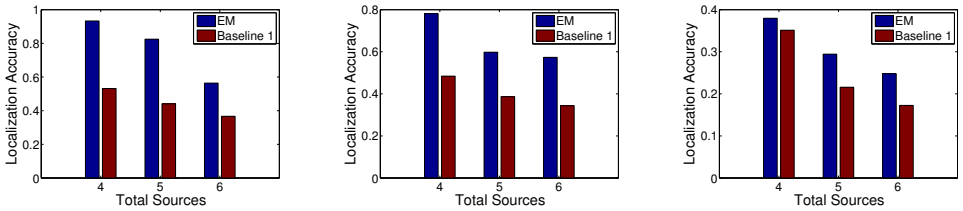


(a) 2000 events (b) 3000 events (c) 4000 events

Figure 4.5: Varying input sources from 100 to 500 and number of events from 2000 to 4000 for constant number of missing fraction of labels (90%).

- **Constant number of missing fraction of location labels** - In this part we ran the simulator to generate events varying from 2000 to 4000, number of input sources from 100 to 500. The number of city level location label was set as 4, 5, 6 and the missing fraction as 90%. The links between sources and events were again generated using the same uniform random distribution. The EM algorithm for each case was run

for 30 iterations to find the average accuracy for localization. The results are shown in Figure 4.5 where each sub-plot corresponds to the number of events, the x-axis for the number of sources, and y-axis for the average accuracy respectively. The bars in different colors correspond to the number of location labels and 90% of the labels were removed randomly during each run to report the accuracy. It can be clearly seen from the figure that as the number of events increased for the same number of sources and missing fraction of location labels the performance of the algorithm improved significantly.



(a) 50% source profile removed (b) 70% source profile removed (c) 90% source profile removed

Figure 4.6: Comparison for simulation study between EM framework and Baseline 1 method for city level location labels varying 4 to 6.

- **Comparison with Baseline 1 method** - For the simulation study we use the first baseline approach as described above in order to compare with our localization framework using EM for estimating city level location labels. We generated 2000 events and 50 sources with location affinity from the given type of location labels. The links between sources and events were generated according to the source parameters using the affinity. For both the methods we removed $X\%$ of source profile information, where X was set as 50, 70, and 90, in order to localize the events using the respective algorithms. The first baseline method estimated the location labels using a maximum vote scheme among the available location labels from source profile information and then these events with estimated location labels from baseline method were used as input to the EM algorithm. We compared the estimated locations by both the methods with the actual ground truth location labels in order to determine the accuracy for localization of events and sources. This process was repeated 30 times and the comparison between both the methods for average accuracy is shown in Figure 4.6. The x-axis represents the number of location labels and the y-axis is the corresponding localization accuracy. As seen from the plots the average localization accuracy for EM method is always better than the first baseline method for any percentage of missing source profiles. The best localization accuracy (50% source profile removed, 4 location labels) for EM method is 93.3% compared to that of first baseline method which is

53.16%.

4.3.2 Real-world datasets

In this part we perform the localization of events detected from real world data collected from different locations using event related keywords on the crawler service. The focus is specifically to localize the events among city level labels using the EM algorithm followed by a finer granularity localization if possible. To ensure detection of events over different topics we selected three data sets : (i) the 2015 North American Blizzard which had a severe impact in Boston city from January 27-30, 2015, (ii) 2014 FIFA World Cup Semi Final 2 in Sao Paulo city and Final in Rio de Janeiro city from July 9-13, 2014, and (iii) One of the strongest tropical cyclone ever recorded - Typhoon Yolanda which affected the cities Cebu and Iloila in Philippines from November 9-11, 2013. The raw feeds corresponding to these datasets where extracted only during and after the events and provided as input to our localization framework. The events are detected using the information gain approach using refinement followed by localization to highest possible granularity. To assign the initial ground truth location labels at a city level we manually looked at the events. There were 5 different location labels corresponding to each city mentioned in the three datasets selected for the evaluation. The landmarks associated with the events were obtained by looking at the top 50 single keywords ranked by information gain during event detection. The choice of landmarks from the ranked list required a manual inspection as well.

- **Comparison with Baseline 1 method** - We first study the localization accuracy at city level for events using the baseline 1 method and compare it with that of EM method. Briefly we remove $X\%$ of source profile information and estimate the location at city level for the events using a maximum voting scheme from all the linked sources with available profile. This gives the set of localized events according to baseline 1 method. Next we take this localized events list as input for the EM method along with the sources and landmarks and get a new set a localized events along with source affinities. We then compare the localized event labels at city level with the actual ground truth location labels to find the accuracy. We repeat this process for 100 iterations for each missing percentage of source profile information. The average accuracy comparison for both the methods is reported in table 4.1. From this table we can see that for a realistic Twitter dataset with only 25% source profile information available we are able to localize events along with sources with more than 10% accuracy compared to baseline 1 method. Finally we use the grammar rule as described in [46] to extract

finer level locations if present. The longest hierarchical location label is obtained with the help of GoogleMaps.

Table 4.1: Average localization accuracy for EM and Baseline 1

% source profile removed	**EM**	**Baseline 1**
20%	81.27%	78.5%
30%	69.96%	66.07%
50%	60.89%	54.41%
70%	42.63%	32.15%
90%	15.13%	10.54%

- **Comparison with Baseline 2 method** - The baseline 2 method as described in the beginning of evaluation section makes use of a probabilistic distribution over the city location labels for the words describing an event. Thus in order to compare the localization accuracy we randomly remove a fraction (varying from 0.2 to 0.7) of ground truth location labels. The events with location labels belong to training set and the rest to test set. To build the model from the training set we perform a preprocessing step to remove all the stop words belonging to English language and consider only those words that have a string length above (i) 5, and (ii) 6. The location labels are estimated as described in equation 4.7a and compared with the available ground truth location labels. We also provide the same list of events with partial location lables along with linked sources as input to the EM algorithm with completely random initialization for the source parameters(θ). For both the methods we performed 100 iterations for each missing fraction value of location labels to find the average accuracy over the entire run. The comparison between the two methods is reported in table 4.2. On average the EM method outperforms the selected baseline method for all the cases.

Table 4.2: Average accuracy with partially localized events for EM and Baseline 2

Fraction non-localized	**EM**	**Baseline 2**	
		word length ≥ 5	word length ≥ 6
0.2	50.14%	41.85%	38.92%
0.3	48.63%	40.8%	38.21%
0.4	47.72%	40.24%	38.14%
0.5	46.59%	40.13%	37.86%
0.6	46.05%	39.53%	37.43%
0.7	44.89%	38.88%	36.44%

4.4 SUMMARY

This chapter presents an approach for *joint* localization of both sources and events in social networks. Our algorithm is not based on any language model once events are detected from the text streams from social networks. The intuition is that sources tend to post tweets corresponding to events happened in near proximity, which is verified in this paper using the real datasets collected from Twitter. Our approach is built on top of the Expectation-Maximization algorithm to jointly estimate the source location as well as the event location. Evaluation results, from both extensive simulations and real-world datasets of Twitter, show that our proposed algorithm outperforms the state-of-the-art solutions.

CHAPTER 5: EVENT DETECTION WITH INSTAGRAM

This chapter describes methods to exploit picture-oriented social networks to localize urban events. We choose picture-oriented networks because taking a picture requires physical proximity, thereby revealing the location of the photographed event. Furthermore, most modern cell phones are equipped with GPS, making picture location, and time metadata commonly available. We consider Instagram as the social network of choice and limit ourselves to urban events (noting that the majority of the world population lives in cities). This chapter introduces a new adaptive localization algorithm that does not require the user to specify manually tunable parameters. We evaluate the performance of the algorithm for various real-world datasets, comparing it against a few baseline methods.

5.1 OVERVIEW

This chapter investigates social networks that carry pictorial information as a means to localize urban events of interest in time and in space. In turn, the ability to localize events gives rise to new search services that allow users to view important events matching a category of interest on a map, and remotely experience those events through the lenses of eye-witnesses. Since the majority of the world population lives in cities [47], we restrict ourselves to *urban* events.

The work is made possible thanks to the proliferation of picture-taking devices (e.g., over 2 billion smart phone users at present [48]) and picture-sharing media that offer a real-time view of ongoing events. We consider Instagram [49] as our social medium of choice. Instagram is a real-time picture sharing network, whose popularity has increased dramatically in recent years. At the time of writing, Instagram has more than 500 million users, who collectively upload 80 million pictures a day [50]. This is up from 400 million, 300 million, 150 million, and 30 million users in 2015, 2014, 2013, and 2012, respectively. Based on an experiment from a sample of images we collected that are publicly viewable, more than 15% contain location metadata, making it meaningful (given the large total volume) to consider Instagram as a tool for *localization*.

One should mention, at this point, that a variety of other social networks also carry location information, pictures, or references to geographic events, leading to the question: *why Instagram?* There are four reasons behind our particular choice of social network in this paper. First, many networks, such as Facebook or Nextdoor, consider their content private to the user or group. As such, the content is not accessible for general browsing. In contrast,

by default (unless explicitly designated otherwise), Instagram content is publicly available via resellers, such as `picodash.com`, who offer an interface that allows anyone to search for Instagram images (using tags or keywords). Hence, content access for a general event localization service becomes feasible. Second, unlike *text-based* social networks with publicly available content, such as Twitter, Instagram features a content type that generally requires physical proximity to the event. While it is possible to tweet about a volcano from across the globe, it is harder to take pictures of it without physical proximity. Hence, the spatial distribution of Instagram content has a better correlation with actual event *locations*. Third, unlike other picture-based social networks, such as Flickr, Instagram content is much more *real-time*. For example, Flickr is often used to share art photography, scenic landscapes, and other inspiring visuals. In contrast, Instagram is used to capture the moment, from a meal being consumed to a local event of interest. Thus, the temporal distribution of Instagram images offers a better reflection of event *times*. Finally, some social networks, such as Foursquare, are explicitly location-centric, offering sign-ins to a set of participating locations and associating all user posts with the sign-in location. Unfortunately, since the set of participating locations is discrete (e.g., coffee shops, landmarks, etc), an event that does not occur in the neighborhood of a sign-in location is harder to localize. In contrast, Instagram can offer coverage anywhere that a person with a camera is present.

Localizing user-specified types of events based on Instagram pictures calls for a capability to associate the pictures with specific event keywords. Fortunately, Instagram users frequently associate customized metadata with images to identify what an image is of. Specifically, Instagram allows users to *tag* images they upload (in addition to associating a spatial location based on the GPS). This makes it possible to search Instagram images for those matching event-specific keywords.

The above suggests that a text query for an event such as "*#JapanEarthquake*" or "*#Chicago-Marathon*" can retrieve pictures with annotations matching the query, from which the corresponding event can be localized. The manner in which pictures matching a set of keywords are identified is not the challenge addressed in this paper (It constitutes a standard database indexing problem). The challenge we address below is the way one might identify and localize events in space and in time *given* the set of retrieved pictures matching a query. While several prior efforts used Instagram for localization, as we elaborate in related work, ours is distinguished in that we try to design an intuitive algorithm to learn the parameters based on the data being processed. As such there is no prior data based learning which makes our algorithm more adaptive and robust to real time developing events.

The event localization solution we propose is based on a technique that uses the distribution of pictures in the time domain along with a spatial range to observe the events to

generate clusters followed by a false alarm elimination. We eliminate any manual inspection for parameter settings with the help of a self-evaluation scoring metric. In order to help us design an algorithm, we propose a set of requirements and hypotheses that guide us in the derivation. The proposed requirements define the scope of applicability of the design while the hypotheses are assumptions on content features that are verified during the evaluation with the help of collected datasets.

The rest of this chapter is organized as follows. The assumptions for deriving the algorithm are described in Section 5.2. We present the design of our system in Section 5.3. The collection of datasets, verification of assumptions, and algorithm performance results are discussed in Section 5.4. We then describe the state of the art and related work in Section 5.5. Finally, we present conclusions from our work in Section 5.6.

5.2 ASSUMPTIONS

In this section, we describe a set of assumptions that we use to design our localization algorithm. We divide our assumptions into two categories. The first category, called *requirements*, list application conditions for which the algorithm is most suited. They may or may not always be borne out by data, and as such can be thought of as targets for which algorithm design is optimized, as opposed to truth about the physical world. The second category, called data *hypotheses*, list properties we expect to see in input data. These properties are validated using experiments with real data sets. The first category allows us to set ground rules on top of which to design our system. We do not expect these rules to always be true, and instead evaluate the degree to which algorithm performance departs from ideal in the real world (where such assumptions may or may not hold). For the second category, the stated assumptions are empirically derived from data and exploited to make sure that algorithm design works for real datasets.

5.2.1 Requirements

- Assumption 1: We need to uniquely distinguish only one event occurrence at a specific point location (latitude, longitude) during a particular time interval. Hence, if more than one event are exactly co-located in space and time, our algorithm thinks of them as one.

- Assumption 2: Two or more independent events can take place (at different locations) during the same time interval or have some overlap in their respective time intervals.

- Assumption 3: The users (sensors) generating the signals (pictures) are conditionally independent of each other (conditioned on event occurrence). This means they respond independently to the event as opposed to responding to one another. Hence, for example, we filter out re-posted images (and subsequent images by the same user) as they do not offer further independent evidence. We further assume that original valid responses to an event (excluding noise and false positives) will generally occur within a range R, which determines the maximum distance at which an event can be directly observed.

- Assumption 4: True events generate response by multiple users. Hence, if there is only one user producing (any number of) images of the event, we mark it as a false alarm.

5.2.2 Hypotheses

- Assumption 5: It is possible for users to post pictures of an event from a (*false*) location different from the actual place of occurrence (or *true* location) of the event. For example, multiple users watching a sports match at the stadium and those watching the same match in a group at a bar in a different city can post pictures tagged by the same event.

- Assumption 6: The number of users (sensors) generating signals (pictures) from the *true* (actual) location of an event are always more than those located at any *one* of the *false* locations for the same event. However, it is possible that the number of users from the *true* location of an event are less than the total from all *false* locations. For example, the number of images posted from bars and clubs tagged by (and thus commenting on) a specific sports event can be globally larger than the number of images taken at the stadium where the event occurs. However, the number of users posting such images from any one bar or club is smaller than those posting from the stadium.

- Assumption 7: Events generally belong to one of two major categories; *expected* and *unexpected*. The distribution of the signals (pictures) generated in the time domain follows a different pattern in the two cases:

    1. *Expected Events* : Events that are scheduled well ahead of time, such as music concerts, generate attention from users much before the event begins, and reach a peak in observations after the event has actually started (and then gradually fade over time).

2. *Unexpected Events* : Events that are not scheduled ahead of their occurrence, such as an earthquakes, generate no posts prior to the event, followed by a sharp increase in observations at event occurrence leading to a peak during the event, then gradually decrease over time.

Since the very first pictures that tag the event have a different relation to event start time in both cases, we find it more robust to consider the *first quartile* of picture distribution as the estimated start. Similarly, we view the *third quartile* as the estimated end time.

- Assumption 8: There is similarity in the popular tags used to describe an event at both its *true* and *false* locations. This, together with Assumption 6 offers a way to eliminate false positives (among similarly tagged picture clusters).

5.3 SYSTEM DESIGN

The goal of our work is to identify the locations of real-world events in time and space based on the data shared by users on the Instagram social network. We derive an algorithm that is capable of detecting and localizing the events in physical space with the help of the assumptions described in the earlier section.

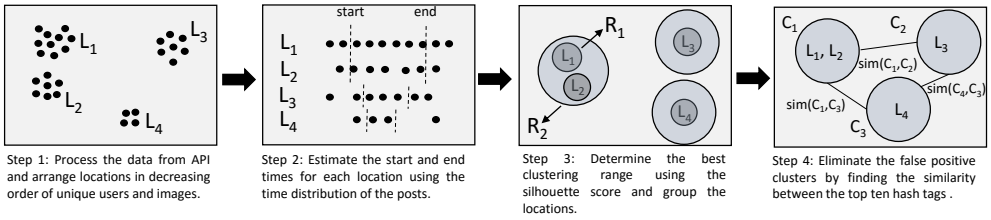


Figure 5.1: Visual representation of the localization algoirthm

5.3.1 Problem Statement

Each picture generated by a user is a tuple of the format (l, t, u, tag), where l is the Instagram location, t is the image post time, u is the user id, and tag is the set of tags. Note that, Instagram does not use the original GPS coordinates for an uploaded image, but rather gives the user a list of identifiers to choose from (e.g., street addresses or landmarks) and associates the image with the user-chosen identifier. For a given time interval of interest, let K denote the number of unique locations (i.e., location identifiers) referred to in images

timestamped within that time interval. Let us further denote those locations by l_1, l_2, ..., l_K. The goal of the localization algorithm is to (i) group these locations into clusters that correspond to true spatio-temporally contiguous events occuring in the area covered by their respective cluster, and (ii) eliminate false positives (i.e., clusters not corresponding to events at the area covered by the cluster). Here, the area covered by a cluster refers to the set of unique Instagram locations included in this cluster.

As mentioned in Assumption 3, events are associated with a radius, R, which determines the maximum range at which an event can be locally observed. Hence, to do the above clustering, we need to find the appropriate radius R for the categry of events we are interested in. Unfortunately, the radius, R, is not known. Hence, the algorithm has to determine its value automatically in an unsupervised fashion. We do so with the help of a metric called, *silhouette score*, which determines the quality of clustering. We use it to comapre quality of spacial clustering of unique picture locations, when clusters are limited to different values of R, until we find the best R. Given a particular clustering output, this computation is a three step process as follows:

- Cohesion Factor (a_i): For the i^{th} data point (i.e., Instagram location l_i), we find the average (Euclidean) distance to all other data points (locations) within the same cluster.

- Separation Factor (b_i): For the i^{th} data point (i.e., Instagram location l_i), we find the average (Euclidean) distance from all the data points (locations) of another cluster to which it does not belong. Then we take the minimum of the average distances from all the clusters.

- Silhouette Coefficient: Finally, we assign the score to the i^{th} data point using the equation $s_i = \dfrac{(b_i - a_i)}{max(a_i, b_i)}$.

The silhouette coefficient for any data point is in the range $[-1, 1]$. The ideal best case is when $a_i = 0$ for which the maximum value of 1 is attained. For our algorithm, we vary the value of R between 0.25 and 30 miles, and keep the clustering that maximizes the silhouette score. Events in the current time interval are then represented by the resultig cluster set, $\xi = \{E_1, E_2, \ldots, E_m\}$, where $m \leq K$.

Finally, we eliminate false positives from the aforementioned set. False positives arise because large events (e.g., football games) may be watched by groups of people at remote locations, such as parties and bars, leading to additional found clusters (we call *echo clusters*) at those locations. However, tags of images at those echo clusters bear much similarity to

tags of images at the real event location. This insight is used to remove the smaller echo clusters. Below, we detail the entire algorithm.

5.3.2 Data Preprocessing

Our system follows the feed subscription model as opposed to the search query model. This allows the user to monitor the events of different types on a timely basis in near real time. The user can view any posts from the past or create a new subscription with the help of a "tag" keyword. The subscribed tag is then queried using the web service `picodash.com`, where Instagram images are returned starting with the most recent posts along with metadata information. In order to avoid spamming, we make the web requests at an interval of one hour. Every image has a tag ID in the metadata that allows our crawler service to identify the tag ID at which the call needs to be stopped for the current interval. The retrieved images along with metadata are then sent to the next component for further processing.

The image posts obtained from the crawler service are processed in this step to remove any noise present. Every image has a metadata component, which contains several fields. We make use of only *image id*, *image url*, *user id*, *created time*, *tags*, and *location*. We filter out any image for which the *location* field is empty. Next we make use of *created time* of the image post to divide the data feed into intervals. This step is repeated for every API call and the image is added to the corresponding interval. Any updated interval is then analyzed by the localization algorithm to detect events.

5.3.3 Localization Algorithm

Consider the set of all unique locations associated with pictures in the current time interval. Let these locations be denoted by l_1,l_2, ..., l_K. For each location l_k, $1 \leq k \leq K$, let us compose the sets $\mathcal{T}_k$, $\mathcal{U}_k$ and $\mathcal{G}_k$, denoting the set of unique timestamps, users, and tags, respectively, that are associated with images from that location. Our algorithm is described below:

1. In the current time interval, we arrange locations l_k in descending order by size $|\mathcal{U}_k|$. (Ties are broken arbitrarily.) Let this be the ordered list of locations. We make use of both Assumptions 1 and 6 for this step.

2. Process the locations from the ordered list one at a time. This refers to Step 1 from Figure 5.1. For a selected location, use the sets $\mathcal{T}_k$, $\mathcal{U}_k$ and $\mathcal{G}_k$ from the current interval

padded by three past days. Using Assumption 7, we find the estimated start and end time (i.e., the respective quartiles) of the distribution of timestamps in set $\mathcal{T}_k$, as indicated in Step 2 from Figure 5.1. There are 4 possible cases:

- Both estimated times are outside the current interval. This means that this event occurred in one of the previous intervals. Discard this location and move to next location.
- Both estimated time are inside the current interval. Use the location for analysis with data within estimated time range.
- One of the estimated time is inside the current interval. Use the location for analysis with data between boundary of interval and the estimated time.
- Both the estimated times capture the boundaries of current interval. Use the location for analysis with all data within the interval boundary.

3. Let each suviving location l_k, from the current interval, have t_{start_k} as start time and t_{end_k} as end time. If this is the very first location in the ordered list, then form a new cluster E representing an event. Initialize the set of found clusters, ξ to $\{E\}$. Let l_k be the prime location, l^E_{prime}, for this newly formed cluster indicating the most probable value.

4. If the l_k is not the first location from the ordered list, then scan through each event cluster, E, from ξ for two conditions:

   - Interval $[t_{start\_k},\ t_{end\_k}]$ has overlap with interval $[t^E_{start\_prime},\ t^E_{end\_prime}]$ for the prime location, l^E_{prime}, of cluster, E.
   - Location l_k is within R miles of distance from the prime location l^E_{prime} (using Assumption 3).

   If both conditions are satisfied, then l_k goes into cluster E. Otherwise, we form a a new cluster with l_k as the prime location.

5. We repeat the Step (iv) for varying values of R, as indicated earlier, and compute the silhouette score in each case as indicated in Step 3 from Figure 5.1. Finally, we select the range R_{sel} with the maximum score and keep the resulting clustering.

6. Once all the locations from the ordered list are analyzed, we eliminate those clusters that have only a single user inside according to Assumption 4.

7. In order to eliminate the false alarm clusters, we use Assumption 8 to compute the similarity in the vectors formed by considering the top 10 commonly used tags from each cluster as shown in Step 4 from Figure 5.1. We process the clusters in the order they were formed and check for similar clusters from the remaining subset. The similarity threshold setting is described later in the text.

8. With false positives removed, the estimated location of a remaining event is set to the weighted average of the locations l inside the event cluster. The weights are derived using the fraction of images posted from a location compared to images present inside the entire cluster.

For elimination of false positive clusters, we first need to identify the type of event. Events can be broadly classified into two categories: (i) Single Entity (SE), and (ii) Multiple Entities (ME). For example, Taylor Swift being a single entity (person) can perform only at one valid location during a particular time interval. If there are several clusters identified for a SE event in the same interval then only one of them can be a true positive, while the remaining are false alarm clusters. However, a ME event such as marathon or tornado can occur at several locations during the same interval. Based on the clusters generated, we looked at a few random samples and noticed that the size of the main cluster in case of SE events was always significantly larger compared to the false alarms. At the same time, the clusters in case of ME events were comparable in size.

Table 5.1: Cluster Statistics For Events With Single Entity

Event	**Date**	**City**	**Top 5 cluster size**
Taylor Swift	09/29/15	St. Louis, USA	[169, 27, 6, 5, 1]
Taylor Swift	10/03/15	Toronto, Canada	[940, 24, 6, 6, 5]
Maroon V	06/12/15	Milan, Italy	[134, 13, 8, 5, 5]
Maroon V	09/17/15	Manila, Philippines	[181, 11, 9]

Table 5.2: Cluster Statistics For Events With Multiple Entites

Event	**Date**	**Cities**	**Top 5 cluster size**
Marathon	10/18/15	Columbus, OH, USA Detroit, MI, USA Toronto, Canada	[266,172,112,79,74]
Marathon	10/25/15	Washington DC, USA Frankfurt, Germany Jakarta, Indonesia	[153,134,88,58,46]

The cluster statistics for a random sample of events are provided in table 5.1 for SE and table 5.2 for ME. The last column in the tables corresponds to the top five clusters by size

(number of data points) for each event type on a particular date. It can be clearly observed that in case of SE events the top most cluster by size is extremely dense in comparison to other clusters, whereas in case of ME events the true clusters don't have a huge difference. This can be attributed to the fact ME events attract the attention of people from all the ground truth locations at more or less the same rate.

Hence, we need to be careful while selecting the similarity threshold for these two types of events. In case of SE events, almost all the clusters can be expected to have high similarity among the popular tags, whereas the ME events may not share the popular tags across all the clusters. This means we might have to set a really low threshold value for SE events but a relatively higher threshold value for ME events. Based on the observations from tables 5.1 and 5.2, we use the size of the ordered clusters as a function to determine the threshold value. A huge drop in size from first cluster (E_1) to second cluster (E_2) signifies a single entity event and thus $threshold = \frac{len(E_2)}{len(E_1)}$ assigns a really small score. At the same time this score will be much larger in case of multiple entities events due to comparable cluster sizes. This function for assigning the score also satisfies the bounding range for similarity score $[0, 1]$. This threshold value is not hard coded but automatically computed based on the generated clusters. We acknowledge that the above approach is a heuristic. In this chapter, initial evidence is shown that it is viable based on the data sets we considered. More experience with use of this heuristic is needed to make more general applicability claims.

5.4 EVALUATION

In this section, we first describe the various real-world datasets that we collected using the Instagram API. With the help of these datasets, we validate the hypotheses presented earlier in this paper. Finally, we show a comparison of the performance of our algorithm against a few baseline methods for localizing events.

5.4.1 Collected Datasets

- **Taylor Swift Music Tour** (Dataset 1) - Taylor Swift, one of the most popular American singers, conducted a music concert tour called **The 1989 World Tour** in various cities across the world. We collected the complete set of Instagram posts related to this tour using the hashtag #1989*worldtour* starting from May 5, 2015, until December 12, 2015. We evaluate a total of 28 events spanning across the last three months of the event tour that happened in various cities in United States, Canada, Asia and Southeast Asia, and Australia. The ground-truth locations for all the events

were obtained from the Wikipedia page [51] associated with the tour. The average number of users per event was 113.5 and the average number of photos per user was 2.8 from the collected Instagram posts.

- **Maroon V Music Tour** (Dataset 2) - The **Maroon V Tour** is a music concert tour by the popular American band Maroon V. We collected the Instagram posts related to this tour using the hashtag *#maroonvtour* starting from February 16, 2015, until October 4, 2015. We evaluate a total of 17 events from the months of September and October spanning different cities in south east Asia and Australia. The ground-truth locations for all the events were obtained from a Wikipedia page [52] associated with the tour. The average number of users per event was 78.94 and the average number of photos per user was 2.2 from the collected Instagram posts.

- **Marathons** (Dataset 3) - According to the 2014 annual marathon report [53], more than $1,100$ races were completed across the United States, making it one of the most popular urban sporting events. For the purpose of evaluating our work, we considered the top 30 cities in the United States, ranked by population [54] that hosted a popular marathon [55] during the fall of 2015. Based on this list we looked at the 5 popular marathons in the largest cities by population as listed in table 5.3. The choice of restricting to largest cities and most popular marathons was made to reduce the manual effort in verifying the ground truth of detected events that were significantly high in number as compared to music concert events. The average number of users per event was 122.4 and the average number of photos per user was 2.85 from the collected Instagram posts.

Table 5.3: List of major US Marathons, Fall 2015

Event	City	Marathon	Date
1	Chicago	Bank of America Marathon	Oct 11
2	Baltimore	The Under Armour Marathon	Oct 17
3	Washington D.C.	Marine Corps Marathon	Oct 25
4	NY City	TCS Marathon	Nov 1
5	Las Vegas	Rock n Roll Marathon	Nov 15

Instagram posts related to marathon events were collected using the hashtag *#marathon*. It is important to note that this search query tag is not targeted towards a particular entity such as a name (Maroon V) as in the case of previous datasets. Thus, this data

set is much more "noisy" compared to others, making it a very interesting case to consider.

- **Tornadoes** (Dataset 4) - The number of fatalities caused by **tornadoes** in the United States during the year 2015 [56] is estimated at 480, an exponential increase compared to 54 recorded during 2014. The strength of a tornado is computed using the *EF* scale ranging between $(0, 5)$ based on the damage caused. In 2015 (Jan - Oct), there were no *EF5* tornadoes, while the count of *EF4* was 2 and *EF3* was 8. The *EF3* tornadoes have mostly occurred in rural areas with populations less than $5,000$, except for one urban location. Instagram posts related to tornadoes were collected using the hashtag *#tornado*. Table 5.4 lists the tornadoes that we selected by looking at the amount of fatality caused in urban areas with a population of at least $5,000$. The average number of users per event was 6 and the average number of photos per user was 2.16 from the collected Instagram posts.

Table 5.4: List of Fatal Tornadoes, 2015 (Jan-Oct)

Event	City	EF	Date
1	Rochelle, IL, USA	4	April 9, 2015
2	Oklahoma City, OK, USA	3	May 6, 2015
3	Venice, Italy	4	July 8, 2015

5.4.2 Validation of Hypotheses

Before we present the performance results of our localization algorithm, we demonstrate the validity of the assumptions that were made earlier while deriving the algorithm. Specifically, we focus on *Hypotheses* that can be validated with the help of experiments using the datasets collected. We evaluated a total of 53 events from these 4 datasets and for each of the following validations we randomly sample few events in order to ensure that our assumptions are not holding true only on the basis of best quality events.

- **Validation of Assumptions 5 and 6** - In figure 5.2, we show the distribution of unique users present in *True* versus the *False* clusters for fifteen events that were randomly selected from the output of our localization algorithm using the collected datasets. The x-axis represents the event ID while the y-axis represents the fraction of users who posted images for that particular event. This figure validates two assumptions at the same time. Firstly, we can see that there are some groups of users who are located at places other than the actual event location (Assumption 5), and secondly,

the fraction of users from the *True* location is always greater than the *False* location for the same event (Assumption 6).

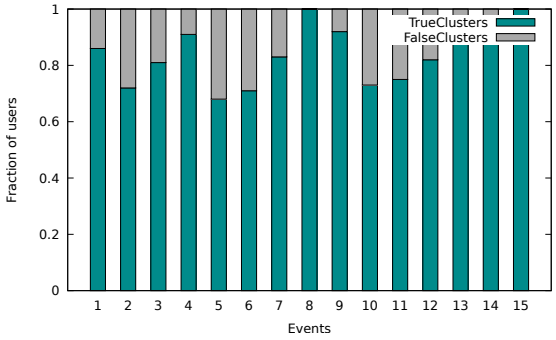


Figure 5.2: True and False event clusters

- **Validation of Assumption 7** - For this validation we select two random event samples from each of the two categories (*Expected* and *Unexpected*) to plot the distribution of the frequency of images shared form the *True* location. For these plots, the x-axis represents the timestamp ID and y-axis represents the frequency of images that were shared for a particular time interval. Figures 5.3, 5.4, consists of two subplots each corresponding to *Expected* and *Unexpected* events. For each subplot the start and the end time has also been indicated using the ground truth. Thus, it can be validated that a peak in the frequency of images shared happens within the range of event occurrence.

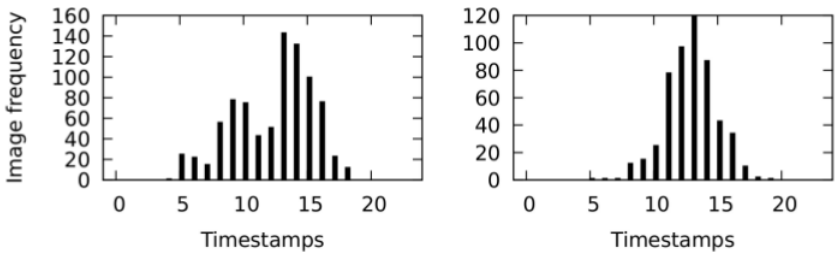


Figure 5.3: Distribution of images for expected events

- **Validation of Assumption 8** - In figure 5.5, we validate Assumption 8 using the same random fifteen events that were selected for validation of Assumption 5. For each event, we first identify the top 10 commonly used tags according to frequency (we remove the tag word used for search query) from both *True* and *False* clusters. Next, we determine

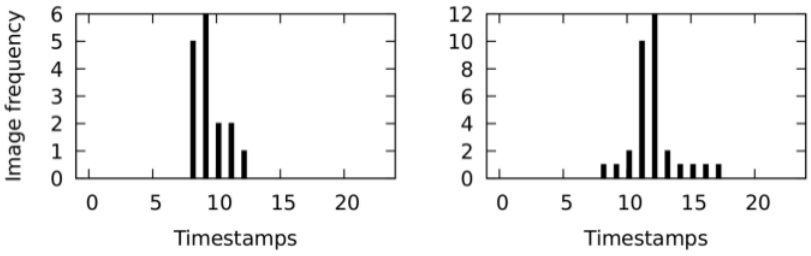


Figure 5.4: Distribution of images for unexpected events

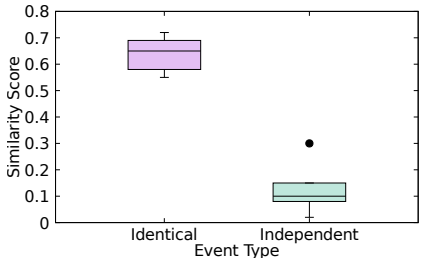


Figure 5.5: Prevalence of Commonly Used Tags

the similarity between the *True* cluster vector with each of the corresponding *False* cluster vector as well as other *True* cluster vectors that are independent. Figure 5.5 shows the boxplot representation for similarity between identical events in plot A and independent events in plot B. It is evident that the median for events and their echos is around 0.65 and the minimum score is well above 0.5. At the same time independent *True* events are well separated with a maximum outlier score of 0.3. Thus, there exists some amount of prevalence of common tags between the *True* and *False* clusters of the same event.

5.4.3 Performance of our Localization Algorithm

With the establishment of the validity of the assumptions that we made in order to derive our localization algorithm, we now compare the performance of the results against a few baseline methods using different metrics. The baselines and the metrics are discussed in detail below followed by the comparison tables.

- **Baseline Method 1 - Tag Similarity Localization** - The first baseline method is based on the intuition that all the observations for an event are closely linked to each other in terms of common tags used for description (this is according to our Assumption 8). We follow the same processing method for the incoming feed of data

using the crawler. For any current interval, we consider all the unique K locations $(l_1, l_2, \ldots, l_k)$ along with the associated sets $\mathcal{T}_k$, $\mathcal{U}_k$ and $\mathcal{G}_k$. We then form a cluster by grouping all the l's for which the similarity score among the top 10 common tag words is at least $X\%$. We vary the value of X as 20, 40, 60, and 80 respectively. For each case, we use the same false alarm cluster elimination technique as described in our own algorithm. The higher the threshold for grouping locations, the better the results will be.

- **Baseline Method 2 - Geo Event Detection** - For the second baseline method, we use the work described by the authors of [57] for geographical social event detection in social media. This work is very closely related to our motivation in terms of using geo-tagged data to detect events. We implement their algorithm as mentioned to detect the events on our collected datasets. Specifically, we do per day analysis for the four time slots on each geographic region present for that day. A region comprises of geo-coordinate with maximum number of users and all points within 30 miles of radius from it. There is a threshold requirement for abnormal geographic regions. We vary this θ value as 0.2, 0.4, 0.6 and 0.8 to see the effect on localization. The minimum number of observations required in a cluster is set as 3. Values of θ greater than 0.8 provided the same results and hence we do not show them explicitly.

- **Baseline Method 3 - Points of Interest** - For the third baseline, we use the work described by the authors of [58] in order to find points of interests using pictures shared by users on the Instagram network. This work can be very well applied to our interest of finding the locations of events. However, the authors conducted the experiments on very popular locations. Thus, we again set the minimum number of observation required in a cluster as 3 and use the approach as described in the paper.

- **Metrics for comparison** - We use three metrics in order to compare the performance of our localization algorithm against the selected baseline methods:

  - *Recall* : Determines the count of events that were detected and localized from the available set of events.
  - *False Positives (FP)*: Determines the count of events that were falsely classified as positive.
  - *Average Localization Error (ALE)* : Determines the average error in the estimated location from the actual ground truth for all the localized events.

Table 5.5: Recall

Dataset	**Our Localization Algorithm**	**Tag Similarity Localization**				**Geo Event Detection [57]**				**Points of Interest [58]**
		$X =$ 20%	$X =$ 40%	$X =$ 60%	$X =$ 80%	$\theta = 0.2$	$\theta = 0.4$	$\theta = 0.6$	$\theta = 0.8$	
Taylor Swift	28/28	24/28	25/28	26/28	26/28	28/28	28/28	28/28	28/28	27/28
Maroon V	17/17	13/17	15/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17
Marathon	5/5	3/5	4/5	4/5	4/5	5/5	5/5	5/5	5/5	5/5
Tornado	3/3	2/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3

Table 5.5 is the recall value comparison between our localization algorithm and the baseline methods under different settings. Our method performed consistently well in correctly identifying all the events. Baseline 2 method also gave a perfect recall.

Table 5.6: False Positives

Dataset	**Our Localization Algorithm**	**Tag Similarity Localization**				**Geo Event Detection [57]**				**Points of Interest [58]**
		$X =$ 20%	$X =$ 40%	$X =$ 60%	$X =$ 80%	$\theta = 0.2$	$\theta = 0.4$	$\theta = 0.6$	$\theta = 0.8$	
Taylor Swift	2	18	10	5	4	35	16	9	9	26
Maroon V	0	5	4	4	2	19	8	8	8	14
Marathon	0	16	10	7	6	17	11	11	11	15
Tornado	1	3	3	3	2	6	6	6	6	19

Table 5.6 is the false positives value comparison between our localization algorithm and the baseline methods under different settings. It can be clearly seen that our method generated the least number of false alarm clusters for any dataset.

Table 5.7 is the ALE comparison between our localization algorithm and the baseline methods under different settings. It can be clearly seen that our method has the best average error rate for the estimated location from the actual ground truth. In case of first two datasets (which are immobile events), the average error is almost close to zero, but for the other two datasets (mobile events), the average error is close to 6 miles in worst case.

Table 5.7: Average Localization Error (miles)

Dataset	**Our Localization Algorithm**	**Tag Similarity Localization**				**Geo Event Detection [57]**				**Points of Interest [58]**
		$X = 20\%$	$X = 40\%$	$X = 60\%$	$X = 80\%$	$\theta = 0.2$	$\theta = 0.4$	$\theta = 0.6$	$\theta = 0.8$	
Taylor Swift	0.03	102.87	25.06	2.86	1.02	0.78	0.78	0.78	0.78	0.17
Maroon V	0.12	75.34	32.78	10.23	2.67	1.23	1.23	1.23	1.23	1.32
Marathon	3.45	141.43	34.33	16.18	4.12	4.82	4.82	4.82	4.82	5.86
Tornado	6.02	40.23	25.23	11.34	11.34	9.06	9.06	9.06	9.06	8.47

5.5 RELATED WORK

The exploitation of social networks that expose *location information* had been studied in depth long before Instagram became popular. These networks provide location data in different formats (text, images, etc.) enabling localization.

Foursquare: Foursquare is a widely used social network for checking into visited places and sharing reviews online with other users. It features around 100 million check-in venues worldwide. In [59], a study was reported using Foursquare to reveal user mobility patterns in urban spaces. Other work [60] focused on analyzing the mobility patterns of users to identify social ties based on co-location history, and determine the relation between location visits and network strength of a user. In [61], clustering techniques were presented for finding the dynamics of a city based on the check-ins posted by users on Foursquare. Noulas et al. [62] proposed a method that uses Foursquare check-ins to identify regions that are similar within a geographic area. Foursquare coverage, however, is much sparser than Instagram. With more than 80 million pictures uploaded per day, Instagram sees more pictures in 10 days than Foursquare since its creation in 2009.

Instagram and Flickr: Due to an explosive increase in the user base over the past three years, Instagram has emerged as a popular platform among researchers to analyze social networks from a crowdsensing point of view. In [63], Instagram was studied 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 [64], 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.

Another work [**?**] explored the feasibility of using Instagram pictures along with metadata to find a correlation between obesity patterns and fast food restaurants located in few selected counties within the United States. In the work by Mejova et al. [65], a further analysis was performed on the food habits of users on a global scale to answer questions related to health research. Prior work [58] 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 [66] 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. In [67], an analysis of Flickr was presented to show the variation in the popularity of photos around a geographical location. Related event detection work considered geo-tagged data from Flickr [68]. They focused on nine events using an online event directory to define a bounding box around using GPS data from Flickr images. The events were then detected using time-series analysis within the box based on a threshold. The work presented in [69] offered another example of event detection techniques using geo-tagged images from social networks. A hybrid similarity graph was constructed based on tags and images to form clusters that were then classified using a trained model.

Detecting and localizing events using Instagram (or Flickr) offers several challenges. For example, not all events are equally popular. A baseball game or a large concert might have more observers than a local flashmob event, and observers of the game or concert will generally be more tightly packed in space than observers of, say, an earthquake. Some events have *echos*; namely, observers that are concentrated in space (e.g., at a large night club or bar), who are observing the event remotely (e.g., on a TV screen). This may lead to sets of clustered images tagged with the event name (as in "Bob watching #TaylorSwiftConcert in bar") that are not at the actual event location. Our work adds two contributions compared to the aforementioned prior efforts. First, we adapt to the size and nature of the event in an entirely usupervised fashion (without prior training), despite the large variability in size (e.g., an echo of a Taylor Swift concert may be larger than an original flash mob event). Hence, we provide a simple and robust approach that works online for streaming data feeds without using a supervised classification model. Second, we offer mechanisms that discard "echos" (of larger events) that may otherwise result in false positives. To our best knowledge, this approach has never been explored before for event localization in urban spaces.

5.6 SUMMARY

This chapter presents an algorithm for localizing urban events using geo-tagged media from the Instagram social network. We chose Instagram because of its availability and advantages in reflecting spatio-temporal event distribution. Our solution employed a clustering technique that is entirely unsupervised. It chooses an event radius adaptively to maximize the quality of clusters generated. It further considers similarity between clusters to minimize false positives. For evaluation, we considered three baseline methods and compared the results with our localization algorithm. The results show that we outperform the baselines in all the three metrics considered for comparison. We achieve this result without the need to tune any manual parameters.

CHAPTER 6: EVENT PATH ESTIMATION WITH INSTAGRAM

This chapter describes a method to estimate the path of real-world events in space using the data shared by users on social media platforms. Specifically, we focus on Instagram to perform case studies on global as well as local events. In the previous chapter we demonstrated that it is possible to detect events and we now extend the research to estimating the path of mobile events. Past works using traditional estimation models such as particle filtering work with an underlying assumption that arrival of observations is chronological in order. However, in this chapter, we consider the case when observations are shared in a random order and demonstrate a new capability to identify the path of an event and also reconstruct its timeline as it moves across space. Our approach is completely unsupervised and requires no prior training to identify the path. We evaluate our algorithm using case studies on four different real-world events falling under different categories. The results indicate we perform better than the baseline methods in estimating the path.

6.1 OVERVIEW

Target tracking is an age old problem in sensor systems and many efforts have been made over the years to propose different solutions. For example, in [70] the authors have presented an approach to track a moving object with the help of a new particle filtering style algorithm. In physical sensor domain, most of the tracking problems [71–74] try to use some variant of the particle filtering method [75] where a probabilistic model of state estimation is derived with the help of measured data and motion of the object. This popular technique has also been used for social network platform [5] to perform tracking of earthquakes and typhoons with the help of tweets shared by users in real-time. One basic assumption in all the previous works is that the measured data arrives in a timely order and the noise is introduced by the sensors while transmitting the data. This correct order of arrival of observations allows the model to learn the estimation parameters under a mobility constraint. The problem we are trying to solve takes a different approach using the fact that observed data can arrive in random order during the event occurrence. Using the approaches described in past literature will cause the estimation to always rely on the mobility model and never take into consideration the observations made by sensors if arrival order is too noisy. Hence, we need a new technique that uses a noise model along with a mobility model for sensing events in such an environment.

In this chapter we solve this problem by introducing a novel algorithm to identify the

path along which an event moves in space over a period of time. We consider Instagram as the social media to evaluate the performance of our algorithm in estimating the path of the events. One important thing to note is that it is possible to share a picture at a timestamp other than actual event occurrence. For example, an user taking pictures of a parade decides to share them after going back home. Smart phones enabled with geotagging service encode the location information in the picture which can be extracted by the Instagram app. Thus, we see that even though the observation is geotagged with the true location but it arrives in the network with some time delay. The problem arises when some observations are shared in real-time and others shared with random delays which is the focus of this chapter for estimating the path. Our approach can be used for any social media platform that provides information containing geolocation information in the metadata.

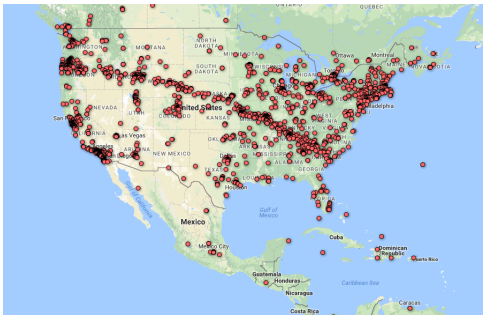


Figure 6.1: Instagram posts shared during Total Solar Eclipse

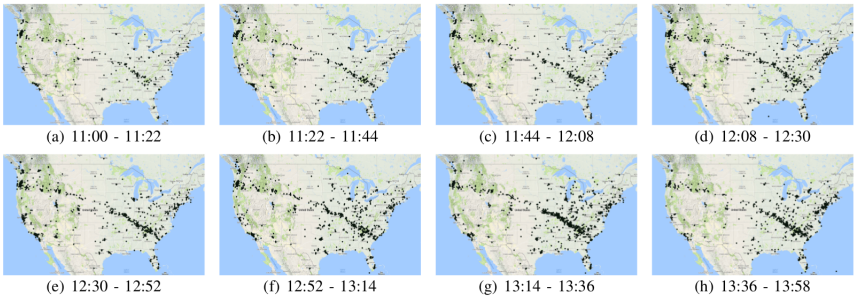


Figure 6.2: Images posted by users on Instagram during different time intervals of the Total Solar Eclipse

The key contribution of this work is an unsupervised approach to estimate the path of an event for a random order of observation timestamps along the actual path and consists of two components - the first is elimination of noise and the second is estimation of path from candidate points. It is important to note that the term "*noise*" in this paper refers to off-track points, however, they might be describing information of the actual event. For example, figure 6.1 shows a map plot of the Instagram posts shared during the Total Solar

Eclipse which occurred on August 21, 2017. These posts were retrieved from the Instagram API using the query *#totalsolareclipse*. It can be clearly seen there is a sharing pattern aligned with the with the actual path of the event (west coast to east coast) but at the same time several images are posted from off-track locations. In figure 6.2 we show the shared posts at every 22 minutes interval for the entire duration of the eclipse. The subplots reveal that some users started sharing posts from locations (towards east) on the actual path before the event as they were observing a partial eclipse and some users where still sharing posts from locations (towards west) after the total eclipse ended.

The off-track points in our example occur mainly due to two reasons. One is the popularity of the event that attracts attention from all cities that have a dense population. Cities such as Los Angeles, San Francisco, and New York are the hot-spots in this example. The second reason is that even the users observing a partial eclipse from other locations share their images. Thus, we need to eliminate such noisy posts before estimating the path of the event. The solar eclipse example used here is to only visulaize the existing problems that motivate us to derive an approach which can be applied in general for any event. To demonstrate this we provide two case studies using events pertaining to both a global and a local scale.

The rest of this chapter is organized as follows. Section 6.2 describes the architecture of our model and the problem formulation followed by derivation of the algorithm. Section 6.3 introduces the real-world data collected from Instagram for our case studies. The evaluation is discussed in Section 6.4. Related work is described in Section 6.5. Finally, conclusions are presented in Section 6.6.

6.2 SYSTEM DESIGN

6.2.1 Pipeline Components

Our system comprises of four different modules each of which perform different tasks as described below:

- **Retrieve** - The first module in this architecture is the crawler that sends request to the Instagram API to retrieve posts that contain the user provided keywords. This module also allows to specify the range of timestamps between which the posts are required. Finally, the posts are processed to retain only the geo-tagged images.
- **Eliminate** - This module is the first core part of our algorithm that takes input from the retrieve module and eliminates all the off-track points. The remaining candidate points are then passed to the next module in the pipeline for estimation.

- **Estimate** - This module is the second core part of our algorithm and it analyzes the candidate points to estimate the approximate path taken by an event during a period of time.

- **Visualize** - This final module of the systems helps us visualize the tracked path of an event on a map interface.

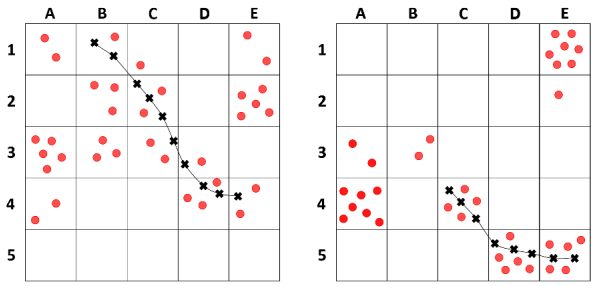
6.2.2 Problem Formulation

The output obtained form the *Retrieve* module consists of geotagged posts. Each post in this list ξ can be considered as an observation from a location L_i. Our goal is to first analyze all the unique locations in ξ such that we can produce two disjoint sets - (i) Φ containing all the off-track locations, and (ii) Ψ containing all candidate locations for estimating the path. This is done using the *Eliminate* module. We then process the set Ψ in the *Estimate* module to produce the set Ω (where $\Omega \subseteq \Psi$) which contains the ordered set of points corresponding to the path of the event. The first location in this ordered set is the start point of the event and each subsequent location is a next point on the continuous path. The notations used in deriving our algorithm are indicated in table 6.1.

Table 6.1: Definition of Notations

ξ	List of geo-tagged Instagram posts
Φ	Set of off-track locations
Ψ	Set of candidate locations for estimation
Ω	Set of ordered locations on the event path
$minsup$	Minimum number of unique users required at a location
L_i	A (lat, long) coordinate
N	Number of unique locations in Ψ
C_i	Frequency of L_i in ξ
s_{ij}	Distance of L_i from L_j in miles
θ_{ij}	Compass bearing direction of L_j from L_i in degrees

Our approach has two core modules - *Eliminate* and *Estimate*. The derivation of the approach is completely unsupervised and uses the observed data for estimation. Figure 6.3 shows two independent events that share the same space. The red dots are the user observations and black cross points are the event locations at a given time t. Since the observations do not come in a chronological order we strip the time factor and consider all the data points at once. We also assume that the path taken by these events have no overlap and the density of population is constant for each grid irrespective of the event.



(a) Event X (b) Event Y

Figure 6.3: Two independent events

6.2.3 Eliminate

As seen in figure 6.3 we have a significant number of user observations made from grids that are far off from the actual path. The expected value of an event can be calculated using the sum of the expected values of the individual grids. Thus we have the following:

$$E(X(t)) = E_{A1} + E_{A2} + \ldots + E_{E5} \tag{6.1a}$$

$$E(Y(t)) = E_{A1} + E_{A2} + \ldots + E_{E5} \tag{6.1b}$$

Every grid can have two possible states for any event, $s = 1$ indicating that it is a true grid where the event actually happened and $s = 0$ indicating that it is not a true grid. Using this we can derive the expected value of any grid as following:

$$p(s = 0) = g(Density) \tag{6.2a}$$

$$p(s = 1) = f(|TrueLoc, Observation|) \tag{6.2b}$$

where $g()$ is a function that depends on the density of a grid and $f()$ is a function that depends on the distance of the observation from the true location. For simplicity we can consider the observation for each grid as the weighted average of data points within that grid. Thus, with the help of an exponential distribution we can define $p(s = 1)$ as following:

$$p(s = 1) = \frac{1}{\alpha} e^{-\frac{dist(T,obs)}{\alpha}} \tag{6.3}$$

Next we perform grid wise subtraction of the two events. This means for every grid if there is an observation for both events then we simply drop that observation. Using the above equations we can have 3 cases for $E(X(t)) - E(Y(t))$ as following:

- Observation is present in a grid for X but not Y. For this case the $g()$ values cancel out each other and we are left with only the true value function. For example:

$$E_{B2}(X(t)) - E_{B2}(Y(t)) = \frac{1}{\alpha} e^{-\frac{dist(T_x, obs_x)}{\alpha}} \tag{6.4}$$

- Observation is present in a grid for Y but not X. Like previous case $g()$ function again cancel out but since there is no user observation for X, the $f()$ function value is considered as 0. This step can be considered as a regularization of the grid to ensure only non-negative values are retained. For example:

$$E_{E5}(X(t)) - E_{E5}(Y(t)) = 0 \tag{6.5}$$

- Observation is present in a grid for both the events. In this case the $g()$ function values cancel out again but the $f()$ function values is dependent on the observation distance from either events. If it is close to X then the value of Y gets subtracted and if it is close to Y then it becomes 0 due to regularization. If the observation is far from both the events then the $f()$ values are close to each other and even they cancel out. For example:

$$E_{A4}(X(t)) - E_{A4}(Y(t)) = 0 \tag{6.6}$$

Thus the subtraction operation of the two events retains the observations around the true location as much as possible with a smaller amount of noise that is independent of the density. In addition to this subtraction of locations,we also perform a couple of pre-processing steps such remove all locations that have a frequency (C_i) less than a threshold. This threshold value is 1 by default but can be changed to a different value as per the popularity of the event. It needs to be also noted that for global scale events like eclipse or hurricane, the path does not have sudden sharp turns but rather follows a smoother curve. We can further eliminate the remaining off-track points by fitting a polynomial of degree k using the points and retaining only those that are within certain distance from the curve. This property cannot be applied to local scale events as the path might have sharp turns which requires us to fit an infinite degree polynomial. Hence, for global scale events, the operation consists of an additional elimination techqniue after the subtraction. The output of the entire operation generates the sets Φ and Ψ.

6.2.4 Estimate

In this module, we take the set Ψ produced by the *Eliminate* module to estimate the path of an event. In deriving this algorithm we focus on two important factors - *direction* and *continuity*. Every event, irrespective of the scale, needs to satisfy these factors. For example, the eclipse event has a direction from west to east and the path is continuous along this direction and the parade event has a direction from north to south with a continuous path. Our second hypothesis is that an event cannot randomly jump off to a location in a direction not within the range of its true motion or to a relatively far off location violating continuity. Using this property, if the current location is $L_i \in \Psi$, then the next possible location $L_j \in \Psi$, is selected according to the following criteria:

$$L_j = \Big\{ min(s_{ij}), \quad \theta_{ij} \in [\Theta_1, \Theta_2]$$

This condition indicates that the next possible location in the continuous path is the one that is closest to the current location and in a direction satisfying the motion. For example, in the case of eclipse, we are aware of the direction being west to east, a valid range of motion can be $[0, 180]$ in degrees. The north direction of the compass is 0 and the south direction is 180. We can reduce the range of motion for a better estimate of the path. Hence, at any given point of time, all the locations out of the range of motion and the current location can be marked as *observed*. When the count of the *observed* locations is equal to N, the current location will be the last point of the tracked path and our algorithm ends. The final output is Ω that contains the ordered set of locations lying on the tracked path of the event. To initiate this module we need to provide with the start location as an input.

6.3 DATA DESCRIPTION

In this section we describe in detail the real-world data collected using the Instagram API. We selected a total of four events. The following subsections provide a detailed information on the datasets:

6.3.1 Dataset 1: Total Solar Eclipse

On Monday, August 21, 2017, an eclipse was observed all over the North America. The path of totality stretched from Salem, Oregon to Charleston, South Carolina. We used the keyword *#totalsolareclipse* to retrieve all the Instagram posts that contained this specific hashtag along with geo-tag information for the day of the event. A total of *11,084* images

were retrieved and the map plot corresponding to these geo-tagged images is shown in figure 6.1.

6.3.2 Dataset 2: Hurricane Irma

Hurricane Irma is one of the strongest Atlantic basin hurricane ever recorded and lasted from August 31, 2017 until September 11, 2017. We used the keyword *#hurricaneirma* to retrieve all the Instagram posts that contained this specific hashtag along with geo-tag information for the duration of the hurricane. A total of *5,009* images were retrieved and the map plot corresponding to these geo-tagged images is shown in figure 6.4.

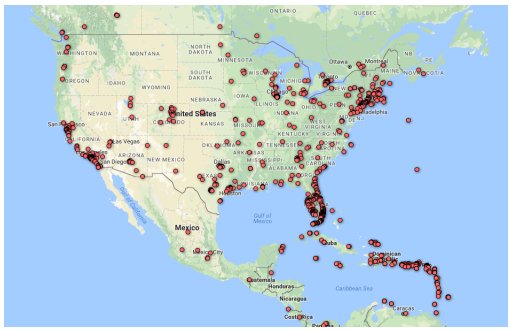


Figure 6.4: Instagram posts shared during Hurricane Irma

6.3.3 Dataset 3: New York City TCS Marathon

The New York City Marathon is an annual marathon that courses through the five boroughs of New York City. It is the largest marathon in the world, with 98,247 applicants for the 2017 race. This event took place on November 5, 2017. For this event we used multiple keywords to retrieve the Instagram geo-tagged posts that contained the following hashtags - *#nycmarathon, #nycmarathon2017, #tcsnycmarathon, #tcsnycmarathon2017*. A total of *21,550* images were retrieved and the map plot corresponding to these geo-tagged images is shown in figure 6.5.

6.3.4 Dataset 4: New York City Thanksgiving Parade

The annual Macy's Thanksgiving Day Parade in New York City, the world's largest parade, is presented by the U.S.-based department store chain Macy's. The 91st annual parade took place on November 23, 2017. We again used a set of keywords to retrieve all Instagram

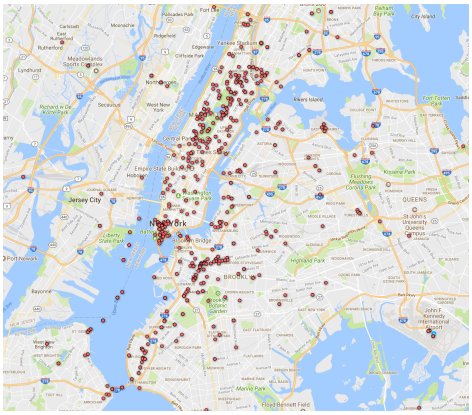


Figure 6.5: Instagram posts shared during NYC Marathon

geo-tagged posts that contained the following hashtags - *#thanksgivingparade, #macys-thanksgivingparade, #nycparade*. A total of *3,826* images were retrieved and the map plot corresponding to these geo-tagged images is shown in figure 6.6.

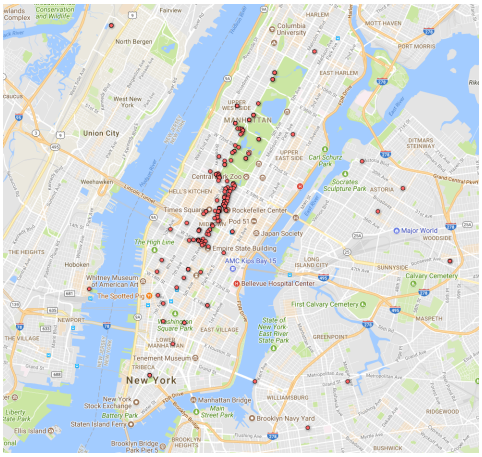


Figure 6.6: Instagram posts shared during NYC Thanksgiving Parade

6.4 EVALUATION

Our evaluation is divided into two case studies. For each we consider the collected datasets and run our algorithm for path estimation. In addition we also use past literature for comparing our performance. The input for each baseline is the raw observations from the dataset. Following are the baseline description:

- **Baseline 1**: We use the Kalman filtering method to track the path of an event as described in [5]. The matrix parameters are considered as mentioned in the paper,

velocity is set according to the event type and acceleration is set as zero.

- **Baseline 2**: We use the Particle filtering method to track the path of an event as described in [5]. The value of N is set as 50 and weight distribution is calculated according to number of users. The velocity is again set as per the event type and acceleration is set as zero.

- **Baseline 3**: We use the path based target tracking as described in [76] in which the observed points are weighted to generate candidate for a short time interval and then line fitting is performed to estimate the path. The weighting scheme is based on the distance of observed points from predicted location as per the laws of motion. After this step the candiate is generated using the weighted centroid of the observations.

6.4.1 Case Study 1: Global Scale Events

In this case study, we show how our algorithm performs in estimating the path of two global scale events. The first event is the *Total Solar Eclipse* and the second is *Hurricane Irma*. For each of these events we perform the two operations - *Eliminate* and *Estimate*, and show the ouput produced at the end of each operation in the following sections. We also compare the estimated path of the events with the true path.

- **Total Solar Eclipse**: For this event we use *Dataset 1* and process all the retrieved images to get the list ($\xi_{eclipse}$) of locations from where observations were recorded. With the help of this list we then use the *Eliminate* operation to produce the candidate set ($\Psi_{eclipse}$) of locations that can be used to estimate the path of the event. During this operation, we first use the list ($\xi_{hurricane}$) of locations from *Dataset 2* to remove all locations $L_i \in \xi_{eclipse}$ that are within $Distance_{threshold} = 30$ miles from any location $L_i \in \xi_{hurricane}$. This choice of the threshold was made keeping in mind that we are estimating the path of a global scale event and a typical radius of densely populated city is around 30 miles. The *minsup* value was set as 2 to get rid off all locations that have no support from more than one user. Finally, we use the remaining candidate locations to fit a polynomial of degree $k = 5$ and removed all the locations that were more than 10 miles away from the curve. The output of all these steps is shown in the first row of figure 6.7. The last image shows the set ($\Psi_{eclipse}$) of locations produced by the *Eliminate* operation. During the *Estimate* operations, we provide the input *current* as a location from Salem, Oregon and restrict the range of motion between $[80, 135]$ since we know that the eclipse moved from west coast to east coast. Based on

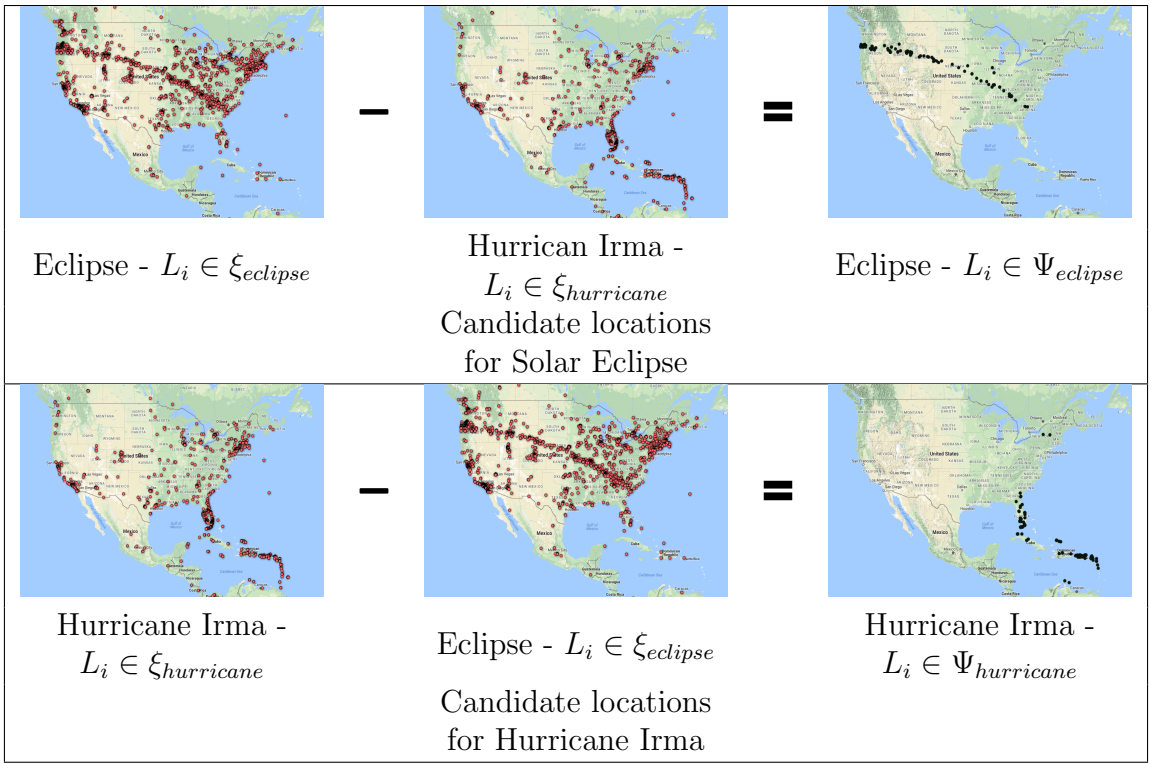
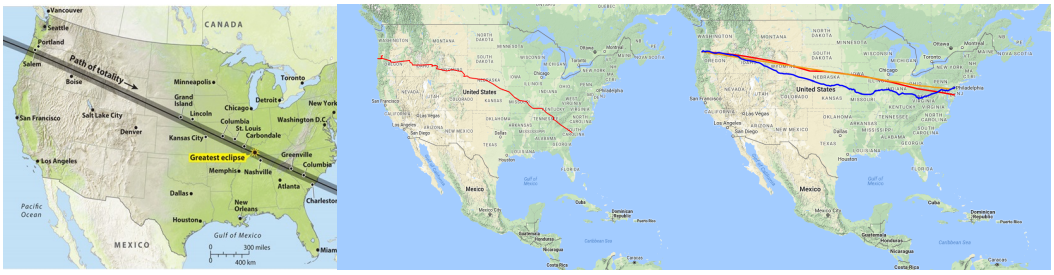


Figure 6.7: Eliminate operation for Dataset 1 and Dataset 2

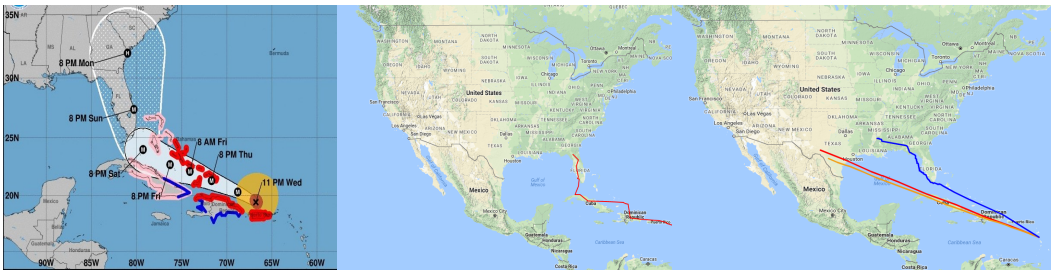
these parameters, the estimated path $\Omega_{eclipse}$ was generated and in figure 6.8 we show the comparison of our estimation against the baseline methods. The last subfigure consists of 3 different paths - the red path is using baseline 1, the orange path is using baseline 2 and the blue path is using baseline 3. The same start point is provided to the baselines and the initial velocity is set as $v_x = 1600 miles/hr$ towards east and $v_y = 500 miles/hour$ towards south. These values are based on typical eclipse speeds and the actual trajectory.



(a) Actual Path (b) Our Algorithm (c) Baselines

Figure 6.8: Tracked path for Total Solar Eclipse

- **Hurricane Irma**: For this event we use *Dataset 2* and process all the retrieved images to get the list ($\xi_{hurricane}$) of locations from where observations were recorded. With the help of this list we then use the *Eliminate* operation to produce the candidate set ($\Psi_{hurricane}$) of locations that can be used to estimate the path of the event. During this operation, we first use the list ($\xi_{eclipse}$) of locations from *Dataset 1* to remove all locations $L_i \in \xi_{hurricane}$ that are within $Distance_{threshold} = 30$ miles from any location $L_i \in \xi_{eclipse}$. We again selected this radius due to analysis of a global scale event as indicated for the previous dataset. The *minsup* value was set as 2 to get rid off all locations that have no support from more than one user. Finally, we use the remaining candidate locations to fit a polynomial of degree $k = 10$ and removed all the locations that were more than 10 miles away from the curve. The output of all these steps is shown in the second row of figure 6.7. The last image shows the set ($\Psi_{hurricane}$) of locations produced by the *Eliminate* operation. During the *Estimate* operations, we provide the input *current* as a location from right most candidate and restrict the range of motion between $[0, 15] \cup [270, 360]$ since we know that the hurricane moved from right to left direction until the Keywest coast of Florida, USA and then moved in an upward direction. Based on these parameters, the estimated path $\Omega_{hurricane}$ was generated and in figure 6.9 we show the comparison of our estimation against the baseline methods. The color code for baselines is same as previous dataset. The same start point is provided to the baselines and the velocity is set as $v_x = 34m/s$ towards west and $v_y = 40m/s$ towards north. These values are based on typical hurricane speeds and the actual trajectory.



(a) Actual Path (b) Our Algorithm (c) Baselines

Figure 6.9: Tracked path for Hurricane Irma

6.4.2 Case Study 2: Local Scale Events

In this case study, we try to show that our approach can be directly applied to a city level event as well for estimating the path. The dynamics of a city remain the same while sharing posts on social media from the densely populated blocks. In other words, we can assume that the posts from these regions occur so frequently that they appear uniformly for all the events taking place in that city. Thus, this enables us to perform the *Eliminate* operation in the same way as we do for global scale events. We again consider two events - the first one is the *NYC Marathon* and the second is the *NYC Thanksgiving Parade*. For each of these events we perform the two operations - *Eliminate* and *Estimate*, and show the ouput produced at the end of each operation in the following sections. We also compare the estimated path of the events with the true path.

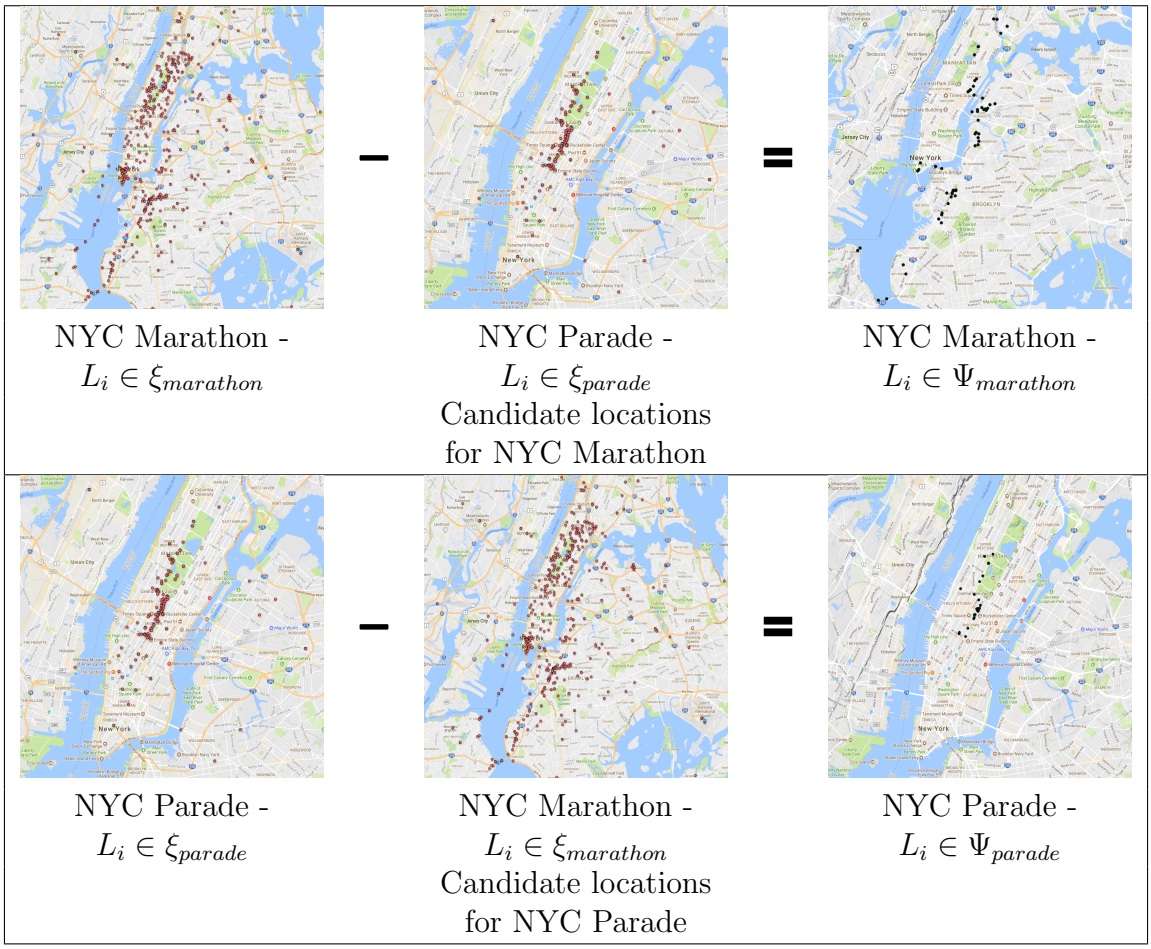
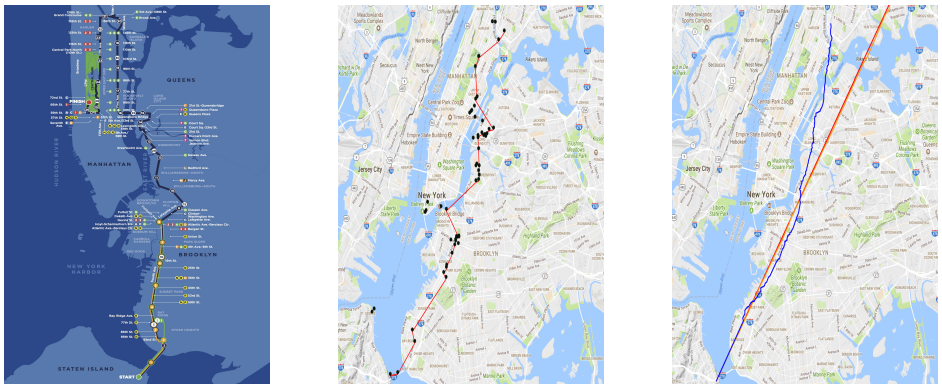


Figure 6.10: Eliminate operation for Dataset 3 and Dataset 4

- **NYC Marathon**: For this event we use *Dataset 3* and process all the retrieved images to get the list ($\xi_{marathon}$) of locations from where observations were recorded. With

the help of this list we then use the *Eliminate* operation to produce the candidate set ($\Psi_{marathon}$) of locations that can be used to estimate the path of the event. During this operation, we first use the list (ξ_{parade}) of locations from *Dataset 4* to remove all locations $L_i \in \xi_{marathon}$ that are within $Distance_{threshold} = 0.5$ miles from any location $L_i \in \xi_{parade}$. This choice of the threshold was made keeping in mind that we are estimating the path of a local scale (within a city) event and the blocks are in 0.5 miles distance on average from each other. The *minsup* value was set as 5 to get rid off all locations that have no support from more than four users. The output of all these steps is shown in the first row of figure 6.10. The last image shows the set ($\Psi_{marathon}$) of locations produced by the *Eliminate* operation. During the *Estimate* operations, we provide the input *current* as a location near the actual start point of the marathon and restrict the range of motion between $[0, 30] \cup [345, 360]$ since we know that the marathon runners moved in south to north direction with few sharp turns on the path. Based on these parameters, the estimated path $\Omega_{marathon}$ was generated and in figure 6.11 we show the comparison of our estimation against the baseline methods. The color code for baselines is same as previous dataset. The same start point is provided to the baselines and the velocity is set as $v_x = 1 mile/hour$ towards east and $v_y = 8 miles/hour$ towards north. These values are based on average marathon speed and the actual trajectory.



(a) Actual Path (b) Our Algorithm (c) Baselines

Figure 6.11: Tracked path for NYC Marathon

- **NYC Parade**: For this event we use *Dataset 4* and process all the retrieved images to get the list (ξ_{parade}) of locations from where observations were recorded. With the help of this list we then use the *Eliminate* operation to produce the candidate set (Ψ_{parade}) of locations that can be used to estimate the path of the event. During this operation, we first use the list ($\xi_{marathon}$) of locations from *Dataset 3* to remove all locations $L_i \in$

ξ_{parade} that are within $Distance_{threshold} = 0.1$ miles from any location $L_i \in \xi_{marathon}$. This choice of the threshold was made keeping in mind that the movement of parade was relatively small event in comparison to marathon. The *minsup* value was set as 2 to get rid off all locations that have no support from more than one user. The output of all these steps is shown in the second row of figure 6.10. The last image shows the set (Ψ_{parade}) of locations produced by the *Eliminate* operation. During the *Estimate* operations, we provide the input *current* as a location near the actual start point of the parade and restrict the range of motion between $[90, 270]$ since we know that the parade moved in north to south direction with few sharp turns on the path. Based on these parameters, the estimated path Ω_{parade} was generated and in figure 6.12 we show the comparison of our estimation against the baseline methods. The color code for baselines is same as previous dataset. The same start point is provided to the baselines and the velocity is set as $v_x = 0.2 miles/hour$ towards west and $v_y = 2.5 miles/hour$ towards south. These values are based on average parade speed and the actual trajectory.

(a) Actual Path (b) Our Algorithm (c) Baselines

Figure 6.12: Tracked path for NYC Parade

In addition to the path estimation for the collected datasets, we also perform a quantitative comparison of our algorithm with the baseline methods. To do this we consider the difference between the area under the actual path and the estimated path. Table 6.2 shows the values obtained for this comparison.

Table 6.2: Difference between area under actual and estimated paths

Dataset	EventMap	B1	B2	B3
Solar Eclipse	23.13	231.2	215.78	167.65
Hurricane Irma	8.16	23.33	21.76	15.36
NYC Marathon	0.13	0.65	0.65	0.37
NYC Parade	0.105	0.87	0.437	0.105

6.5 RELATED WORK

The availability of geotagged data from social media has led to many new research problems over the past few years ranging from identifying interesting geographic regions, recommending places to users, building knowledge about a spatial region, detecting patterns and monitoring unusual events over time. Many of these problems focus on finding intrinsic characteristics associated with a region that govern its dynamics during the occurrence of an event. Two of the earliest works include [77, 78] where the authors use the location data from a particular geographic area and try to build models in order to improve the knowledge about these areas. Using the visit information of individual users they mine the trajectory patterns which are later ranked. In another work [58] the authors use Instagram data to find places of interest within a city. The city dynamics are shown to be correlated with temporal photo sharing. The authors of [79] take advantage of the tourist activity within a city by analyzing their check-ins and then construct a travel diary of important trajectories. They suggest that such diary can be useful for applications in location and transportation management. Similar applications using geotagged data from social media have been proposed by the authors of [80–82].

Apart from improving the knowledge about a geographic area the other problem which gained a lot of interests among researchers was detecting an event. For example, in [68] the authors present a very simple approach to identify events using Flickr data from a selected set of geographic regions with the help of past activity. The work described in [83] uses geotagged posts from Instagram to detect hyper-local events within New York City. The authors use a time-ordered stream of media to detect abnormal signals followed by a classifier implementation to decide if the signal corresponds to an actual signal. [84] is another event detection work in which the authors used geotagged Instagram photos to analyze the time series followed by a classification approach. In another work [85], the authors presented an unsupervised approach to detect events and localize them based on the geotagged posts from Instagram.

The problem we are trying to solve in this chapter is different from these past works. Our focus is to estimate the path of an event in space during its time span. We try to achieve this goal in a completely unsupervised way without any use of historical data from different regions. One of the works which comes closest to our approach is presented by the authors of [5]. An event detection model is first implemented based on temporal features which produces a set of candidates for location estimation. The positive examples are then processed in the spatial model using the principles of Kalman filter and Particle filter techniques. One drawback of using this kind of approach is that the observations need to be in chronological

order along the true path. This might not be true in most cases because the human sensors, unlike the physical sensors, do not have the strict property to send a signal as soon as something is sensed by them. Thus, this leads to a random order of data arrival which does not fit the existing estimation models and we need a new solution to handle the scenario where noise is present in both space and time domains.

6.6 SUMMARY

This chapter presents a novel algorithm using social media data to estimate the path of an event as it moves across the space over a period of time. One important factor which we consider in this chapter is that the observations made by users can be reported in random order as the event moves. This poses a challenge for the traditional target estimation models and requires a new solution. We propose a simple unsupervised algorithm that first eliminates the noisy locations to produce a set of candidate locations which are then used to estimate the path of the event. We also reconstruct the timeline of the event that matches with the ground truth. Our performance evaluation using four real-world dataset shows that we estimate the path more accurately than the baseline methods.

CHAPTER 7: FUSING TWITTER AND INSTAGRAM

This chapter describes the implementation of a service to identify and geo-locate real world events that may be present as social activity signals in two different social networks. Specifically, we focus on content shared by users on Twitter and Instagram in order to design a system capable of fusing data across multiple networks. In the first two chapters we demonstrated the capability to detect physical events using Twitter and Instagram. However, many of these signals need corroboration in order to handle events that lack proper support within a single network. We leverage this insight to design an unsupervised approach that can correlate event signals across multiple social networks. Our algorithm can detect events and identify the location of the event occurrence. We evaluate our algorithm using both simulations and real world datasets collected using Twitter and Instagram. The results indicate that our algorithm significantly improves false positive elimination and attains high precision compared to baseline methods on real world datasets.

7.1 OVERVIEW

The main contribution of this chapter lies in developing a service that uses a fusion algorithm for physical event detection from *multiple social networks* as a way to improve the accuracy of event detection. Specifically, we fuse data feeds from Twitter and Instagram. The two networks have complementary advantages. Twitter data are more prolific, leading to detection of more events, but as shown in our evaluation, it is also more noisy, generating more false-positives. In contrast, Instagram data feeds are sparser, which leads to the benefit of fewer false positives at the expense of detecting fewer events. We show that fusing the two together can offer a solution that features the benefits of both; the results have a much smaller fraction of false positives compared to using Twitter alone, and have more events detected, compared to Instagram. We believe that the solution described in this chapter offers a new point in the trade-off space between precision and recall in event detection techniques from social media data, aiming to combine the benefits of past solutions.

The key underlying analytical contribution lies in a new expectation maximization algorithm that enables event detection using fusion of data feeds from different social networks. By combining data from multiple social media, we are able to detect events that may not have enough corroboration in one network or be indistinguishable from "noise" in another. The algorithm considers the smaller of the data feeds (presently, it is Instagram). For each object in that feed, it attempts to find related objects in the larger feed (Twitter). It then

uses a novel model to statistically estimate the likelihood that the found set of data objects describe a consistent event. If so, an event is said to have been detected. Events detected using this algorithm strike a better balance between false positives and false negatives, compared to either network in isolation, which is the main contribution of the new work.

In Chapter 2 and 5, respectively, we described a system that uses feeds from Twitter [23] (alone) and Instagram [85] (alone) to detect events. This chapter builds on such prior work by offering a novel fusion algorithm that aims to offer a better trade-off between precision and recall of the individual approaches.

The rest of this chapter is organized as follows. Section 7.2 describes the complete architecture of the system we implemented on which the fusion algorithm runs followed the problem formulation and the algorithm of our approach. The evaluation is discussed in Section 7.3. Related work is described in Section 7.4. Finally, conclusions are presented in Section 7.5.

7.2 SYSTEM DESIGN

Our service consists of several runtime modules as illustrated in Figure 7.1. The functionality of each module is described below:

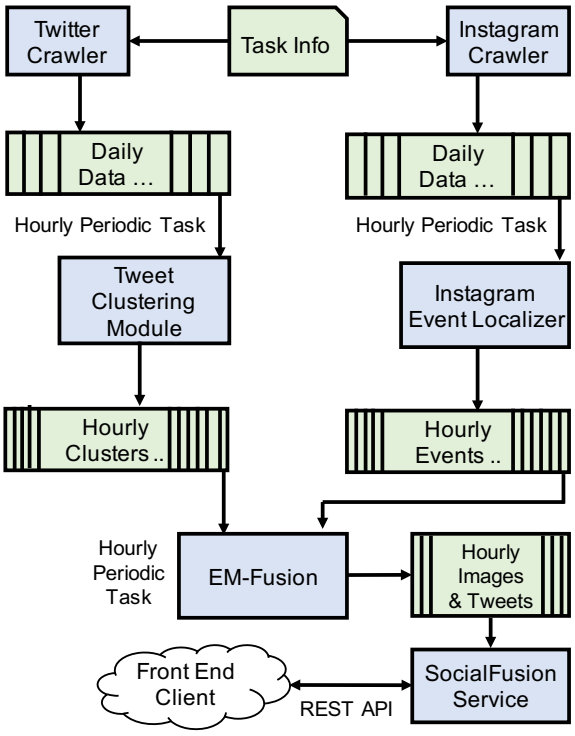


Figure 7.1: Implementing Social Fusion as a Service

1. *Crawler*: This module is provided with a *Task Info* file which contains the keywords entered by the user to initiate the search query. We crawl data every one hour from both Twitter and Instagram with the help of APIs and store it on the disk by grouping data into daily bins.

2. *Tweet Clustering Module*: We use the clustering approach described in [86] to remove all the redundant tweets. This module is initiated as soon as a bin gets updated by the crawler module and outputs a set of hourly clusters in a separate file on the disk.

3. *Instagram Event Localizer*: This module runs an user provided event detection technique on the crawled data from Instagram and generates hourly event clusters for each bin.

4. *EM-Fusion*: This module contains the main fusion algorithm which we describe later in the paper. It reads the data from disk generated by previous two modules and produces an output for all detected events comprising of tweets and images.

5. *Social Fusion Service*: The last module of our service takes care of interaction with the front end client and the output generated using the fusion algorithm.

7.2.1 Problem Statement

In our previous work, we used Instagram and Twitter separately as social networks to localize events in urban spaces. In order to understand whether fusion is feasible, we need to establish that the same events can leave a signature on both networks. Towards that end, we collected data on the topic of protests (i.e., collected tweets and Instagram images tagged "protest"). We then clustered Instagram posts on the topic with the expectation that clusters of images containing the "protest" tag, that originate roughly from the same time and space, likely describe a protest at the indicated time and location. We conducted a small study on a few such clusters of Instagram objects, to check if the corresponding events are also mentioned on Twitter. Table 7.1 contains two examples from the Instagram *protest* dataset. Each of these correspond to cluster locations originating multiple pictures tagged as a protest. The set of all hashtags of these pictures is indicated. We scanned the Twitter *protest* dataset for the same 24 hour interval during which events were identified using our Twitter-based event detection technique [23]. The technique identifies events together with their salient keywords. It is clearly evident from the corresponding tweets and keywords that they refer to the same events and locations. Thus, we can see that a mapping exists between events detected on the two individual networks. The next step is to figure out a

way that can automatically identify this mapping without user interference, even for events that are not independently detected in one or both of the networks.

Table 7.1: Event match examples using Instagram and Twitter

Instagram Location	**Instagram Tags**	**Tweets**	**Event Signature**
(39.045417, -95.721562)	['picket', 'brainwashed', 'westboro', 'protest', 'important', 'wbc', 'truth', 'spreadtheword', 'westborobaptistchurch', 'true', 'dontworrybehappy']	(1) you realize christians protest westboro baptists right is wrong (2) westboro baptist church really protest gunderson production laramie project put years ago (3) fisher westboro protest offers gunderson students opportunity show grizzly pride	(westboro, protest)
(37.7870288, -122.407553)	['protest', 'themission', 'gentrification', 'valenciacorridor', 'googlebus', 'displacement']	(1) laylamrazavi el desalojo ya basta protest googlebus displacement gentrification valenciacorridor (2) video tech workers displaced googlebus protest catch another bus (3) tech buses blocked 45 minutes 2 yrs amp 2 months 1st googlebus protest sfbos sfmayor sb50	(googlebus, protest)

The mapping between related Instagram and Twitter feeds, referred to above, is done on two steps. First, we start with the smaller feed (Instagram). For each object posted in this feed, we identify all *potentially related* posts in the larger (Twitter) feed. Second, with the set of potentially related posts identified, we make a decision on whether they correspond to a real event, or whether the similarity is accidental. These steps are described in the next subsections, respectively.

7.2.2 Finding Potentially Related Posts

To find which Instagram posts are potentially related to which Twitter posts, we need a logical distance metric between an Instagram post and Twitter posts. A convenient metric is the location referred to in the post. However, most tweets do not mention location. Thus, we also need to consider keywords. Instagram posts contain image tags (*hashtags*). We therefore need to identify whether words contained in a tweet are related to these hashtags or not. In this chapter, we choose a "quick and simple" approach that relies on string matching, but does not consider semantics. Better algorithms can be developed by considering semantic distance between different strings.

In developing a string-matching approach, an important question is which string to match? A further look at the event examples in Table 7.1 reveals that not all the Instagram hashtags are equally important in finding matching tweets. For example, in case of the first event

"*westboro*" and "*westborobaptistchurch*" are the only significant tags that help identify the three tweets corroborating the detection. Similarly, in the case of the second event "*google-bus*", "*gentrification*", "*valenciacorridor*", and "*displacement*" are the significant tags that can identify the related tweets. We need to define a metric that helps us find all the tweets that are potentially related to a given set of Instagram hashtags.

To reduce the noise, we first do some preprocessing on tweet text by removing the English stopwords, special characters (non alphanumeric), and weblinks. We also do not consider the query keyword (e.g., *protest*) as it will be present in all the Instagram/Twitter posts by default.[1] It is also important to note that the hashtags are sometimes composed of multiple words merged together. For example, consider the first event again from Table 7.1 in which the significant tag "*westborobaptistchurch*" is actually composed of three different words - *westboro*, *baptist*, and *church*. In order to overcome this issue, we use the processed tweet text and remove all the white spaces to form a single string. Next, we determine the number of hashtags from the Instagram post that are present as a substring within the modified tweet string. This metric known as *tag similarity* is defined as below:

$$tag\_sim = \frac{\#\ of\ tags\ present\ as\ substring\ in\ tweet\ string}{\#\ of\ tags} \tag{7.1}$$

Based on equation 7.1, the similarity score for the tweet - "*westboro baptist church really protest gunderson production laramie project put years ago*" will be $\frac{2}{10}$ (We do not consider the query keyword (*protest*) which was used to collect the datasets in this calculation). Thus the only tags that are present as substrings within the main string are "*westboro*" and "*westborobaptistchurch*".

Table 7.2: Top 5 tweets for Instagram location using tag similarity metric

Instagram Location: (-33.89102, 151.277726) **Location Name**: Bondi Beach, New South Wales, Australia **Tag Similarity Tweets**
(1) great symbolic protest happening right bondi beach sydney bondi electorate turnbull time (2) the people wentworth tell letthemstay protest morning bondi beach (3) photos morning letthemstay protest bondi (4) people gather across australia protest return asylumseekers naru letthemstay (5) saudiarabia wants behead teenager taking part protest humanrights humanity bbc

Using the above defined metric we can now identify tweets that are potentially related to a given Instagram post. If multiple Instagram posts originate from the same location, we can

[1]Remember that in our example, the data was collected by querying Instagram and Twitter for all posts containing the query word, "protest".

combine their tags and compute distance of individual tweets with respect to that combined tag set. This distance will yield similarity to a potential event at the given location. Table 7.2 shows the top five tweets using the metric for a given Instagram location. We emphasize that these are *potentially relevant* tweets. We do not yet know, based on the above distance metric alone, if they are truly relevant or not (i.e., only accidentally similar). A contribution of our work, described below, is to offer a maximum likelihood estimate of actual relevance. This algorithm leads to the discovery of three separate quantities: (i) whether an Instagram location is an actual event location or not, (ii) for a given Instagram event location, what are the significant tags and the corresponding relevant tweets corroborating the observation, and (iii) what is the exact geo-coordinate where the event happened. We propose an unsupervised method in which we assume that we have no prior knowledge of the significance of the Instagram tags as well as the relevance of the tweets using the above similarity metric. The details of this model are described in the following subsection.

7.2.3 Fusion Model

Let us assume that a selected Instagram event detection technique generates cluster $(E_1.E_2.\ldots E_K)$ within a time interval. We then identify the union of the hashtag words $W_1, W_2, \ldots W_M$ that are present in each event cluster E_k. With the help of the geo-tagged coordinates associated with a cluster we also retrieve the exact location name using the Google Maps API [37] service. This location name is of the form $L_1, L_2, \ldots L_L$ where each L_l is a component in the address hierarchy L. Let T be the set of tweets $T_1, T_2, \ldots T_N$ retrieved using the tag similarity metric for the hashtags. Since a tweet can have more than one hashtag, we define A_i as the signature (comprising of one or more hashtags) which retrieves the tweet T_j. We define R_j as the relevance variable ($R_j \in \{0, 1\}$) indicating if a particular tweet T_j is relevant to an event cluster E_k or not. For every hashtag signature A_i we have a group of associated tweets. This enables us to find the average word vector that can be related to the hashtag signature A_i. We define the average word vector as the list of all distinct words from the associated tweets using their average count. We also link L_l to a tweet T_j depending on whether the location name appears in the tweet or not. The definition of all the notations used are mentioned in Table 7.3.

For every tweet T_j we can now define a score based on its distance (using cosine similarity) from the average word vector of corresponding hashtag signature A_i. It can be assumed that all the relevant tweets are more likely to represent the same information. Thus a hashtag signature A_i generating relevant tweets will have an average word vector close to all the relevant tweets resulting in high similarity scores. Whereas a tag signature generating noisy

Table 7.3: Definition of Notations

E_k	Instagram Event Cluster
A_i	Signature composed of hashtags used in a cluster E_k
T_j	Tweet associated with a cluster E_k
L_l	Location name associated with a cluster E_k
R_j	Relevance of a tweet $\in \{0, 1\}$
C_{ij}	Coherence score using word vector of hashtag A_i and corresponding tweet T_j
L_{lj}	Indicator if location L_l appears in $T_j \in \{0, 1\}$
$B(\alpha, \beta)$	Beta distribution with parameters α and β

tweets will produce a word vector that results in low similarity scores. We define this property as *Coherence* which tries to distinguish between the two set of classes. At the same time we also use the location information to increase the confidence of our assumption. For every location name L_l we define p_l as the probability that it appears in the tweet T_j given that it is relevant and q_l as the probability that it appears in the tweet T_j given that it is not relevant. Mathematically, we can define these terms as follows:

$$p_l = P(L_{lj} = 1 | R_j = 1) \tag{7.2a}$$

$$q_l = P(L_{lj} = 1 | R_j = 0) \tag{7.2b}$$

We consider that a location name is more likely to be a part of relevant tweet than the irrelevant tweet and hence put the condition $p_l \geq q_l$. For example, in Table 7.2 the location name *Bondi Beach* appears in all the relevant tweets. Also the *Coherence* property varies in the range [0,1] which allows us to define a *Beta* distribution for the two classes. The motivation behind using the *Beta* distribution is that it is more suitable for a random behavior of proportions. We set the parameters as (α_R, β_R) for $R_j = 1$ and $(\alpha_{\tilde{R}}, \beta_{\tilde{R}})$ for $R_j = 0$. We can now define the conditional probabilities for a tweet T_j using the coherence score and the location names as defined below:

$$P(C_{ij}|R_j = 1) = B(\alpha_R, \beta_R, C_{ij})$$
$$P(C_{ij}|R_j = 0) = B(\alpha_{\tilde{R}}, \beta_{\tilde{R}}, C_{ij})$$
$$P(L|R_j = 1) = \prod_{l=1}^{L} p_l^{L_{lj}} (1 - p_l)^{(1-L_{lj})}$$
$$P(L|R_j = 0) = \prod_{l=1}^{L} q_l^{L_{lj}} (1 - q_l)^{(1-L_{lj})}$$

We use the Expectation-Maximization (EM) algorithm in order to find the relevance (latent variable) of the tweets and also estimate the unknown parameters for the Coherence and location names. Given an observed data X, that is the Instagram tags and location names along with retrieved tweets, one should carefully select the values of the latent variable R and the unknown parameters θ to formulate the likelihood function $f(\theta; X, R) = p(X, R|\theta)$. The EM algorithm finds the maximum likelihood estimate by iteratively performing the following steps:

- E-step: Compute the expected log likelihood function, where the expectation is taken with respect to the computed conditional distribution of the latent variables given the current settings and observed data:

$$Q(\theta|\theta^{(t)}) = E_{R|X,\theta^{(t)}}[log f(\theta; X, R)] \tag{7.4}$$

- M-step: Find the parameters that maximize the Q function in the E-step to be used as the estimate of θ for the next iteration:

$$\theta^{(t+1)} = argmax\ Q(\theta|\theta^{(t)}) \tag{7.5}$$

We denote the probability of a tweet being relevant $P(R_j = 1)$ as d. Thus, the set of unknown parameters for the observed data X is given by $\theta = (p_l, q_l, \alpha_R, \beta_R, \alpha_{\tilde{R}}, \beta_{\tilde{R}}, d)$. The likelihood function $f(\theta; X, R)$ is given by:

$$p(X, R|\theta) = \prod_{j=1}^{N} \left\{ \prod_{l=1}^{L} p_l^{L_{lj}} (1 - p_l)^{(1-L_{lj})} \times B(\alpha_R, \beta_R, C_{ij}) \times d \times R_j + \prod_{l=1}^{L} q_l^{L_{lj}} (1 - q_l)^{(1-L_{lj})} \times B(\alpha_{\tilde{R}}, \beta_{\tilde{R}}, C_{ij}) \times (1 - d) \times (1 - R_j) \right\} \tag{7.6}$$

In eq. (7.6), d represents the overall prior probability that an arbitrary tweet is relevant. We can now formulate an expectation maximization algorithm that jointly estimates the parameter vector θ and the probability that latent variable $R_j = 1$.

7.2.4 Deriving the E-step and M-step

Given the likelihood function as described in eq. (7.6), we substitute it to the definition of Q function of the Expectation Maximization. Thus the E-step becomes:

$$\begin{aligned} Q(\theta|\theta^{(t)}) &= E_{R|X,\theta^{(t)}}[logf(\theta; X, R)] \\ &= \sum_{j=1}^{N} \Bigg\{ P(R_j = 1|X_j, \theta^{(t)}) \times \Bigg[\sum_{l=1}^{L} (L_{lj} logp_l + (1 - L_{lj}) log(1 - p_l)) \\ &\qquad + logB(\alpha_R, \beta_R, C_{ij}) + logd \Bigg] + \\ &\qquad P(R_j = 0|X_j, \theta^{(t)}) \times \Bigg[\sum_{l=1}^{L} (L_{lj} logq_l + (1 - L_{lj}) log(1 - q_l)) \\ &\qquad + logB(\alpha_{\tilde{R}}, \beta_{\tilde{R}}, C_{ij}) + log(1 - d) \Bigg] \Bigg\} \end{aligned} \tag{7.7}$$

where X_j is the location names and the hashtag signature A_i associated with a tweet T_j and $P(R_j = 1|X_j, \theta^{(t)})$ is the conditional probability of the latent variable R_j to be true for the given set of observations, which is given by:

$$\begin{aligned} P(R_j = 1|X_j, \theta^{(t)}) &= R(t, j) \\ &= \frac{P(X_j, \theta^{(t)}|R_j = 1)P(R_j = 1)}{P(X_j, \theta^{(t)}|R_j = 1)P(R_j = 1) + P(X_j, \theta^{(t)}|R_j = 0)P(R_j = 0)} \\ &= \frac{U(t, j) \times d}{U(t, j) \times d + V(t, j) \times (1 - d)} \end{aligned} \tag{7.8}$$

where $U(t, j)$ and $V(t, j)$ are defined as:

$$U(t, j) = \prod_{l=1}^{L} p_l^{L_{lj}} (1 - p_l)^{(1 - L_{lj})} \times B(\alpha_R, \beta_R, C_{ij}) \tag{7.9a}$$

$$V(t, j) = \prod_{l=1}^{L} q_l^{L_{lj}} (1 - q_l)^{(1 - L_{lj})} \times B(\alpha_{\tilde{R}}, \beta_{\tilde{R}}, C_{ij}) \tag{7.9b}$$

Similarly, $P(R_j = 0|X_j, \theta^{(t)})$ can be represented as:

$$P(R_j = 0|X_j, \theta^{(t)}) = 1 - R(t,j)$$
$$= \frac{V(t,j) \times (1-d)}{U(t,j) \times d + V(t,j) \times (1-d)} \quad (7.10)$$

Substituting from eq. (7.8) and eq. (7.10) into eq. (7.7) we get:

$$Q(\theta|\theta^{(t)}) = E_{R|X,\theta^{(t)}}[logf(\theta; X, R)]$$
$$= \sum_{j=1}^{N} \Bigg\{ R(t,j) \times \Big[\sum_{l=1}^{L} (L_{lj} log p_l + (1 - L_{lj}) log(1 - p_l)) + logB(\alpha_R, \beta_R, C_{ij}) + logd \Big] +$$
$$(1 - R(t,j)) \times \Big[\sum_{l=1}^{L} (L_{lj} log q_l + (1 - L_{lj}) log(1 - q_l)) + logB(\alpha_{\tilde{R}}, \beta_{\tilde{R}}, C_{ij}) + log(1-d) \Big] \Bigg\} \quad (7.11)$$

For the M-step we select θ^* that maximizes $Q(\theta|\theta^{(t)})$. Thus, we set the derivatives $\frac{\partial Q}{\partial p_l} = 0$, $\frac{\partial Q}{\partial q_l} = 0$, $\frac{\partial Q}{\partial \alpha_R} = 0$, $\frac{\partial Q}{\partial \beta_R} = 0$, $\frac{\partial Q}{\partial \alpha_{\tilde{R}}} = 0$, $\frac{\partial Q}{\partial \beta_{\tilde{R}}} = 0$, and $\frac{\partial Q}{\partial d} = 0$. With respect to d we have the following equation:

$$\sum_{j=1}^{N} \frac{R(t,j)}{d} + \sum_{j=1}^{N} \frac{(1 - R(t,j))}{1-d} \quad (7.12)$$

Solving the eq. (7.12) we get the following value of d:

$$d^{(t+1)} = d^* = \frac{\sum_{j=1}^{N} R(t,j)}{N} \quad (7.13)$$

Since we have an inequality defined with respect to p_l and q_l, we use the Karush-Kuhn-Tucker (KKT) conditions while performing the maximization step. Thus our inequality constraint ($g : q_l - p_l \leq 0$) allows us to define two regions depending on whether the constraint is inactive or active. In the case where g is inactive the Lagrangian multiplier (λ) will have a value 0 and we get the following equations:

$$\sum_{j=1}^{N}\left[R(t,j)(\frac{L_{lj}}{p_l^*}-\frac{1-L_{lj}}{1-p_l^*})\right]=0 \tag{7.14a}$$

$$\sum_{j=1}^{N}\left[(1-R(t,j))(\frac{L_{lj}}{q_l^*}-\frac{1-L_{lj}}{1-q_l^*})\right]=0 \tag{7.14b}$$

Solving the above set of equations we get the following values of p_l^* and q_l^*:

$$p_l^{(t+1)}=p_l^*=\frac{\sum_{L_{lj}=1}R(t,j)}{\sum R(t,j)} \tag{7.15a}$$

$$q_l^{(t+1)}=q_l^*=\frac{K_l-\sum_{L_{lj}=1}R(t,j)}{N-\sum R(t,j)} \tag{7.15b}$$

where K_l is the total number of tweets in which location name L_l is present. However, if the constraint is not satisfied and we are in the active region, then we need to solve for the Lagrangian multiplier subject to the condition $\lambda \geq 0$. By solving for the optimal values, we get the following equation:

$$p_l^*=q_l^*=\frac{\sum_{j=1}^{N}R(t,j)}{N} \tag{7.16}$$

From the above equation we see that p_l and q_l have the same value, which indicates that the location name does not have a pivotal role in determining the relevancy of the tweet. For the *Beta* distribution parameters, we get the following set of equations:

$$\psi(\alpha_R^*)-\psi(\alpha_R^*+\beta_R^*)=\frac{1}{N}\sum_{j=1}^{N}R(t,j)logC_{ij} \tag{7.17a}$$

$$\psi(\beta_R^*)-\psi(\alpha_R^*+\beta_R^*)=\frac{1}{N}\sum_{j=1}^{N}R(t,j)log(1-C_{ij}) \tag{7.17b}$$

$$\psi(\alpha_{\tilde{R}}^*)-\psi(\alpha_{\tilde{R}}^*+\beta_{\tilde{R}}^*)=\frac{1}{N}\sum_{j=1}^{N}(1-R(t,j))logC_{ij} \tag{7.17c}$$

$$\psi(\beta^*_{\tilde{R}}) - \psi(\alpha^*_{\tilde{R}} + \beta^*_{\tilde{R}}) = \frac{1}{N}\sum_{j=1}^{N}(1 - R(t,j))log(1 - C_{ij}) \tag{7.17d}$$

In order to find the optimal values of the parameters, we use the Newton-Raphson method on the above set of equations. The work described in [87] covers the Newton-Raphson method derivation for maximum likelihood estimation. Once we have relevance computed as a probability value, we can next run the event detection technique for Twitter as well. For every Instagram cluster, we say that the event is true if it has a corresponding set of relevant tweets and for every Twitter cluster we only retain the tweets that got classified as relevant. In this way, we achieve our goal of corroborating the events detected on both the networks.

7.3 EVALUATION

Our evaluation is divided into two sections. The first one is a simulation study on a synthetic data which allows us to verify the formulated Expectation-Maximization (EM) approach. The second one is an actual experiment on the *Social Fusion* service using real world dataset. The details of both the experiments are presented below.

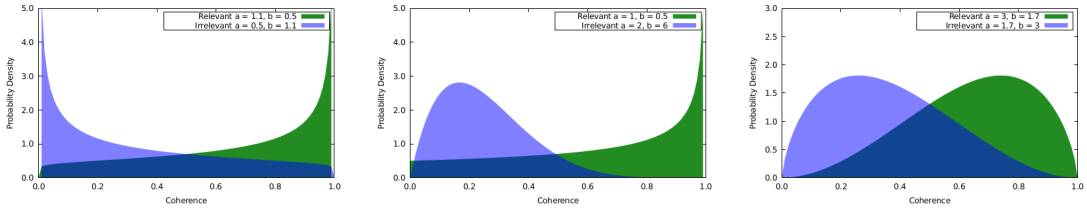
7.3.1 Simulation Study

We use *Python* programming language to code the simulator and the final algorithm. The number (N) of tweets is varied between $\{100, 200, 500, 1000\}$ and the *Coherence* score is obtained using the Beta distribution. For every N tweets, we also pre-define the fraction (d) of tweets that will be labeled as relevant ($R = 1$). The value of d is varied between $\{0.1, 0.2, 0.5\}$. We also use three location names for every run of the simulation. The values for the parameters associated with location names are shown in Table 7.4. For the case of *Street Name* we expect the tweet to be more likely relevant while *State* has an equal chance of being present in relevant and irrelevant tweets. For every N selected tweets we select the fraction d that will be marked as relevant. Once we have the ground truth available for the tweets we next use the the location specifier parameters to link the location names with the tweets depending on the relevance value. Finally we assign the *Coherence* score using the Beta distribution as described above.

Figure 7.2 shows the different parameter (α, β) settings that we consider to generate the *Coherence* score of a tweet given the label. The x-axis is the *Coherence* score and the y-axis

Table 7.4: Location specifier for simulation

Type (L_l)	p_l	q_l
Street Name	0.8	0.2
City	0.6	0.3
State	0.5	0.5



(a) Parameter Setting I (b) Parameter Setting II (c) Parameter Setting III

Figure 7.2: Different Parameter Settings on Beta Distribution for Coherence

is the probability density. The region in *green* color represents the relevant tweet scores and the region in *blue* color represents the irrelevant tweet scores. Parameter setting I is the case where majority of the relevant tweets are concentrated towards the high coherence score and majority of the irrelevant tweets are concentrated towards the low coherence score. Parameter setting II is the case where we keep relevant tweet score distribution same as setting I but change the irrelevant tweet score distribution slightly towards a moderate score range. Finally Paramete setting III is the case where there is a significant overlap between the two distributions. In a real world environment, it would not be surprising to observe this kind of distribution. For each parameter setting, we run our algorithm and compare the expected labels with the original labels. We use three metrics - Precision, Recall, and Accuracy to measure the performance. For every combination of N, d, and *Coherence* parameter settings we run the simulator and the algorithm 10 times and take the average value for each metric.

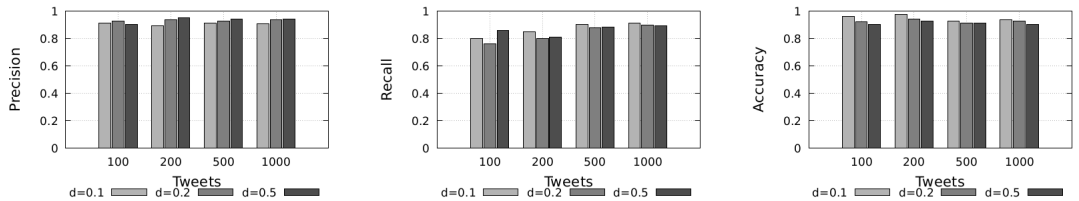
Figure 7.3 is the metric evaluation plot for simulation using the parameter setting I. The first subplot is for precision, which measures the fraction of expected relevant tweets that are correctly labeled (as relevant). The x-axis represent the number of tweets with d fraction of tweets labeled as relevant. The y-axis represents the average precision value over 10 runs for the corresponding settings. The second subplot is for recall, which measures what fraction of relevant tweets that have been identified as such. The x-axis represents the number of tweets with d fraction of tweets labeled as relevant and the y-axis represents the average recall value over 10 runs for the corresponding setting. The third subplot shows the accuracy of the overall algorithm at correctly labeling the relevant and irrelevant tweets. Figure 7.4 is the metric evaluation plot for simulation using the parameter setting II and figure 7.5 is

the metric evaluation plot for simulation using the parameter setting III. For the first two parameter settings, precision and accuracy of the model are well above 90% and recall is above 80% on average. The third parameter setting, which has a significant overlap in the *Coherence* distribution between the two classes generates slighlty lower values in terms of precision and recall compared to the previous parameter settings.

Table 7.5: Average error in parameter estimation

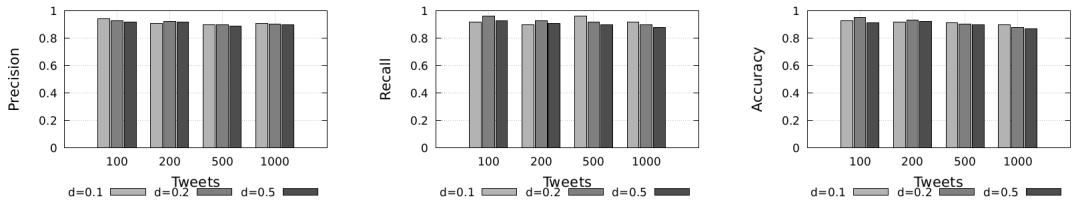
Parameter	**Average Error**
d	0.0122
p_l, q_l	0.0103, 0.0292
α_R, β_R	0.0231, 0.0366
$\alpha_{\tilde{R}}$, $\beta_{\tilde{R}}$	0.0128, 0.0488

In addition to the above comparisons, we also determine the average error in the estimation of the fraction of tweets d, location name parameters, and the Beta distribution parameters used for *Coherence* score. Table 7.5 indicates the average error values over all the runs with different combinations of parameter and tweet count N settings. The average error in estimating the value for different parameters is well within 0.05. Thus, with the help of simulation experiments, we have established the fact that our fusion model using the EM algorithm is very good at identifying the relevance of a given set of tweets associated with an Instagram location and hashtags. It remains to verify that this is indeed the case with real Twitter and Instagram data, which is the topic of the next section.



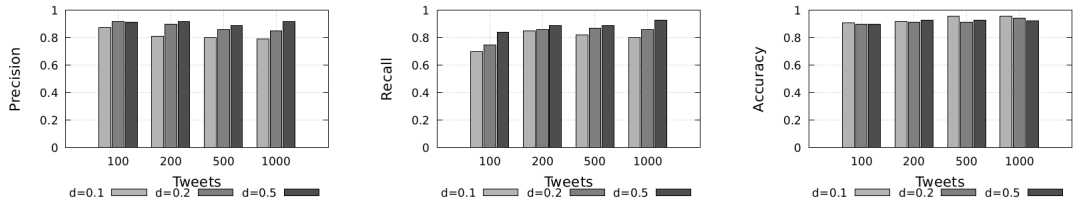
(a) Precision (b) Recall (c) Accuracy

Figure 7.3: Evaluation plots for simulation using Parameter Setting I



(a) Precision (b) Recall (c) Accuracy

Figure 7.4: Evaluation plots for simulation using Parameter Setting II



(a) Precision (b) Recall (c) Accuracy

Figure 7.5: Evaluation plots for simulation using Parameter Setting III

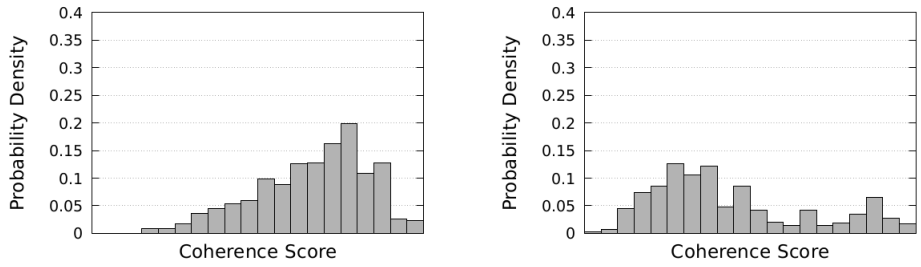
7.3.2 Dataset Experiments

In this section, we discuss evaluation using a real world dataset. We conduct this experiment on the *Social Fusion* service that we implemented to run the fusion algorithm. The *Task Info* file was provided with the keyword "*protest*" which initiated the crawler module to collect data from both Twitter and Instagram. We specifically consider the data logged on to the disk for the time duration February 1, 2016 to February 29, 2016. Table 7.6 summarizes the data collected during this period. For each row we show the total number of tweets, the fraction of tweets that are geotagged (tweets with latitude and longitude information available), and the number of Instagram posts. We retain only those Instagram posts that have location information available.

Table 7.6: Statistics of collected datasets

Dataset	**# Tweets**	**Geotag Fraction**	**Insta-gram posts**
Feb 2016 Week 1	77001	0.0016	1377
Feb 2016 Week 2	78334	0.0012	1424
Feb 2016 Week 3	75639	0.0015	1489
Feb 2016 Week 4	64669	0.0015	1398

We first show that our *Coherence* metric indeed meaningfully distinguishes relevant and non-relevant tweets (to a given Instagram post). To do so we consider an arbitrary sample of clusters generated by the Instagram Event Localizer module along with the candidate tweets. For each cluster we manually label the tweets as relevant and non-relevant. We then generate a frequency distribution of the respective *Coherence* scores which is shown in figure 7.6, where the left subplot corresponds to the relevant tweet scores and the right subplot corresponds to the non-relevant tweet scores. It can be observed that we have two different Beta-like distributions that can be approximated using our model. This plot validates our model assumptions regarding the distribution of *Coherence* of relevant and non-relevant tweets.



(a) Relevant Tweets (b) Non-relevant Tweets

Figure 7.6: Frequency distribution for Coherence scores

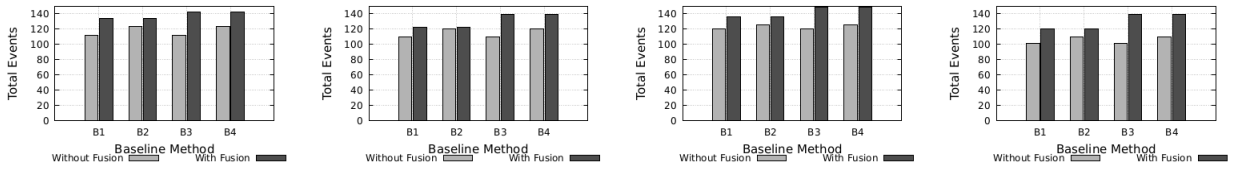
In order to evaluate the performance of our fusion model at event detection, we select two individual event detection techniques for each Instagram and Twitter. The Points of Interest (POI) method described in [58] and the Instagram Event localization (InstaLoc) [85] are used for detecting events on Instagram dataset. The Earthquake detection (TweetEvent) [5] and ClariSense [23] are used for detecting events on Twitter dataset. The evaluation is done using two separate criteria. The first one is the improvement in the amount of detected events against the Instagram detection techniques itself. The second one is the fraction of false positives present in the data against Twitter based detection techniques.

For both the Instagram event detection techniques, we eliminate the below threshold clusters as mentioned in the respective papers but do not follow the same while applying the social fusion method. This allows us to see if the clusters that got eliminated due to lack of support can actually be identified using the fusion method. At the same time we use both the Twitter detection techniques with the same parameters mentioned in the papers, and for the fusion method we only retain those clusters that contain any relevant tweets. With the mentioned techniques we have four pairs of baselines - (i) POI and TweetEvent (B1), (ii) InstaLoc and TweetEvent (B2), (iii) POI and ClariSense (B3), and (iv) InstaLoc and ClariSense (B4).

Figure 7.7 shows the plot for comparison with and without the fusion model for each of the baseline methods in order to find the improvement in the number of total events considering only Instagram detection techniques. There are four subplots for each week in the dataset with x-axis representing the baseline method and the y-axis representing the total detected events. It is evident from the plot that with the help of fusion model we are able to detect more events in general for any selected baseline method. Figure 7.8 shows the plot for comparison with and without the fusion model for each of the baseline methods in order to find the precision considering only Twitter detection techniques. There are four subplots for each week in the dataset with x-axis representing the baseline method and the

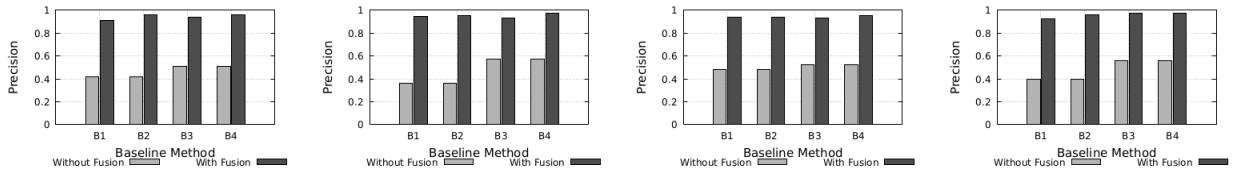
y-axis representing the precision. This plot shows that with the help of fusion model we are able to remove a significant amount of false positives thereby resulting in a higher precision.

The results substantiate the contribution claim made in this paper. Namely, the new fusion based technique offers a better trade-off between false-positives and false-negatives attained using techniques that exploit individual networks separately. We offer significantly fewer false-positives than Twitter-based detection, and significantly fewer false-negatives (i.e., more true positives) than Instagram-based detection, thereby attaining a new point in the aformentioned trade-off space.



(a) Week 1 (b) Week 2 (c) Week 3 (d) Week 4

Figure 7.7: Instagram detection improvement - comparison of different baseline methods with and without fusion method



(a) Week 1 (b) Week 2 (c) Week 3 (d) Week 4

Figure 7.8: Twitter detection precision - comparison of different baseline methods with and without fusion method

7.4 RELATED WORK

Past works [5], [88], [89] have demonstrated that events can indeed be detected using techniques that try to model the behavior of the pattern of extracted features before, during, and after the events. One such technique, is described by the authors of [90] where they apply wavelet analysis on the raw frequency of the words used on Twitter stream and then remove trivial words using the signal correlation. In our previous work [23], we showed how Twitter posts can be used to detect events in an unsupervised fashion. A few papers have also focused on using the geo-tag information available within the content in order to find clusters that have unusual behavior compared to a stored history within a spatial region. Instagram

is another social network where people post pictures and videos with a higher percentage of geo-tag compared to Twitter. The use of locations by the users tends to deliver much credible information. However the amount of such data available is less but considerably higher than Twitter. Various event detection techniques using Instagram have also been studied in the past. One such work described by the authors of [84] has been promising for monitoring city level local events. [58] is the earliest work that uses Instagram to study the urban social behavior and the city dynamics. In our own recent work [85] we showed how to identify events for urban spaces in an unsupervised way. Contrary to all the previous approaches not much work has been done in fusing the same entities (or events) detected in multiple networks thereby enhancing the overall credibility of the events. The work described by authors of [91] considers Twitter and Instagram data to detect and summarize events but they rely on supervised techniques along with geo-location information. However in this chapter we aim at improving event detection by fusing data across multiple networks without any dependence on historical data and detect events with varying degree of popularity. Such events can be effectively retrieved by our fusion method provided enough correlation exists between the data posts on different networks, even when it is hard to detect them by analyzing each network independently. Our work provides an important means to fill the gap in identifying and corroborating the events present in multiple networks.

7.5 SUMMARY

This chapter describes a service which uses a fusion model for integrating data from two different social media platforms, namely, *Twitter* and *Instagram*. The work offers a better trade-off between false-positives and false-negatives compared to approaches that utilize individual networks independently. Specifically, we show that we offer fewer false positives compared to Twitter and fewer false negatives compared to Instagram, offering a new point on the trade-off curve. The motivation for our work comes from the fact that many events offer signatures in multiple networks that can somehow be correlated with the help of intrinsic characteristics such as location mentions and coherence among event descriptions. We design an algorithm that is capable of fusing content from Twitter and Instagram in an unsupervised way. We first study the validity of our model using simulations and evaluate the performanace using precision and recall metrics. Finally, we use real world datasets to confirm the advantages of the fusion approach. We would like to mention that even though our experiments are based on two specific social networks but the same approach can be generalized to other social networks provided they share some feature space (such as words) to generate potential candidates from one of the networks.

CHAPTER 8: CONCLUSIONS

This chapter provides a summary of the research presented in this dissertation and also a follow-up discussion section focused on the things that we learnt from all the research, what still needs to be explored, and future work.

8.1 SUMMARY

The explosive growth in social network content suggests that social sensing might be the future of sensing. In this dissertation, we have presented a series of fundamental approaches to detect, localize, and track events taking place in the real world with the help of data shared by users on social media platforms. A key feature that distinguishes our research from the past literature is the ability to produce quality output in an unsupervised fashion while being language agnostic. We first start with a simple statistical approach to detect and track events using Twitter feeds and evaluate the performance against some well known supervised approaches. We then extend this work to measure the quality of information available within the Twitter network in order to localize the detected events to a specific coarse-grained location. We next move our focus to a different social network, Instagram, that offers pictorial observations with more geo-tagged data as opposed to Twitter. The higher fraction of geo-tagged data allows us to device a new approach to detect and localize events for Instagram. In addition, we also try to estimate the trajectory of the events that are mobile in the geospace. The path estimation problem poses new challenges due to sporadic nature of the observations made by users. Finally, we come up with a new solution that looks at the benefits associated with the two social networks, Twitter and Instagram, in order to perform better than the approaches geared towards the individual networks. A fusion method is applied that offers a trade-off between precision and recall for event detection. The different research works described in this dissertation have also been integrated as a unified software package that is available for public use. We hope that our work provides a base to the future researchers and helps them build a stronger foundation.

8.2 DISCUSSION

The work presented in this dissertation opens up a new research direction for solving simple yet challenging problems pertaining to event detection and tracking using social media. At first, one needs to be specific with the definition of an *event* before buidling up the solution to

handle the different challenges involved. For the purpose of our research, we confine ourselves to events that are associated with temporal and spatial attributes. However, a broad number of event categories can still fulfill the criteria of this definition, such as a person going for a trip from home to office between 8:00 am to 8:30 am. This example has both spatial and temporal attributes due to which it qualifies as an event. Rather, we focus on events that not only have these two attributes but also carry some attraction from the users (acting as human sensors) who are distibuted in the space. The users don't necessarily (i) need to be present at the place of event occurrence, and (ii) report the observations during the event duration. In addition, the events can be irregular in nature, as opposed to a person taking a trip daily from home to office, and also be spread over multiple disjoint regions in a vast space. These are the high level descriptors in our event definition that require us to design techniques capable of identifying the true signals from the user observations.

In Chapter 2 we introduced a language agnostic unsupervised technique to detect and track events using the feeds from Twitter. Our technique is based on a very simple sparsity analysis which lets us to use keyword pairs as signatures to detect events. But the same analysis is valid for even keyword triplet and one might question about why did we stop at keyword pairs? To answer this, we analyzed a subset of the traffic datasets collected from 3 different cities in California as described in Chapter 3.

Table 8.1: Event Signatures Mapping

Signature	**Events per Signature**	**Signatures per Event**
Single Keyword	3.621	1.1579
Keyword Pair	1.1416	1.2725
Keyword Triplet	1.0628	0.4393

The aim was to figure out if can one find a *set* of keywords that would have a one-to-one correspondence with a unique event (to use as event signature)? We considered three types of candidate event signatures; namely, signatures comprised of a single keyword (e.g., "fire"), those comprised of a keyword pair (e.g., "fire", "stadium"), and those comprised of a keyword triplet (e.g., "fire", "stadium", "killed"). We then mapped these signatures to physical events they describe and for each candidate signature we determine (i) how many distinct physical events match a single signature, and (ii) how many distinct signatures match a single physical event? Ideally, both the values should be close to 1 for a one-to-one mapping between events and their signatures. Table 8.1 shows the results for the three types of signatures considered. From the table, it can be seen that keyword pairs come closest to a one-to-one correspondence between independent events and their keyword signatures.

Even though the number of events per signature for a keyword triplet is the least but the keyword pairs create a sparse space that is enough to find the separate instances of the events. Another factor to consider is the computational complexity associated with the candidate signatures which increases monotonically from n for single keywords, to n^2 for keyword pairs, and to n^3 for keyword triplets, assuming the language vocabulary has n distinct words.

There are also a few future work that can be addressed for this component in order to improve the quality of information being generated. One extension is to understand the semantics of the tweets present within an event cluster. Our approach is capable of detecting and demultiplexing the event instances but does not completely answer the $5W$s - *who, where, when, what,* and *why.* A few popular techniques, such as the ones described by the authors of [92, 93], have focused on using tweets to determine these traits. This extension will enhance the output quality of our service thereby providing a storyboard of event instances with rich context. Another extension is to add a multilingual flavor which can improve the accuracy by detecting events in different languages. For example, we can use the discriminative keyword signatures from one language and transfer that knowledge to another language. Word embedding [94] is a popular technique to represent the words from a corpus as vectors in a N dimensional space. The cross-lingual embedding technique is designed using this principle that can find similar words in different languages. Thus, this enables us to represent event signatures from one language as vectors and use this information to find corresponding tweet cluster in a different language.

In Chapter 5 we introduced an event detection and localization technique for Instagram. During the noise elimination step we categorized the events into two types: Single entity (SE) and Multiple entity (ME). The false positive clusters for each type were eliminated on the basis of content (hashtags) similarity between the clusters and the threshold was obtained based using the size of the clusters generated. However, there are a couple of cases which pose some limitation to the current technique - (i) when both SE and ME events occur together within a time frame, and (ii) when an event on the same topic occurs at different locations, for example, a protest event for the same reason in different cities across the United States. One way to handle the first limitation is to consider the distribution of images in time within the clusters. If two or more clusters are related to the same event, then the time distribution of images will mimic each other along with a similarity in the words present within the clusters. For example, if we take a sports event, then the time distribution of images will be similar from the true location (stadium) as well as a false location (a bar) in another city. However, the total number of images at each timestamp will be different due to more people being present in the stadium. If all the goals are scored between 10pm to 10:30pm, then maximum number of images will be shared between these timestamps

irrespective of the location. With the combination of both words and time distribution, we can handle SE and ME events together in a better way. For the second limitation we might have to define a new type of event since it does not qualify as SE or ME. Based on this definition our technique can be extended to analyze the properties (from the data) of such events that distinguish them from the other two types.

In Chapter 6 we presented an approach to estimate the trajectory of events that move across the space during a timeframe. The main contribution of this approach is the removal of background noise associated with population density to generate candidate points. The steps for estimating the trajectory from these candidate points are not fully automated. This can be a possible extension for the future work. One of our challenge while developing the solution was that of observations arriving out of order. But even then we can assume that a larger fraction of observations will be present closer to the actual start point. Under this assumption, we can use the localization technique from Chapter 5 to provide a default start point for the estimation algorithm.

In Chapter 7 we introduced a fusion technique that considers both the social networks, Twitter and Instagram, in order to perform better than the individual networks for event detection. Specifically, we look at the candidate clusters from Instagram and try to find corresponding events from Twitter which leads to a corroboration. During evaluation we showed that the number of detected events increased against Instagram only technique and the number of false positives decreased against Twitter only technique. In this discussion section we take a look at the same dataset and analyze the different techniques for a randomly selected period of 7 days to determine the trade-off in precision versus recall. Table 8.2 summarizes the precision and recall of all three algorithms at event detection. More precisely, since we do not know exactly how many events occurred that might have not been reflected in the dataset, we define recall by referring to the absolute *number of true events* detected. From the table, as expected, we observe that the Instagram only technique has the highest precision but lowest recall, whereas the Twitter only technique has the lowest precision.

Table 8.2: Precision and recall

Technique	**Total# events**	**Precision**	**Recall**
Instagram	54	**87.037%**	47
Twitter	174	63.218%	110
Fusion	211	70.616%	**149**

We also investigate the F1 score for all the techniques, as shown in Figure 8.1. 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 8.1, we observe that although the Instagram technique 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.

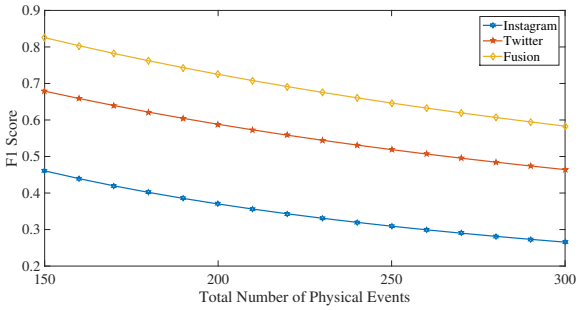


Figure 8.1: F1 score comparison with varied ground truth of total number of events

One future work extension for the fusion technique is to define the coherence in a better way. At present, there is a significant overlap between the two coherence score distributions and this happens due to the fact that we are considering all the words (even if some of them are meaningless). Not all words are equally important and we can perform some simple word statistics before forming the central vector. For example, we can simply keep a count of all words in a timeframe and impose a constraint based on this count as well as word length. This will eliminate the insignificant words and not include common words that are too frequent. An additional technique to may be required to handle synonyms that are used to describe the same event in different ways. We can again use the word embedding as described earlier to understands the relations between words and train on a large training corpus of the dataset before we actually begin the fusion for new data.

In addition to the above discussed limitations and future works, one can also argue about the robustness of the entire system that encompasses the different components presented in this dissertation. For example, *What is the effect of population density on quality of social sensing?* We observed in Chapter 5 how the average number of users vary for different types of events. Much of our focus is on detecting events that occur in urban spaces since we expect a higher rate of smartphone adaptation as well as popularity of social media usage from the corresponding population. There have been a few studies conducted in the past to measure the quality assurance using social media streams. For example, the work [95] takes a look at Twitter feeds to study the effect of different sampling techniques on information diffusion. Similarly, one can conduct a series of new experiments in which the detected events are sampled from different regions to study the relation of different metrics (precision, recall,

F1) with the average number of feeds generated from a particular region. The average count can be based on just the region or on a specific topic within a region. With the latter, we also get an estimate of the regions that are more active on social media platforms for a given topic leading to higher rate of event discovery for that topic.

In this discussion we looked at a few limitations of our work and proposed different ways to handle them. We hope that our proposal will provide a wonderful opportunity for future researchers to make a new contribution in the field of social sensing.

REFERENCES

[1] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, ser. IPSN '12. New York, NY, USA: ACM, 2012. [Online]. Available: http://doi.acm.org/10.1145/2185677.2185737 pp. 233–244.

[2] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "Sociablesense: Exploring the trade-offs of adaptive sampling and computation offloading for social sensing," in *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2030613.2030623 pp. 73–84.

[3] A. Madan, M. Cebrian, D. Lazer, and A. Pentland, "Social sensing for epidemiological behavior change," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, ser. UbiComp '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1864349.1864394 pp. 291–300.

[4] D. Wang, T. Abdelzaher, and L. Kaplan, *Social Sensing: Building Reliable Systems on Unreliable Data*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2015.

[5] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: Real-time event detection by social sensors," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1772690.1772777 pp. 851–860.

[6] 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*. ACM, 2012, pp. 155–164.

[7] 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*. IEEE, 2014, pp. 375–382.

[8] C. C. Aggarwal and K. Subbian, "Event detection in social streams." in *SDM*, vol. 12. SIAM, 2012, pp. 624–635.

[9] R. Parikh and K. Karlapalem, "Et: events from tweets," in *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee, 2013, pp. 613–620.

[10] 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*, ser. ICDE '12. Washington, DC, USA: IEEE Computer Society, 2012. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2012.125 pp. 1273–1276.

[11] 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*, ser. ECIR'13. Berlin, Heidelberg: Springer-Verlag, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36973-5_30 pp. 356–367.

[12] 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*, ser. GREENCOM '12. Washington, DC, USA: IEEE Computer Society, 2012. [Online]. Available: http://dx.doi.org/10.1109/GreenCom.2012.59 pp. 358–367.

[13] 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*, 2016, pp. 513–522.

[14] 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*, ser. CIKM '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2063576.2064014 pp. 2541–2544.

[15] 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.

[16] T. Hua, F. Chen, L. Zhao, C.-T. Lu, and N. Ramakrishnan, "Automatic targeted-domain spatiotemporal event detection in twitter," *Geoinformatica*, vol. 20, no. 4, pp. 765–795, Oct. 2016. [Online]. Available: http://dx.doi.org/10.1007/s10707-016-0263-0

[17] 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*, ser. DCOSS '15. Washington, DC, USA: IEEE Computer Society, 2015. [Online]. Available: http://dx.doi.org/10.1109/DCOSS.2015.14 pp. 179–188.

[18] 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*, no. 4, pp. 32–43, 1999.

[19] M. Jäger, C. Knoll, F. Hamprecht et al., "Weakly supervised learning of a classifier for unusual event detection," *Image Processing, IEEE Transactions on*, vol. 17, no. 9, pp. 1700–1708, 2008.

[20] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in *SIGIR*. ACM, 1998.

[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*. ACM, 2012, p. 21.

[22] 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*, vol. 88, pp. 1–24, 2013.

[23] P. Giridhar, M. T. Amin, T. Abdelzaher, L. M. Kaplan, J. George, and R. Ganti, "Clarisense: Clarifying sensor anomalies using social network feeds," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*. IEEE, 2014, pp. 395–400.

[24] "Twitter," http://www.twitter.com/.

[25] J. H. Lau, N. Collier, and T. Baldwin, "On-line trend analysis with topic models:\# twitter trends detection topic model online." in *COLING*, 2012, pp. 1519–1534.

[26] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[27] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Information Processing in Sensor Networks (IPSN), 2012 ACM/IEEE 11th International Conference on*. IEEE, 2012, pp. 233–244.

[28] D. Wang, M. T. Amin, S. Li, T. Abdelzaher, L. Kaplan, S. Gu, C. Pan, H. Liu, C. C. Aggarwal, R. Ganti et al., "Using humans as sensors: an estimation-theoretic perspective," in *Information Processing in Sensor Networks, IPSN-14 Proceedings of the 13th International Symposium on*. IEEE, 2014, pp. 35–46.

[29] 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*. ACM, 2011, pp. 355–358.

[30] J. Benhardus and J. Kalita, "Streaming trend detection in twitter," *International Journal of Web Based Communities*, vol. 9, no. 1, pp. 122–139, 2013.

[31] C. C. Aggarwal, "A survey of stream clustering algorithms." 2013.

[32] C. Ordonez, "Clustering binary data streams with k-means," in *ACM SIGMOD workshop*. ACM, 2003.

[33] S. Zhong, "Efficient streaming text clustering," *Neural Networks*, vol. 18, no. 5, pp. 790–798, 2005.

[34] 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.

[35] X. Zhou and L. Chen, "Event detection over twitter social media streams," *The VLDB journal*, vol. 23, no. 3, pp. 381–400, 2014.

[36] S. Wang, P. Giridhar, H. Wang, L. Kaplan, T. Pham, A. Yener, and T. Abdelzaher, "Storyline: Unsupervised geo-event demultiplexing in social spaces without location information," in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, April 2017, pp. 83–94.

[37] "Google maps api," https://developers.google.com/maps/documentation/geocoding/.

[38] K. Lee, R. K. Ganti, M. Srivatsa, and L. Liu, "When twitter meets foursquare: Tweet location prediction using foursquare," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ser. MOBIQUITOUS '14. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014. [Online]. Available: http://dx.doi.org/10.4108/icst.mobiquitous.2014.258092 pp. 198–207.

[39] J. Mahmud, J. Nichols, and C. Drews, "Home location identification of twitter users," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, pp. 47:1–47:21, July 2014. [Online]. Available: http://doi.acm.org/10.1145/2528548

[40] S. Chandra, L. Khan, and F. Muhaya, "Estimating twitter user location using social interactions–a content based approach," in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, Oct 2011, pp. 838–843.

[41] O. Ozdikis, H. Oguztuzun, and P. Karagoz, "Evidential location estimation for events detected in twitter," in *Proceedings of the 7th Workshop on Geographic Information Retrieval*, ser. GIR '13. New York, NY, USA: ACM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2533888.2533929 pp. 9–16.

[42] Z. Cheng, J. Caverlee, and K. Lee, "You are where you tweet: A content-based approach to geo-locating twitter users," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ser. CIKM '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871535 pp. 759–768.

[43] W. Li, P. Serdyukov, A. P. de Vries, C. Eickhoff, and M. Larson, "The where in the tweet," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, ser. CIKM '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2063576.2063995 pp. 2473–2476.

[44] P. Serdyukov, V. Murdock, and R. van Zwol, "Placing flickr photos on a map," in *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1571941.1572025 pp. 484–491.

[45] D. Wang, M. T. Al Amin, T. Abdelzaher, D. Roth, C. R. Voss, L. M. Kaplan, S. Tratz, J. Laoudi, and D. Briesch, "Provenance-Assisted Classification in Social Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, pp. 624–637, Aug. 2014.

[46] P. Giridhar, T. F. Abdelzaher, L. M. Kaplan, and J. George, "On quality of event localization from social network feeds," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom 2015 Workshops, St. Louis, MO, U.S.A., March 23-27, 2015*.

[47] "United nations - world population statistics," https://www.un.org/development/desa/en/news/population/world-urbanization-prospects.html.

[48] "Smartphone user statistics," http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[49] "Instagram," https://www.instagram.com.

[50] "Instagram statistics," http://mediakix.com/2016/03/top-instagram-statistics-you-should-know/#gs.GiCN2lw.

[51] "1989 world tour wikipedia page," https://en.wikipedia.org/wiki/The_1989_World_Tour.

[52] "Maroon v tour wikipedia page," https://en.wikipedia.org/wiki/Maroon_V_Tour.

[53] "Marathon annual report," http://www.runningusa.org/index.cfm?fuseaction=news.details&ArticleId=332.

[54] "Top us cities by population," http://www.infoplease.com/ipa/a0763098.html.

[55] "Best fall 2015 marathons," http://dailyburn.com/life/fitness/best-marathons-fall/.

[56] "Tornadoes 2015 wikipedia page," https://en.wikipedia.org/wiki/Tornadoes_of_2015.

[57] X. Gao, J. Cao, Q. He, and J. Li, "A novel method for geographical social event detection in social media," in *5th Intl Conference on Internet Multimedia Computing and Service*, ser. ICIMCS '13. New York, NY, USA: ACM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2499788.2499819 pp. 305–308.

[58] 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*, May 2013, pp. 123–132.

[59] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, "An empirical study of geographic user activity patterns in foursquare," 2011. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2831

[60] J. Cranshaw, E. Toch, J. Hong, A. Kittur, and N. Sadeh, "Bridging the gap between physical location and online social networks," in *Proceedings of the 12th ACM UbiComp*, ser. UbiComp '10, 2010. [Online]. Available: http://doi.acm.org/10.1145/1864349.1864380

[61] J. Cranshaw, R. Schwartz, J. Hong, and N. Sadeh, "The livehoods project: Utilizing social media to understand the dynamics of a city," 2012. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM12/paper/view/4682/4967

[62] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, "Exploiting semantic annotations for clustering geographic areas and users in location-based social networks." *The social mobile web*, vol. 11, p. 02, 2011.

[63] N. Hochman and L. Manovich, "Zooming into an instagram city: Reading the local through social media," *First Monday*, vol. 18, no. 7, 2013. [Online]. Available: http://www.firstmonday.org/ojs/index.php/fm/article/view/4711

[64] 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.

[65] Y. Mejova, S. Abbar, and H. Haddadi, "Fetishizing Food in Digital Age: #foodporn Around the World," *ArXiv e-prints*, Mar. 2016.

[66] 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*. Springer, 2015, pp. 276–280.

[67] R. van Zwol, "Flickr: Who is looking?" in *Proceedings of the IEEE/WIC/ACM*, ser. WI '07. Washington, DC, USA: IEEE Computer Society, 2007. [Online]. Available: http://dx.doi.org/10.1109/WI.2007.60 pp. 184–190.

[68] X. Liu, R. Troncy, and B. Huet, "Using social media to identify events," in *Proceedings of the 3rd ACM SIGMM Intl. Workshop on Social Media*, ser. WSM '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2072609.2072613 pp. 3–8.

[69] S. Papadopoulos, C. Zigkolis, Y. Kompatsiaris, and A. Vakali, "Cluster-based landmark and event detection for tagged photo collections," *IEEE Multimedia*, vol. 18, no. 1, pp. 52–63, 2011.

[70] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003. [Online]. Available: http://doi.acm.org/10.1145/958491.958509 pp. 150–161.

[71] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee, "Using adaptive tracking to classify and monitor activities in a site," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ser. CVPR '98. Washington, DC, USA: IEEE Computer Society, 1998. [Online]. Available: http://dl.acm.org/citation.cfm?id=794191.794779 pp. 22–.

[72] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," *IEE Proceedings - Radar, Sonar and Navigation*, vol. 146, no. 1, pp. 2–7, Feb 1999.

[73] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings F - Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, April 1993.

[74] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999. [Online]. Available: http://www.jstor.org/stable/2670179

[75] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb 2002.

[76] W. Kim, K. Mechitov, J. Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, April 2005, pp. 301–308.

[77] M. Naaman, "Geographic information from georeferenced social media data," *SIGSPATIAL Special*, vol. 3, no. 2, pp. 54–61, July 2011. [Online]. Available: http://doi.acm.org/10.1145/2047296.2047308

[78] Z. Yin, L. Cao, J. Han, J. Luo, and T. Huang, *Diversified Trajectory Pattern Ranking in Geo-Tagged Social Media*, pp. 980–991. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611972818.84

[79] H. Q. Vu, G. Li, R. Law, and Y. Zhang, "Tourist activity analysis by leveraging mobile social media data," *Journal of Travel Research*, vol. 0, no. 0, p. 0047287517722232, 0. [Online]. Available: https://doi.org/10.1177/0047287517722232

[80] T. T. Nguyen, D. Camacho, and J. E. Jung, "Identifying and ranking cultural heritage resources on geotagged social media for smart cultural tourism services," *Personal and Ubiquitous Computing*, vol. 21, no. 2, pp. 267–279, Apr 2017. [Online]. Available: https://doi.org/10.1007/s00779-016-0992-y

[81] C. Junker, Z. Akbar, and M. Cuquet, "The network structure of visited locations according to geotagged social media photos," *CoRR*, vol. abs/1704.04739, 2017. [Online]. Available: http://arxiv.org/abs/1704.04739

[82] I. L. Johnson, S. Sengupta, J. Schöning, and B. Hecht, "The geography and importance of localness in geotagged social media," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858122 pp. 515–526.

[83] K. Xie, C. Xia, N. Grinberg, R. Schwartz, and M. Naaman, "Robust detection of hyper-local events from geotagged social media data," in *Proceedings of the Thirteenth International Workshop on Multimedia Data Mining*, ser. MDMKDD '13. New York, NY, USA: ACM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2501217.2501219 pp. 2:1–2:9.

[84] C. Xia, R. Schwartz, K. Xie, A. Krebs, A. Langdon, J. Ting, and M. Naaman, "Citybeat: Real-time social media visualization of hyper-local city data," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14 Companion. New York, NY, USA: ACM, 2014. [Online]. Available: http://doi.acm.org/10.1145/2567948.2577020 pp. 167–170.

[85] 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.

[86] M. T. A. Amin, S. Li, M. R. Rahman, P. T. Seetharamu, S. Wang, T. Abdelzaher, I. Gupta, M. Srivatsa, R. Ganti, R. Ahmed, and H. Le, "SocialTrove: A self-summarizing storage service for social sensing," in *International Conference on Autonomic Computing (ICAC'15)*. IEEE, July 2015, pp. 41–50.

[87] C. E. B. Owen, "Parameter estimation for the beta distribution," 2008.

[88] J. Chae, D. Thom, H. Bosch, Y. Jang, R. Maciejewski, D. S. Ebert, and T. Ertl, "Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition," in *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*. IEEE, 2012, pp. 143–152.

[89] H. Becker, M. Naaman, and L. Gravano, "Learning similarity metrics for event identification in social media," in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, ser. WSDM '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1718487.1718524 pp. 291–300.

[90] J. Weng and B.-S. Lee, "Event detection in twitter." *ICWSM*, vol. 11, pp. 401–408, 2011.

[91] C. Xia, J. Hu, Y. Zhu, and M. Naaman, "What is new in our city? a framework for event extraction using social media posts," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2015, pp. 16–32.

[92] C. Wang, M. Danilevsky, N. Desai, Y. Zhang, P. Nguyen, T. Taula, and J. Han, "A phrase mining framework for recursive construction of a topical hierarchy," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2487575.2487631 pp. 437–445.

[93] W. Guo, H. Li, H. Ji, and M. Diab, "Linking tweets to news: A framework to enrich short text data in social media."

[94] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, http://is.muni.cz/publication/884893/en. pp. 45–50.

[95] M. De Choudhury, Y.-R. Lin, H. Sundaram, K. S. Candan, L. Xie, and A. Kelliher, "How Does the Data Sampling Strategy Impact the Discovery of Information Diffusion in Social Media?" in *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, May 2010. [Online]. Available: http://www.public.asu.edu/\~{}mdechoud/pubs/icwsm_10.pdf