LOCALIZATION OF EVENTS USING UNDERDEVELOPED MICROBLOGGING DATA

by

Usman Anjum

MSc Telecom, University of Maryland College Park, 2011

Submitted to the Graduate Faculty of

the School of Computing & Information in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING & INFORMATION

This dissertation was presented

by

Usman Anjum

It was defended on

August 06 2021

and approved by

Dr. Prashant Krishnamurthy, Department of Informatics & Networked Systems

Dr. Vladimir Zadorozhny, Department of Informatics & Networked Systems

Dr. Martin Weiss, Department of Informatics & Networked Systems

Dr. Mai Abdelhakim, Department of Electrical & Computer Engineering

Dissertation Advisors: Dr. Prashant Krishnamurthy, Department of Informatics & Networked Systems,

Dr. Vladimir Zadorozhny, Department of Informatics & Networked Systems

LOCALIZATION OF EVENTS USING UNDERDEVELOPED MICROBLOGGING DATA

Usman Anjum, PhD

University of Pittsburgh, 2021

Event localization is the task of finding the location of an event. Events are defined as significant one-time occurrences that show notable deviation from expected or normal behavior. Event localization has been studied in many domains including medical data, internet-of-things (IoT), sensor data and microblogging/social media domain. In this dissertation we focus on event localization in the microblogging domain. The data in the microblogging presents a unique challenge in that it is *underdeveloped*. Underdeveloped data has low reliability and sporadic delivery slate. Since, microblogging data is underdeveloped it provides subjective and incomplete information, which is unsuitable for event localization. We propose enrichment methods for underdeveloped data that would make the data more suitable for event localization. Our enrichment methods include disaggregation, semantic filtering and data generation using top-down and bottom-up approaches. Once the data is enriched, we identify event signatures that are specific to an event. We find both explicit and latent event signatures within the enriched data. Using these signatures an event can be efficiently localized. We use generated data and data collected from Twitter to test our enrichment methods and implement event localization strategies.

Table of Contents

List of Tables

List of Figures

1.0 Introduction

Event localization is the task of determining the *location or spot*[1] of an event using a sequence of data that includes the occurrence of such an event. The "location" of an event may be in time, space or another dimension in the sequence of data points. Events, for this paper, are defined as significant one-time occurrences that show significant deviation from expected or normal behavior in the sequence. Events have been classified as unexpected or expected [4, 46]. Unexpected events are rare or at least infrequent occurrences that are unpredictable/ unidentified/ unscheduled or unknown. Prior knowledge about event type, time and location may not be readily available until well after the event has occurred which presents a significant challenge in geographically localizing it. Event localization shares similarity with rare event detection. Rare event detection has been used to find where a low frequency event may have occurred in time series data (like sensor data [27]) and even in stock and insurance data [65].

Identifying the spot of an event in a sequence is important in many domains including medical data, internet-of-things (IoT), sensor data and microblogging/social media domain. In the medical domain identifying the event's spot corresponds to *changes* in a patient's medical condition – e.g., it is used in EKG data to find irregular heart-beats or brain activity. Pattern changes may be used in the human rights documents to identify social violations and conflicts ([7], [48]). In IoT and sensor domains, event localization may be used to find the location of malicious or faulty sensors. Recently, inferences about the pandemic have been made using **aggregate** date from sequences of temperature readings by smart thermometers [70], counts of searches [17] or conducting tests on samples obtained from groups of people [51]. In the microblogging domain a *"crowd sourced"* event localization approach has been used to find the location of real-world events. Such real-world events include natural disasters (e.g., earthquakes, floods, fires and typhoons), infectious disease outbreaks, traffic incidents, riots, terrorist acts, etc., which may be unpredictable. Crowd sourced event localization

[1] The term *spot* is used somewhat loosely here to denote the location in a sequence. Later we clarify it to be the estimate of distance that enables us to determine the geographic location or for *Localization*.

monitors the microblogging pattern of the people, who are considered as *"social sensors"*. The people respond to an event by changing their microblogging behavior. Other kinds of aggregated/coarse data have been recently used for event detection (i.e., the event *happened*, but not geographical localization) [58].

The task of localization requires specific data characteristics/features, e.g. spatial localization requires geographical coordinates, etc. However, microblogging data is *underdeveloped* that makes these features not always available. We define undeveloped data in two dimensions. These two dimensions are summarized in terms of *reliability* and *delivery slate*. Data can have high or low reliability and data delivery can have regular or sporadic slate. Table 1 summarizes the differences between high and low reliability and regular and sporadic delivery slate.

High reliability commonly means that the data source has little human involvement and provides objective information [2]. The data format is consistent and homogeneous. Relevant features for localization would be obvious and readily available. On the other hand, low reliability data is provided from sources that have extensive human involvement which means that the data provided would be subjective. The data format would be inconsistent and heterogeneous. Relevant features for localization would be hidden and usually not available. Medical and sensor data from well-defined fault-free instruments have high reliability unlike microblogging and fake news data [42], created by humans or with human involvement, have low reliability.

Data delivery slate can be regular or sporadic. Regular data delivery means that data refresh frequency can be controlled and there is control on the scheduling of data. The data source would provide large amount of data at continuous and regular intervals. The data would be fine-grained to meet application needs. Sporadic data delivery slate means that data refresh frequency cannot be controlled and there is no control on data scheduling. There is limited data available and the data slate is irregular. The data would be aggregated and/or coarse-grained. For example, as explained below, medical and fake news data have regular delivery slate and microblogging and sensor data have sporadic delivery slate.

[2] We do not consider misuse or malfunctioning equipment

Table 1: High/Low Reliability & Regular/Sporadic Delivery Slate Differences

Reliability		**Delivery Slate**	
High	**Low**	**Regular**	**Sporadic**
Source has little to no human involvement	Source has human involvement	Source delivery slate is controllable	Source delivery slate is uncontrollable
Source provides objective information	Source provides subjective information	Source provides unlimited amount of data	Source provides limited data
Structured data with specific schema	Unstructured data with no specific schema	Data slate is continuous and regular	Data slate is sporadic and irregular
Relevant features for localization are available	Relevant features for localization are unavailable	Data is fine-grained	Data is aggregated and coarse-grained
Medical & sensor data	Microblogging & fake news data	Medical & fake news data	Microblogging & sensor data

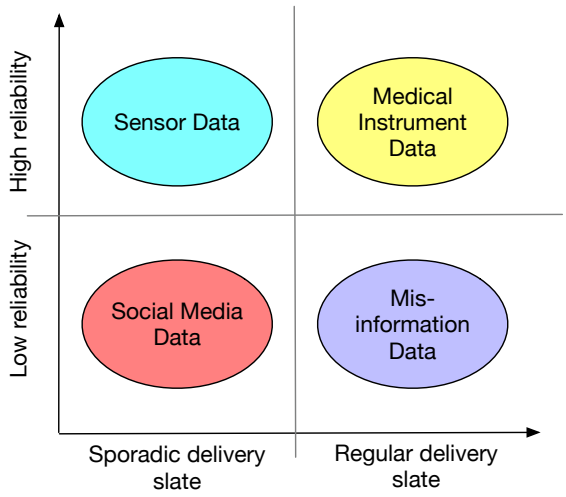


Figure 1: Dimensions of Data

Figure 1 summarizes how the different data types are placed according to their reliability and delivery slate.

Both sensor and medical data sources provides information that is objective and based on the measurement of a parameter. Microblogging and fake news data involves humans as sources, which means that the data could intentionally or unintentionally be distorted and provides subjective information mixed with personal emotions. For example, microblogging data usually does not contain complete spatial information (like latitude and longitude) that is essential for accurate event localization. Use of location anonymization techniques for privacy preservation makes the latitude and longitude (*geotags*) not readily available. Alternatively, researchers have used location name [46] found in the message (also called place name) and user location found in their profiles to localize an event. But again, this information is not reliable as users may use multiple locations and may be slow in updating location information, which means that the location in the profile and the user location may not be consistent. Users may also create false messages or put incorrect location information which further reduces the data reliability [4]. Microblogging messages, like in Twitter, are short in length and can contain ambiguous words making it hard to obtain correct information from the messages. Another factor that effects data reliability is lack of historic data. Since we

wish to localize rare events, we cannot be sure if an event signature has been identified or not. Hence, the features required for localization would not be apparent and there may be latent features in the data that would have to be discovered for localization.

Medical data has quasi-periodically delivery pattern and the data stream can be controlled. Fake news data relies on regular update about news by *reporters*[3]. Sensor and microblogging data have sporadic data delivery (refresh rate). Sensors, those of low power, have limited resources (like battery and memory) and cannot send data regularly. Most of the times sensors send data when there is a trigger, e.g. motion detection sensors sending update about motion only when motion occurs. Similarly, microblogging data also has sporadic data delivery slate. The delivery of microblogging data is sporadic because not all users are sending out event related data and even if they do, it may not be about an event and users may send messages only when it is convenient. Factors like number of users who actively send microblogging message, time of day, population density, significance of the microblogging message or event, etc., influence how frequently people may send out standard and event-related messages. Hence, there would sometimes be a limited data available for localization. In addition, **only sequences[4] of *counts* of microblogging messages and perhaps even then, only aggregate counts of microblogging messages with minimal metadata may be all that is available for localizing an event.**

[3] *Reporters* includes citizens and news organization employees

[4] These sequences are different when they are scraped at different reference spatial points.

1.1 Dissertation objective

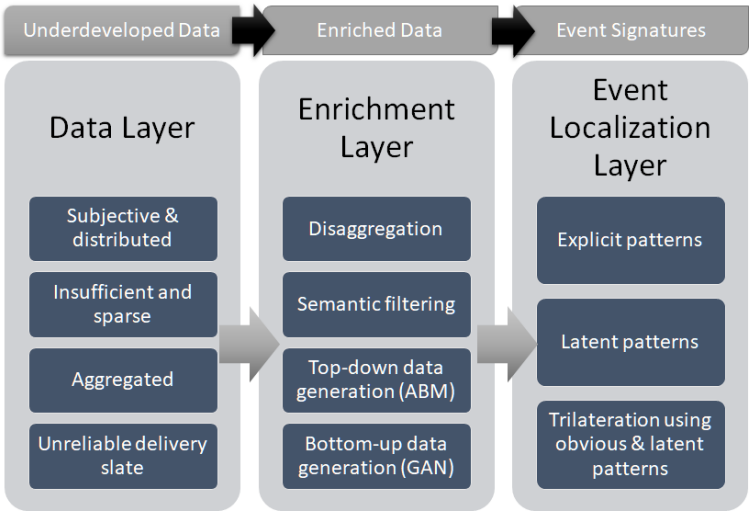


Figure 2: Dissertation Workflow

The objective and the work of the dissertation can be summarized using Figure 2. We aim to enrich underdeveloped data in the *data layer* that is subjective, insufficient/sparse, aggregated and with a unreliable delivery slate and to use the enriched data for event localization. We use microblogging service of Twitter as a use case for this work. Hence, we use the term microblogging messages and tweets interchangeably.

We define an *enrichment layer* that implements novel techniques to enrich the data. The enrichment methodologies include disaggregation, semantic filtering and augmentation. The objective of disaggregation is to obtain fine-grained data from aggregated (coarse-grained) data. We will show later that, as may be expected, the more fine-grained the data is the more accurate will be the localization. We propose different metrics to measure how well the disaggregated fine-grained data reconstructs patterns to match the original data. The next part of enrichment layer is semantic filtering (or simply filtering). The objective of semantic filtering is to enrich the data by filtering out unwanted and noisy information via semantic information like text similarity of microblogging messages or simply low pass filters which will improve event localization accuracy. The final phase of the enrichment layer is data augmentation. The objective of the augmented data is to provide more control on the delivery

slate and the reliability of the data and create data that would more accurately reflect real world microblogging data. The augmented data can also be used for localization of events. We augment data using top-down and bottom-up data generation techniques. The top-down technique generates data using an agent-based model. The agent-based model (ABM) uses different parameters to emulate microblogging behavior. With ABM, we identify different parameters that may affect the real-world microblogging data. On the other hand bottom-up data generation uses machine learning methods, like generative adversarial networks (GAN), to generate copies of the original data by learning its patterns and distribution. Our next objective is to show how the enriched data can be used for localization.

We define the *event localization layer* to implement methods for event localization. We perform event localization on data collected from microblogging service of Twitter and augmented data. To perform localization, it is necessary to extract useful information from the data. An event is localized first by identifying event signatures in the enriched data. Some of the signatures are easy to observe (explicit signatures) and some are hidden within the data (latent signatures). Explicit signatures can be discovered through peaks in the enriched data and latent signatures can be discovered using machine learning methods, like long short-term memory (LSTM)/neural networks, etc. Due to the limited real-world data related to an event, we use enriched data to learn event patterns. Finally, we combine multiple explicit or latent signatures to do trilateration which gives us improved event localization accuracy.

1.2 Thesis statement

In this thesis we aim to examine the challenge of using underdeveloped for event localization and suggest approaches to address the situation. We use the microblogging service of Twitter to collect data.

More specifically this thesis aims to address the following research questions:

Question 1. How can underdeveloped data, that is, data with low reliability and sporadic delivery slate, be enriched so the localization accuracy can be improved?

What techniques can be used to overcome low reliability sporadic delivery slate limitations

of the data to extract useful information from the data? For this purpose we propose the enrichment layer in Figure 2. *How well is the data enriched?* We define the metrics that measure how well the data patterns match in the enriched data and the original data. We use both augmented data and data collected from Twitter for implementation of our enrichment methodology. We further implement semantic filtering to enriched data to improve localization.

Question 2. How to extract useful information from enriched underdeveloped data that can be used for event localization?

What are the latent and explicit patterns in the data that are associated with an event? Using both enriched data and data collected from Twitter we identify patterns that are specific to an event. Some of these patterns are explicit. For example, peaks found in a counts of microblogging messages (tweets) varying with distance from a specific coordinate within a time window can be used as indication of an event. Semantic filtering enriches the data by removing peaks that may not correspond to an event. Latent patterns are identified using machine learning techniques, like LSTM. We use latent and explicit patterns for event localization and further improve localization accuracy by implementing trilateration.

1.3 Methodology

To answer the first question we envision a scenario where there are multiple reference coordinates distributed throughout a geographic region that collect number of tweets that change with distance from reference coordinate within a time window. The collected data are underdeveloped, i.e., aggregated (coarse-grained), sparse and insufficient, subjective/distributed and with an unreliable delivery slate. Using such data for localization is challenging. We enrich the underdeveloped data and then localize an event. It is observed that the event localization accuracy is better with enriched data.

Data enrichment is done through disaggregation, i.e., obtaining fine-grained data from coarse-grained data, data augmentation and filtering techniques. Disaggregation gives us fine-grained information which gives us more accurate event locations. Data augmentation

allow us to measure how well enriched data allows us to perform localization especially when there is no data with ground truth. Filtering further enriches the data by emphasizing the event related patterns and removes any unwanted information in the enriched data. We implemented different filters and compare how they effect the patterns in both data from Twitter and augmented data.

For the second question, we begin by defining how events are manifested in enriched data. Using those definitions, we find explicit and latent patterns within the data. We find explicit patterns through heuristic observation of real world data, e.g., one explicit pattern of events is peaks. We use machine learning and deep learning methods, like LSTM, to find latent patterns within the data. Machine learning methods can automatically learn patterns that are hidden in the data and hence, would be most suitable to learn any latent patterns. The patterns are used to find the location of an event. The localization accuracy is improved by implementing trilateration. We also compare the trilateration accuracy in both agent-based and GAN augmented data.

1.4 Dissertation Outline

The dissertation is organized as follows: In Chapter 2, we provide background and literature review. In Chapter 3, we define our data enrichment techniques. In Chapter 4, we introduce the different data sets used for analysis and compare the TBAM, GAN and the data collected from Twitter. In Chapter 5, we show how event patterns are detected. In Chapter 6 the patterns obtained from enriched data is used to localize an event. Finally, in Chapter 7 we conclude.

2.0 Background and Literature Review

The literature review is divided into three sections. The first section reviews related work on data disaggregation and data fusion. The second section explores literature on data augmentation and data imputation. The final section reviews literature on event localization.

2.1 Data fusion and Disaggregation

Data disaggregation is a special kind of data fusion task. Data fusion techniques combine data from multiple data sources to reconstruct a target sequence as accurately as possible. Data fusion has been used widely in many applications like guidance for autonomous vehicles, remote sensing, robotics and medical application ([59], [25]). Liu et. al. ([38] addressed the challenge of disaggregation as recovering a time sequence of counts from possibly overlapping aggregated reports. For example, this task considers the question of how to obtain daily counts of people infected with flu when only monthly or weekly sums are available. They proposed a method called H-Fuse that formulates the disaggregation task as a linear equation and added domain knowledge to improve reconstruction. The work in [38] was modified in [60] such that the disaggregation task was presented as a gray-box model with the aim of reconstructing the time series of counts (and the gray-box model) parameters when only the aggregated counts and the gray-box "model" is known. They used the susceptible-infected-susceptible (SIS) model as their gray-box model to reconstruct epidemics time series data with more accuracy than H-Fuse. H-Fuse [38] was expanded upon in [72] by adding an *annihilating filter technique* as domain knowledge to improve reconstruction of time sequence counts. An annihilating filter is a linear shift-invariant operator which completely suppresses a series. The filter coefficient that completely suppressed a series would then be the approximate for the disaggregated time sequence of counts. The work in Homerun [3] assumed that the aggregated reports are very sparse and represent a time series as a sum of cosine signals. They use Discrete Cosine Transform (DCT) to find the coefficients that can

represent the disaggregated time sequence of counts. They also added domain constraints to improve disaggregation accuracy. These papers have focused on the temporal domain. We focus mainly on spatial series data, within a time window.

2.2 Data Imputation and Augmentation

A part of the enrichment process for underdeveloped data is to use synthetic data. Synthetic data has been used in prior literature for *data augmentation* and *data imputation*. Data augmentation and imputation are recent techniques with very little work found in literature.

Data augmentation has been used in previous literature for image (e.g face data augmentation in [64]), speech and natural language processing (NLP) [11] speech and time-series data to reduce overfitting ([56], [66]). Augmentation increases the size of the training data set by geometric and color transformations and deep learning techniques like Generative Adversarial Networks (GAN). Augmentation also alleviates the issue of class imbalance, which is a data set with skewed majority to minority sample ratios [56]. The effect of different augmentation techniques on time-series data was evaluated in [31] and created a guide for researchers and developers to help select the appropriate data augmentation method for their applications. Generative adversarial networks (GAN) was used to generate synthetic images in the medical domain ([6], [16], [26]). These works generated images of CT images of liver lesions ([6], [16]) and MR images [26] which were very close in comparison to the real data. Similarly, cycle-Consistent Generative Adversarial Networks (CycleGANs) was proposed as an image classification method to detect floods using images found in social media [50]. An agent-based model simulator called *paysim* was created to simulate mobile money transaction and generated data that is similar to the original data set [39].

Data imputation is the task of estimating missing values in a data set. Data imputation was done to find data missing values in traffic data arising from sensor damage, malfunction, or transmission errors, etc., using low-rank matrix decomposition.

Most work on data imputation has focused on using GAN for data imputation by slightly varying its structure or the loss function [33]. Two of the prominent works that have used GAN as a method for finding missing values in time-series data are found in [40] and [74].

With regards to the use of Agent-Based Modelling (ABM) to understand user tweeting behavior there is very little literature that has addressed the issue of using generated data. Some research studies have used Agent-Based Modelling (ABM) but none of the works focused on how user tweeting behavior changes when an event occurs. In [10], ABM was used to investigate how information was spread during the 2011 Wenzhou train crash through the Sina Weibo. They use an ABM framework to compare information diffusion through word-of-mouth and mass media and to determine which is a more significant means of spreading information when it comes to social media. ABM has been used to create an information propagation model to study how retweeting occurs ([71], [49]). In [49], a retweeting model was created based on two main parameters: the influence of the user (number of followers a user has) and the time at which the tweet was received. In [71], the retweeting model was based on the susceptible-infected-refractory (SIR) model. Similarly in [19], ABM was used to study user behavior in a social network. The model was created to predict user's sentiment and if they choose to forward, reply or do nothing about a topic.

2.3 Event Localization

Localization has been widely used in locating unknown sensors ([73], [61], [12]), and in global positioning systems (GPS). Localization has also been extended to social networks and to find the location of an event within a time window. A list of surveys that summarized the work done on event detection in the microblogging domain are [4], [62], [9], [18], [30], and [28]. The survey in [4] identified challenges and limitations arising in event detection and localization methodology due to use of content in tweets like, ambiguous texts in tweets, and lack of relevant data. In [62], an advanced systemic literature review is presented on methodologies, applications and use cases of Twitter as a Location-Based Social Network. In [9], a taxonomy of event detection in social media is created and the different methods are

classified under type of event, type of detection method (supervised or unsupervised), and if the event detected is a new event or an old event. Garg et. al. [18] focused on the different types of data sets (images, texts, audio, etc.) in social media used for event detection. The survey in [30] covers approaches, the challenges, and the benefits of different approaches for use of social media messages for detecting emergency events (like natural disasters, etc.). Finally, in [28] a survey on methods for real-time detection of events is done.

The location of an event can be a single geographical coordinate, a geographic area or a name identified by users in their tweets [46]. In [1] the region of interest is divided into cells and then keywords are extracted based on their temporal and geospatial properties and then clustered. A cluster is defined as a localized event if its keywords have a high frequency, is a member of a cluster for a long time and was recently bursty in the same cluster. The *Eyewitness* [34] and its real-time version [8] algorithm looks through a corpus of geotagged tweets over localized regions for unusual spikes in tweet counts. They divide the area of interest into triangles and use time periods of different lengths. An event is defined as a peak above a baseline tweet count, which is obtained through regression. However, during pre-processing they remove retweets and repeat tweets which, we believe, may play a significant role in event detection. Furthermore, they do not discuss spurious peaks which we show later can cause inaccuracies in results. Peaks were also used to identify an event in [5] and they used a Semantic Decay Filter (SDF) to eliminate random peaks. The SDF removes peaks that have low similarity between tweet texts. The work in [36] presented a geo-social event detection method focusing on the geographical regularities of local crowd behaviors to detect events. They implemented their method using a fixed time window and their geographic grids are created based on a clustering-based space partition method.

In [14], similar words are extracted in a stream of tweets to create a time series and a wavelet-based method is applied to measure the similarity. Then different clusters are created based on the similarity to find time and location of events. Shao et. al [55] used keywords to create graphs to find the location and time of an event.

Sakaki et. al. [52] used tweets to find epicenter of an earthquake and trajectory of typhoons. First, semantic analysis on the texts in the tweets is done to extract the relevant tweets. The authors assume that tweets follow an exponential distribution with time which

is used to estimate the probability of occurrence of an event. Next, they use the geographic coordinates of the tweets to estimate the location of an event using Kalman filters and particle filters. Kalman filters assume a Gaussian distribution of the coordinates and particle filters look at how the users are distributed in a region. Later, we use what a disaggregation approach that uses what is called as an observation matrix. The Kalman and particle filters can be considered as additional domain information to the observation matrix to better estimate an event. Another work ([44]) estimates an event's location by assigning probabilities using Dempster–Shafer (DS) theory based on geotags, texts in tweets and user profile. The location of the events was found by clustering. However, they only considered two levels of granularity and require coordinates and names for assigning probabilities. [44] was extended in [45] to incorporate real-time tweets. Similar to [44], the work in [54] uses Dempster–Shafer (DS) to find the coarse-grained information (like city name) and fine-grained information (coordinates of the event). They focus mostly on traffic accidents. In our work we do not rely on the texts and information found within the tweets. In addition, the granularity of the data is defined by the size of annular rings and can be as fine-grained as the coordinate of an event instead of only having only two information granularity (city-level and coordinate of event).

2.4 Relationship with this Dissertation

In this dissertation, we localize an event by finding patterns in counts of tweets rather than using the content of the tweets. We address challenges that have not been considered in previous literature, like aggregated data and lack of data. We use machine learning techniques, like LSTM to find patterns in tweets rather than using clustering or classification. We also use filtering to remove any patterns not related to an event. To our knowledge, our work is first to use enriched data as a method to deal with aggregated data, lack of data and class imbalance and use the enriched data for localization. We augment the data using agent-based model and GAN. We use augmented data, in lieu of real data, for event localization. Use of agent-based model to simulate people's tweeting behavior changes around an event is also something new that has not been considered in previous literature.

3.0 Enrichment of Underdeveloped data

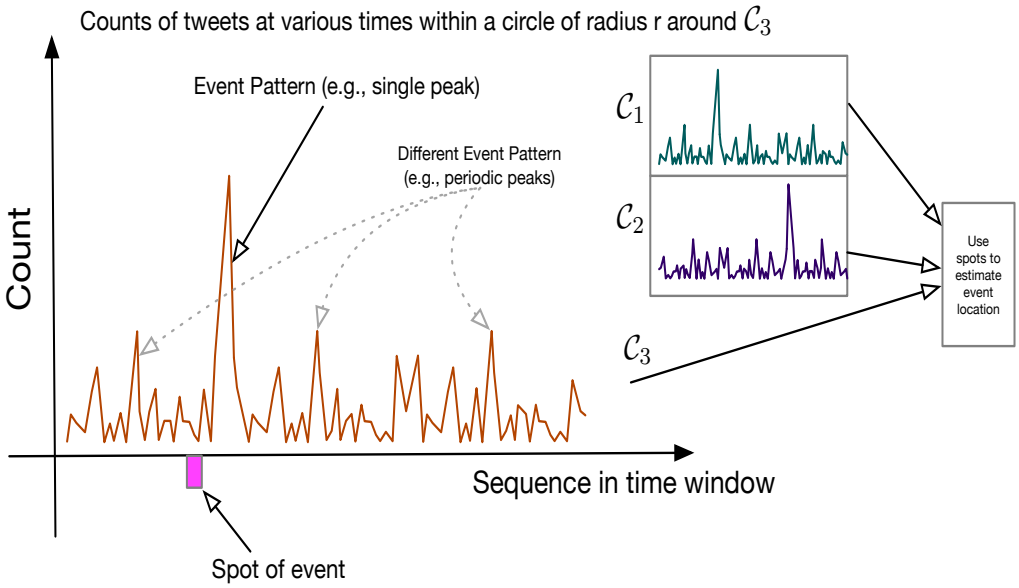
The objective of this chapter is to focus on research question 1 and *describe the experimental design of the data enrichment techniques* for underdeveloped data before the data can be used for event localization. We begin by describing how data is aggregated along both the temporal and spatial domain. Once we understand data aggregation, we describe the disaggregation methodology.

Using the disaggregated data we find event related patterns within the data sequences. We find both obvious (explicit/manifest) and latent event patterns. An explicit event pattern observed in Twitter data collected for several events are *peaks*. Latent patterns are learned through deep learning methods like *long short-term memory (LSTM)*. Different filtering techniques are proposed to assess the peaks and remove any peaks that may be falsely identified as event related patterns.

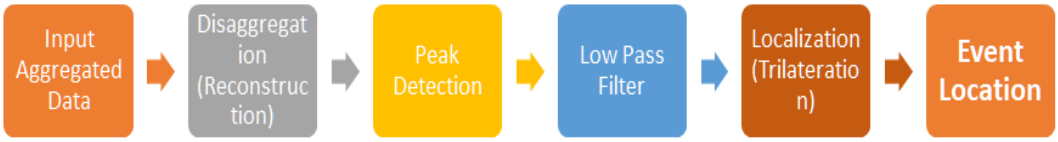
The underdeveloped data for localization would have missing and incomplete information that is necessary for localization. To deal with this issue we augment the data. We propose bottom-up and top-down techniques for augmenting the data, which we explain in the following sections.

3.1 Data Aggregation & Disaggregation

With the data sequences from microblogging, we want to formulate a framework to perform event localization without relying on the spatial or contextual attributes. Instead, we only rely on the **number** or count of microblog messages sent out by people (or social sensors [21]) within a geographic region and time window. In fact, the only known information is an aggregated count of the messages within a circle (also called *layer*) of a specific radius r_i around a reference coordinate (say $\mathcal{C}_i = (x_i, y_i)$ in 2D, which is NOT the event's actual location) within a time interval. To obtain more fine-grained location of an event, the count of messages in rings of a smaller radius would have to be obtained.



(a) Illustration of a sequence of data and peaks



(b) Steps in SPARE towards localization of event

Figure 3: SPARE methodology

Using the microblogging service of Twitter as a data source, we propose a novel approach called *SPARE (SPAtial REconstruction)* that can (a) reconstruct fine-grained counts of messages (called *tweets* In Twitter) by disaggregation of aggregated coarse-grained counts over the spatial (radial) dimension around $\mathcal{C}_i$, (b) detect event related patterns or spots in the fine-grained counts of tweets and (c) combine the spots from multiple reference points ($\mathcal{C}_i$) to localize the event. The complete SPARE methodology is summarized in Figure 3. The count of tweets is disaggregated using an "assumed" basis function. After disaggregation, we further refine the spot in the sequence so that we can identify at what distance from the reference point the event may have occurred. The event pattern or spot in fine-grained count of tweets are identified through peaks (increased but noisy counts). To eliminate spurious peaks or background noise, we use low pass filtering explained in Section 3.2.1. The spots

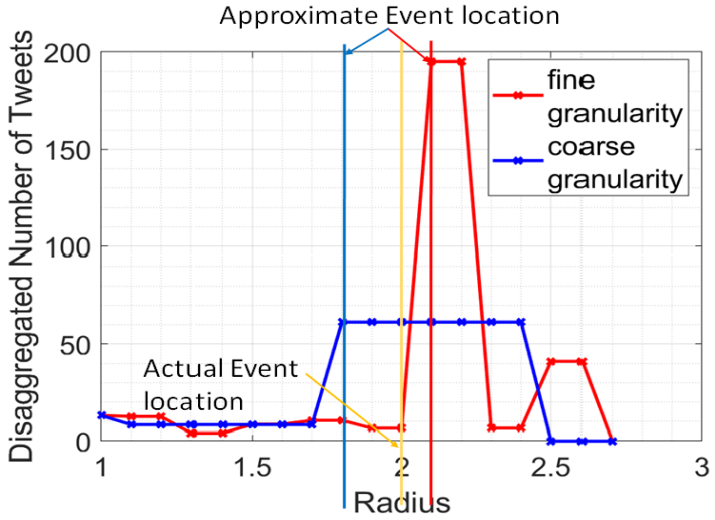
are still not precise and only yield noisy distances of the event from a reference point $\mathcal{C}_i$. Finally, using the noisy distances obtained at each $\mathcal{C}_i$, we use a localization technique (like trilateration) to localize an event.

Figure 4 illustrates an overview of how we use our methods to localize an event. Figure 4(a) shows the disaggregated counts of tweets at different radii from a single reference point $\mathcal{C}$ for the FIFA 2018 World Cup event (which is a known event providing a baseline). In the figure, the position of the peaks corresponds to the *presumed* location of the event. The counts are disaggregated at two different levels of granularity. The coarse-granularity is when only the number of tweets at every fifth layer (or radius from $\mathcal{C}$) is known which is typically how the scraped real-world data from a micro-blogging service like Twitter is available and fine-granularity is when the tweet count at every layer is known. Using the coarse-grained number of tweets the event's distance from the reference point is estimated to be between 1.8 and 2.3 miles. With finer granularity, the event's distance is estimated to be between 2.1 and 2.2 miles from the reference point.

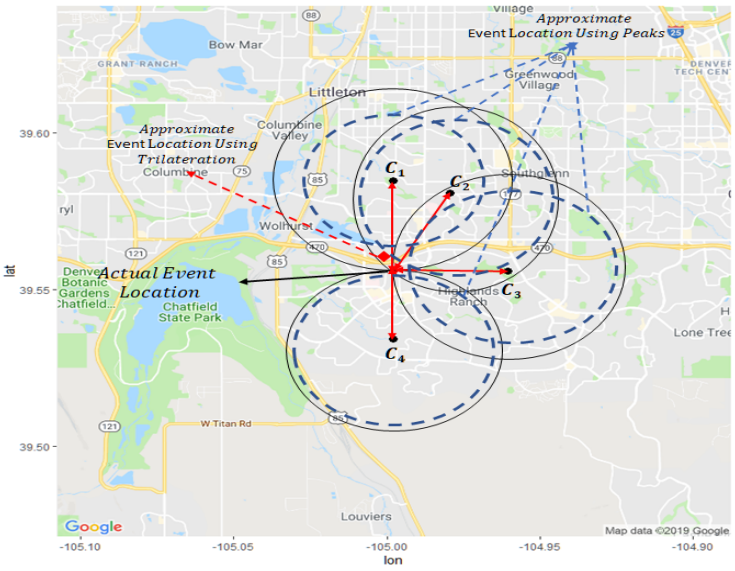
The position of the peak will identify an event's location at a (noisy) distance of r_i from point $\mathcal{C}_i$. If there are multiple reference points $\mathcal{C}_k, \mathcal{C}_j$, etc. each identifying an event location at a distance of r_j, r_k, etc. using the position of the peaks they observe in their scraped data, then the event location from each point can be combined to obtain a single geographic coordinate for an event by using techniques like *triangulation or trilateration* [13]. Figure 4(b) illustrates the trilateration methodology which we use in this paper. The figure shows 4 reference points ($\mathcal{C}_i, i \in \{1, 2, 3, 4\}$) at known coordinates. After disaggregation, each reference point identifies the spot of an event as a peak at a specific radius. We can then use trilateration to obtain an approximate location for the event. *Our results shows that the event location obtained from trilateration is quite close to the actual event location.*

In the above example, we used a *known* event like the FIFA world cup as a proxy to illustrate how SPARE works. In the case of unexpected events, we do not know the ground truth. We apply our approach to both augmented (generated) and actual data collected from Twitter. The augmented data is generated using Agent-Based Model (ABM) called *TBAM (Tweeting Behavior Agent Model)* that can be used to test the applicability of SPARE.

In summary, our contributions in this section are as follows:



(a) Disaggregated Tweets related to FIFA event



(b) STEM data collection

Figure 4: Data Collection Methodology

Formulation and Algorithm: We designed an innovative approach for event localization called *SPARE (SPAtial REconstruction)* Our approach discerns unknown event patterns from a spatial distribution of tweets for event localization and does not completely rely on spatial features and the content of tweets.

Applicability: We consider a novel collection of aggregate data with tweet counts over the spatial dimension (in a time window) instead of the temporal dimension. We then apply SPARE to this data to localize events.

Accuracy: Based on our experiments, events are localized very close to the actual event coordinates and no more than a mile of their occurrence.

Generality: We believe our proposed method may be applied to other domains/signals that have aggregate count information, such as waste water testing for diseases [58].

3.1.1 Temporal aggregation

We begin by describing temporal aggregation which forms the basis for spatial aggregation. The linear equations for temporal aggregation can be generalized as Equation 3.1 where O_t is the temporal observation matrix, N_t is the vector of the smallest granularity time units and Y_t is the vector of aggregated values.

$$O_t \times N_t = Y_t \tag{3.1}$$

Temporal aggregation has been studied in prior work described in [60] and [38]. Consider Figure 5(a), let $n_1, ...n_7$ be the values (e.g. number of tweets) observed at time $t_1, ...t_7$. For example, each value in $t_1, ...t_7$ is an observed value for a single day, then y_1 and y_2 are two aggregated values spanning 4 days each. Since, 4-days of observed data are aggregated together, then the report duration is 4 and there is one overlapping value in the two aggregated values, which is a shift. Report duration and shift are the two parameters that determine the aggregated values. Report duration is the number of fine-grained values aggregated together to obtain a coarser grained value. Shift denotes the number of overlapping values between corresponding aggregated values.

$n_1 + n_2 + n_3 + n_4 = y_1$

$n_4 + n_5 + n_6 + n_7 = y_2$

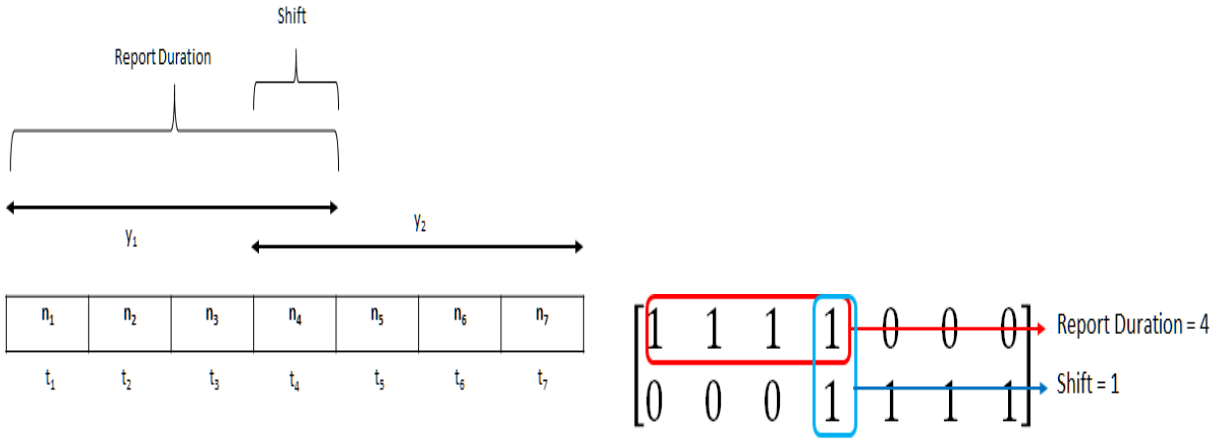
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ n_6 \\ n_7 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{3.2}$$

Temporal aggregation and the example from Figure 5(a) is represented as a system of linear equations - *characteristics linear system* [38]. The combination of linear equations for the aggregated values y_1 and y_2 are shown in Equations 3.1.1.

In the linear equations, the basis matrix that aggregates the values $n_1, ...n_7$ to obtain y_1, y_2 is called the temporal observation matrix denoted O_t. The temporal observation matrix is shown in Equation 3.3.

$$O_t = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.3}$$

The report duration and shift can also be determined from the observation matrix (Figure 5(b)).



(a) Representation of temporal aggregation

(b) Report duration and shift on temporal observation matrix

Figure 5: Temporal Observation Matrix

3.1.2 Spatial aggregation

The linear equation (in matrix form) typically used to represent aggregation [38] is shown in Equation (3.4) where O_s is called the spatial observation matrix, X is the vector of the smallest granularity spatial units (layers) and Y is the vector of the aggregated values (counts of tweets in our case).

$$O_s \times X = Y \tag{3.4}$$

In the problem we consider, Y is what we know, X is what we want, and O_s is a set of basis vectors we use to disaggregate Y. We explain this below.

Using Figure 6 as an example, we will explain how the number of tweets are aggregated along the spatial domain as aggregated counts lying within a circle or layer at a fixed distance (radius) from a point of reference $\mathcal{C}_i$. As the distance (radius) from $\mathcal{C}_i$ changes there would be a change in the number of tweets. In other words, the center of the layer $\mathcal{C}_i$ can also be considered as the reference point that measures the *total number of messages sent out by all the social sensors within a certain radius*. When we scrape data from Twitter, this is how the tweet counts are reported (y_1 tweets in a circle of radius r_1 around $\mathcal{C}_1$, y_2 tweets in a

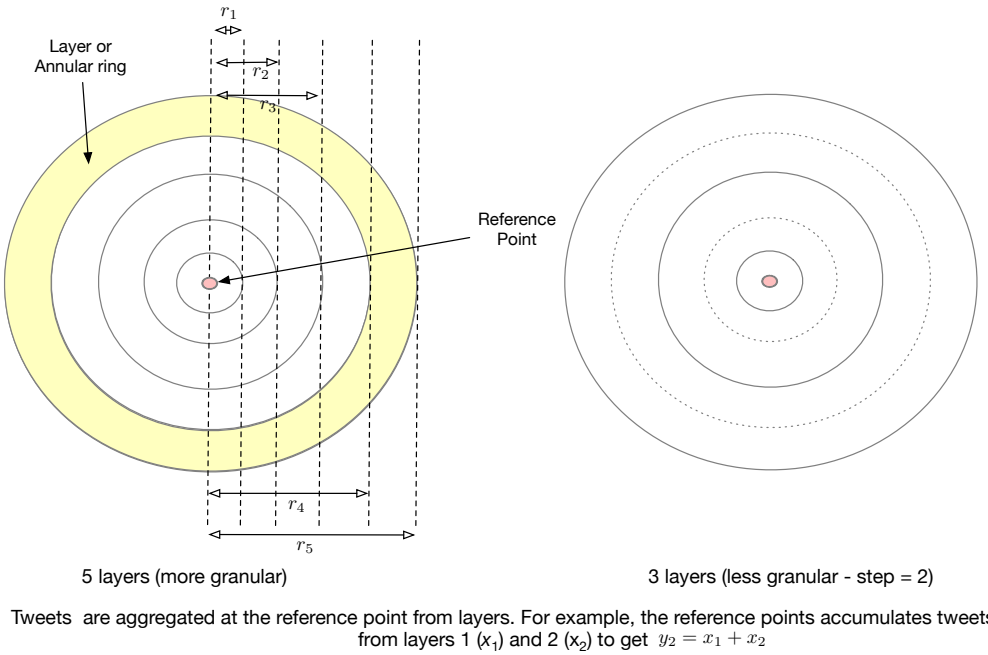


Figure 6: Representation of Spatial Aggregation

circle of radius r_2 around $\mathcal{C}_2$). In other words, we only have aggregated counts y_i. In Figure 6, there are five different layers of radii $r_1, ...r_5$ around reference point $\mathcal{C}_i$. In the annular ring between each pair of radii, the number of tweets are $x_1, ...x_5$ respectively which are the fine-grained values. Then the aggregated number of tweets are represented by $y_1, ...y_5$ - they are aggregated in circles, rather than the annular rings. Since we consider aggregation as the summation of all the number of tweets in the annular rings of smaller radii, aggregation can be represented as a set of linear equations. Consequently, by inserting the values in Equation (3.4), the aggregation for Figure 6 can be written as Equation (3.5).

$$
\begin{aligned}
y_1 &= x_1 \\
y_2 &= x_1 + x_2 \\
y_3 &= x_1 + x_2 + x_3 \\
y_4 &= x_1 + x_2 + x_3 + x_4 \\
y_5 &= x_1 + x_2 + x_3 + x_4 + x_5
\end{aligned}
\qquad
\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} \tag{3.5}
$$

From Equation 3.5, the observation matrix is:

$$
O_s = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}
$$

To represent the data with different granularity, we define the parameter *"step"*. The step size defines how many layers were aggregated and $1 \geq step \leq |X|$. A higher step size indicates coarse-granularity of data. Figure 6 shows that when the step = 2, then only aggregated values in 3 layers are considered. The corresponding characteristic linear equations for spatial aggregation with step = 2 and the corresponding observation matrix are shown below:

$$
\begin{aligned}
y_1 &= x_1 \\
y_3 &= x_1 + x_2 + x_3 \\
y_5 &= x_1 + x_2 + x_3 + x_4 + x_5
\end{aligned}
\qquad
\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_3 \\ y_5 \end{bmatrix} \tag{3.6}
$$

$$
O_s = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}
$$

Our objective is to find (as accurately as possible) the disaggregated values (X values) given some observation matrix and aggregated data. In the next section, we describe the disaggregation process and how the step affects the accuracy of disaggregation.

3.1.3 Information reconstruction via disaggregation

In the previous section how the aggregation of number of tweets occurs in space was described. The raw data that is typically available from Twitter are aggregated, i.e., X in Equation 3.4 is unknown and would have to be reconstructed through disaggregation, as shown below in Equation 3.7:

$$O_s^{-1} \times Y = \hat{X} \tag{3.7}$$

where O_s^{-1} is the inverse of the observation matrix, Y is the vector of the aggregated values, $\hat{X}$ is the vector of the approximation of the fine-grained reconstructed values X.

It should be noted that the observation matrix may be non-square and finding the inverse of a non-square matrix is not straight-forward. To find the inverse of the observation matrix we use the pseudo-inverse of the matrix using the *Moore-Penrose pseudo-inverse* [43]. The Moore-Penrose pseudo-inverse provides a Least Square Solution (LSQ) [22] to the system of linear equations. LSQ finds the minimum-norm solution to a linear system by minimizing the squared distance between the actual values and the product of the reconstructed values and the observation matrix.

$$min||Y - O_s\hat{X}||_2^2 \tag{3.8}$$

where Y is the observed aggregated value, O_s is the spatial observation matrix and $\hat{X}$ are the reconstructed approximate values.

Using Equation 3.5 and 3.6 as an example, disaggregation would be process of obtaining the fine-grained number of tweets, $X = x_1, ...x_5$ when only the aggregated number of tweets $Y = y_1, ...y_5$ and the observation matrix O_s are known. In this case we would obtain $\hat{X} = \hat{x_1}, ...\hat{x_5}$ which would be the approximate values. When step = 1, then $\hat{X}$ approximate values that would likely be equal to X. When step > 1, then $|X| \geq |Y|$, i.e. the number of unknowns which are in X become greater than the known values in Y. Such a characteristic system is called under-determined and has infinite number of solutions. Equation 3.6 illustrates

an under-determined system when step = 2. Using only y_1, y_2, y_3 we aim to obtain $\hat{x_1}, ...\hat{x_5}$ which can result in greater errors of approximation. In this case, only aggregated values at layers of radius r_1, r_3, r_5 are given and our disaggregation process estimates the number of tweets for not only r_1, r_3, r_5 but also r_2, r_4 which are added layers between r_1 and r_3 and r_3 and r_5 respectively. In this way we are adding finer-granularity to the coarse-grained number of tweets and the accuracy of reconstruction is reduced. Theoretically, if we had only a single aggregated value y_i at radius r_i, we could use that value to estimate infinite fine-grained values, but the reconstruction accuracy would greatly be reduced as more inner layers are added. The example below adds 5 smaller layers to y_i by disaggregation:

$$O_s^{-1} \times [y_i] = \begin{bmatrix} \hat{x_1} \\ \hat{x_2} \\ \hat{x_3} \\ \hat{x_4} \\ \hat{x_5} \end{bmatrix} \tag{3.9}$$

where $O_s = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

Reduced reconstruction accuracy with increasing step size is also observed in Figure 4 where we see disaggregated values, using LSQ, when step = 2 and step = 7. Similarly, Figure 7 shows disaggregation using LSQ with step = 3 using the *ground truth data* (described later in Section 5.2.2) where the deviation from the actual values can be seen. We show more experiments with changing step size and reconstruction accuracy later in Section 5.2.2.

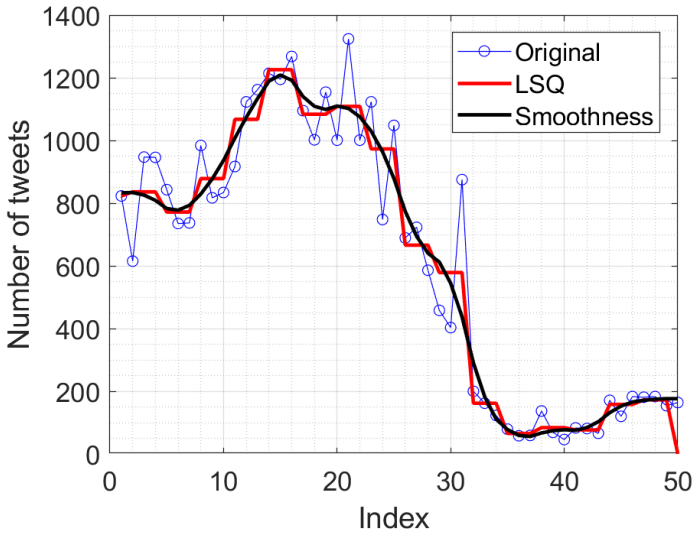


Figure 7: Comparison of original and disaggregated data using LSQ and smoothness

In order to improve the accuracy we do "regularization" by adding a constraint matrix to LSQ to improve the reconstruction ([38], [72]). Regularization aims to better estimate the values in smaller granularity. The disaggregation can be visualized as an optimization problem, where the aim is to minimize sum of the deviation from actual values taking into account the constraint matrix. The constraint matrix considers domain knowledge and the past and present trends to more accurately reconstruct the data. The mathematical representation of the problem is as follows:

$$min(F(\hat{X}) + C(\hat{X})) \tag{3.10}$$

where $F(\hat{X}) = ||Y - O_s\hat{X}||_2^2$ is the deviation from actual values and $C(\hat{X}) = \sum_{t=1}^{N}(x_t - x_{t+1})^2 = ||S\hat{X}||_2^2$ is the smoothness matrix. In $C(\hat{X})$, x_t and x_{t+1} are part of a sequence $X = x_1, ..., x_T$ with T time-ticks or layers. The smoothness matrix S is a $\mathbb{R}^{(T-1)\times T}$ matrix with a 1 in the t^{th} row and -1 in t^{th} and $(t+1)^{th}$ column and zero everywhere else. The smoothness constraint [38] penalizes large variations between successive time or spatial units. Note that there may be other definitions of constraint matrix like periodicity [38] or distance-based constraints. The difference in reconstruction for the ground truth data when adding the smoothness constraint can be clearly seen in Figure 7. The smoothness reconstructed

values appear to match the actual the actual values more closely than reconstruction with LSQ only when step > 1. We use smoothness constraint as an example in this dissertation to demonstrate the possibility of using constraints. We do not explore the "best" constraint matrix in this dissertation.

3.2 Explicit Event Patterns

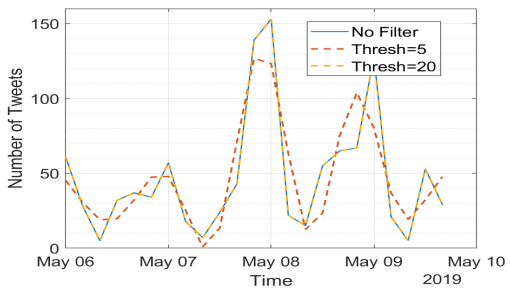
In this section we propose our definition of both explicit and latent event patterns. We also propose filtering techniques to refine event patterns and assess the peaks.

3.2.1 Explicit Event Patterns

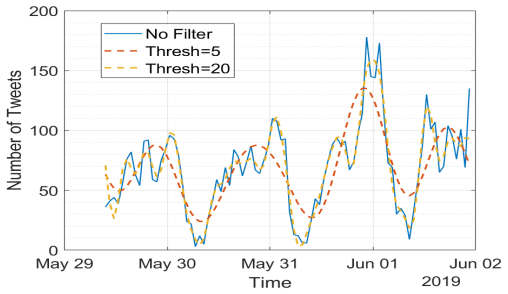
Figure 8 shows a time series of number of tweets collected from Twitter for two considered events (VIRG and STEM). The event data is explained later in Chapter 4. We can observe that the occurrence of an event is followed by a peak as there is a large change in the number of tweets along the temporal dimension around the event. The peaks are an explicit patterns of an event found in the time series. In SPARE and for our dissertation we define a peak as a local maxima. We also consider a point as a peak if it is greater than δ points before it and equal to δ points after it. For simplicity we consider only adjacent points, i.e., $\delta = 1$.

With a sequence of numbers of tweets varying with time or space, the position of the peaks can determine the time or location or spot of an event. There are alternative methods of detecting events – for example, detecting event patterns through machine learning. If we disaggregate the input sequence and obtain finer-grained data, a more precise position of the peaks can be determined which can be used to approximate the location and time of an event.

However, not all peaks are indicative of an event. For example, some peaks could be part of routine Twitter behavior and are part of a standard recurring Twitter pattern. Obtaining peaks from aggregated data can also lead to errors in event detection as some peaks may be removed or appear at a different position. For example, in Figure 7 it can be seen that



(a) Effect of Filtering on STEM event data



(b) Effect of Filtering on VIRG event data

Figure 8: Effect of Filter Threshold on Data Collected from Twitter

reconstructing data using smoothness clearly effects the magnitude and position of the peaks which can influence the accuracy of event detection as peaks corresponding to an event may be lost. We modify how we define peaks relating to events by considering what we call as *significant peaks*.

3.2.1.1 Refining explicit patterns - low pass filters To extract "significant peaks" we use a novel approach based on passing the disaggregated data through a filter. We could use advanced filtering techniques or even look into some form of pattern recognition that compares current data with historical data. For this thesis we propose a simple low pass filter. The low pass filter removes any small variations and will keep the most "significant peaks". The filter threshold can vary the output sequence and the number of significant peaks. Figure 9 shows how the filter affects the ground truth data for different values of filter threshold. There is a trade-off in setting the filter threshold. At low filter thresholds all the peaks are removed and the data only appears as a straight line and does not coincide with the actual data. As the threshold increases the number of peaks detected increase which can result in peaks not relating to events being detected which we observe from our results. In later section, we look at different low pass filters and how they affect event patterns for both real data and data generated using TBAM.

As Figure 8 shows, multiple significant peaks are present after different filter thresholds

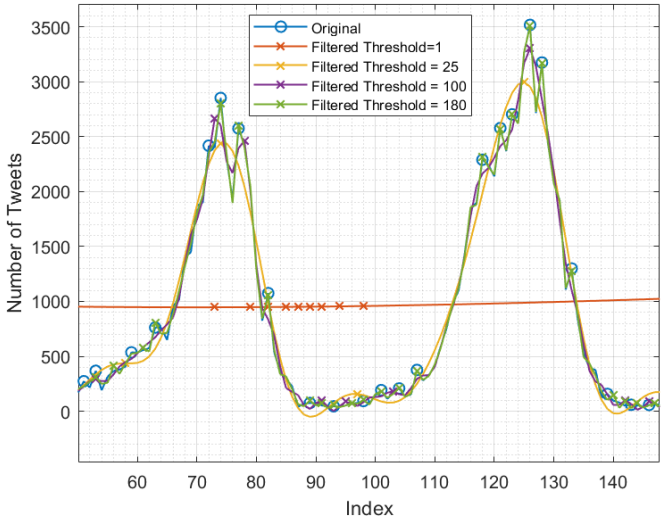


Figure 9: Effect of filter threshold on data

are applied. Ideally all significant peaks should be caused by events. However, that is not always the case. Low pass filtering may not be ideal to identify peaks caused by events. The significant peaks that are due to events are called *event-related peaks*. Event-related peaks can be used to identify the time (and or location) of an event. Differentiating event-related peaks from significant peaks can be complicated and we propose using a similarity based analysis to identify event-related peaks.

3.2.1.2 Refining explicit patterns - semantic decay filters (SDF) The similarity based filter or *semantic decay filter (SDF)* [5] is based on the hypothesis that event-related peaks would have a lot of similar texts in the tweets as users will make references to an event in their tweets. For the similarity measure we use the function F [35] that maps words into the real vector space $\mathbb{R}^n$ in such a way that the distance between two similar words (i.e., non-standard spellings of the same word, or words used in the same context) will be the shortest distance between the corresponding mapping in the real vector space. Consequently, an event peak can be identified as follows: the aggregated decay in similarity over all the tweets in an event-related peak will be slower than decay in similarity in a non-event-related peak.

Let N_t^k be the aggregated number of similar tweets is a set of tweets $T_t = \{T_1, T_2, ..., T_n\}$ posted during a time window t for a threshold a_k. N_t^k is given by the following equation:

$$N_t^k = \sum_{i=1}^{n} \sum_{j=1}^{n} 1_{Similarity(T_i,T_j)>a_k}. \tag{3.11}$$

Where:

$$1_{Similarity(T_i,T_j)>a_k} = \begin{cases} 1 & \text{if } Similarity(T_i, T_j) > a_k \\ 0 & \text{otherwise} \end{cases}$$

The decay λ_t^k is then calculated using the following equation:

$$\lambda_t^k = \frac{1}{a_k - a_0} \log \frac{N_t^k}{N_t^0}. \tag{3.12}$$

Equation 3.12 is an exponential decay function used in many areas, most notably in physics to calculate the radioactive decay [24] by just replacing a_k with the time and N_t^k with the number of atoms. We used that analogy as source of inspiration for Equation 3.12.

A special case can occur in which, during the time window t, the number of posted tweets is low. Nevertheless, those tweets are similar or identical. In this case, will observe a low decay corresponding to depression in the number of tweets. Thus, using the decay alone as an indicator is not sufficient. Our indicator is a linear combination of the normalized decay and the normalized number of tweets $|T_t|$ in a specific time window:

$$SDF_t = \alpha Norm(|T_t|) + \beta Norm(\lambda_t^k) \tag{3.13}$$

where $Norm$ is a normalization function [1], α, and β are real numbers. α and β are added to adjust the importance of normalized number of tweets and the normalized decay. To generate the SDF, the parameters $\alpha = 1$ and $\beta = -1$.

[1] normalization function normalizes by dividing with the total number of tweets or λ so that the values lie between 0 & 1

3.2.2 Latent Event Patterns

To find latent patterns related to an event we use Long short-term memory (LSTM). LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning [29]. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process both images and sequences of data (such as time series data or speech or video). LSTM can classify and predict time series data and has been used for handwriting recognition, speech recognition, estimating future values in a time series data and anomaly detection in network traffic or IDS (intrusion detection systems).

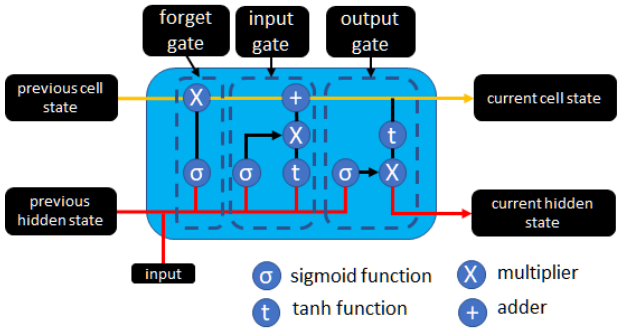


Figure 10: Long Short-Term Memory (LSTM) Architecture

The LSTM architecture is shown in Figure 10. In the figure "input" is the count of tweets at a specific time unit. An LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cells store the values and transfers information all the way down the sequence chain over an arbitrary time interval. The three gates are the neural networks that decide which information is allowed to flow in and out of the cell. The gates use `sigmoid` function and `tanh` function to make the decision about the information flow. The sigmoid function is defined as $\sigma(x) = \frac{1}{1+e^{-x}}$ and maps x between 0 and 1. The `tanh` function is defined as $tanh(x) = \frac{2}{1+e^{-2x}} - 1$ and maps x between -1 and 1 for network regulation.

The forget gate decides which information should be kept or discarded. The forget gate combines the values from previous hidden state and the value of the input sequence at a specific time and uses the sigmoid function to map these values between 0 and 1. The

sigmoid function output is combined with the previous cell state. If the sigmoid function is close to 0, then the previous cell state is forgotten and if the sigmoid function output is close to 1, then the previous cell state is kept.

The input gate controls which values are passed through and if any new information is added to the data from the previous state. First the product of the output of the sigmoid function and the tanh function is taken. The tanh and sigmoid function are computed on the combination of the values from previous hidden state and the value of the input sequence at a specific time. Next the product is added to the previous cell state and becomes the current cell state. The output of the the product is between 0 and 1. A 0 means the value is not important, and 1 means it is important. In this way information is added to the previous cell state that is important and not redundant and passed onto the next cell.

Finally, the output gate computes the next hidden state. First a tanh function is applied on the current cell state, which was obtained from the input gate. The tanh function is multiplied with the sigmoid function applied to the combination of the values from previous hidden state and the value of the input sequence at a specific time. The output of the product becomes the current hidden state which is passed onto the next cell.

Bidirectional LSTM [53] is a special type of LSTM. It can learn long-term dependencies between time steps of time series or sequence data and is able to learn both forward and backward information about the sequence at each step. We use this special type of LSTM for learning latent event patterns. Bidirectional LSTM takes in two sets of inputs: values from past to the future and from future to the past. For example, if a large number of tweets are identified in the future, then the reverse direction would allow such a trend to be identified as an event as people tend to send out a large number of tweets after an event.

3.3 Top-down data generation

We propose using agent-based modeling (ABM) as the top-down data generation model. ABM augments the data by generating synthetic data using parameters which may be learned from real Twitter data. ABM has been used in many different fields for analysis and un-

derstanding of the real world like biology, chemistry, cyber-security, social and economic modeling, etc. [2]. Agent-based models (ABM) [69] have entities or *agents*. An agent can be an individual or an object that has specific properties and actions. The agents move around in a two dimensional grid called the *world*. The interactions between the agents can be quite complex but can be defined according to a set of rules. An agent can be autonomous, flexible, adaptable and self-learning [41]. There may be other models that can simulate scenarios pertaining to human social behavior and social media information dispersion (like system dynamics), however, ABM is able to better represent complex and heterogeneous interactions ([69], [41]) which makes it suitable for creating our model.

To generate synthetic data we propose Tweeting Behavior Agent Model (TBAM) that uses Agent-Based Model (ABM) to simulate how users tweet and how their behavior changes when an event occurs. We use Netlogo to create the agent-based model. In this section we provide a brief overview of the model and the different parameters used to generate synthetic data. We call this as a top-down approach because we pick the parameters of the ABM and try to match the data sequences observed with the microblogging data.

3.3.1 TBAM design

The idea behind TBAM is to simulate user behavior when an event occurs, more specifically to investigate the change in number of tweets as time and distance from event changes. For our model we consider "local events" [46], i.e., events restricted to a certain region. Our model simulates microblogging behavior of people within a city or a few small neighborhoods and helps us examine how people's microblogging behavior changes when they have close spatial and temporal proximity to an event.

We use *Netlogo* [68] to create our agent based model. In Netlogo there are four types of agents: turtles, patches, links and observer. The turtles are agents that move around in the *world*. The world is sub-divided into smaller squares called *patches* and each patch has a unique coordinate. Links are agents that connect two turtles. The observer observes the agents and their interactions. In Netlogo models, time passes in discrete steps called *ticks*.

Figure 11 shows a snapshot of the synthetic world at a particular tick. The parameters in

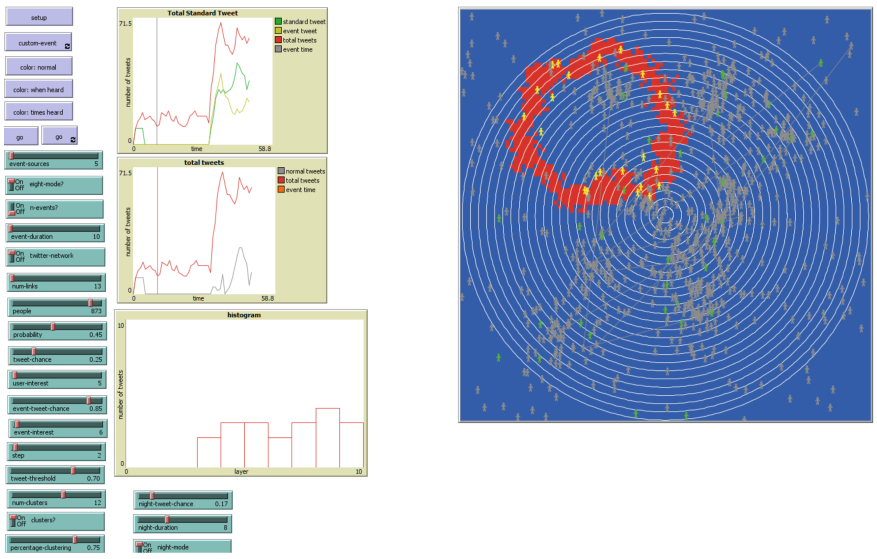


Figure 11: Representation of the ABM world

the figure are summarized in Table 2 and Table 3. In our model, the turtles are the Twitter users (people) who send out the tweets. A tweet can be a non-event related tweet (which is a standard or routine tweet indicated by green colored users), an event related tweet (indicated by users colored yellow) or tweets sent out during low Twitter activity (indication by users colored black). The patches represent the locations over which Twitter users lie and where an event can occur. Initially all patches are colored blue. Once an event occurs, the patches change color to red as the patches are influenced by the event. In a real world setting a patch could represent a geographical coordinate. A tick is a unit of time over which the total number of tweets are measured. Ticks could be in hours, minutes or seconds depending on the time granularity that is required.

3.3.2 TBAM explanation

In order to generate data that accurately reflects real world settings we define different parameters. The parameters are summarized in Table 2 and Table 3. In this section we provide an overview of these parameters and explain how our model simulates microblogging

behavior and event generation. The model is made of two phases. In the first phase, also called the *setup* phase, the synthetic world settings are created in which the users will tweet. In the second phase, also called the *simulation* phase the users tweet and once an event occurs, their microblogging behavior changes.

The setup phase begins by generating N users. The users are randomly distributed throughout the world. Some of the users are clustered together. The number of clusters are defined by the parameter `num-clusters`. The parameter `cluster?` determines if the users are clustered or not and `percentage-clustering` determines what percentage of users are clustered together randomly in each of the clusters. The setup phase also generates concentric circles around the central coordinate, i.e., $(0, 0)$. The central coordinate is the location of the sensor that counts the number of tweets. Each consequent circle increases its radius by the parameter `step`. These circles aim to provide a visual analysis of how tweets change with changing distance.

To simulate a Twitter network, some users are linked together with bi-directional links. The links are generated using *Erdos-Renyi* model which has been used in previous literature to study social networks [15]. In the Erdos-Renyi model, each link between a user has a fixed probability of being present and being absent independent of the links in a network. The parameter `probability` can be varied to change the probability and create networks with a lot of links or with very few links between users. There is also an option for generating a network with random links between users. The `num-links` is a parameter specific to random networks which randomly creates `num-links` number of links between different users. The `twitter-network` can be set to true or false to choose between Erdos-Renyi or random model to generate the network.

In the simulation phase at each tick the total number of tweets sent out are counted. The total number of tweets are the sum of standard tweets, event related tweets and tweets sent out during low Twitter activity. It should be noted that we are able to separate these in TBAM but it may not be possible to do so with scrapped Twitter data. At each tick, a random number z_i is generated for each user. The random number is used to create the random conditions where users may not choose to tweet at a specific time. Since it is hard to determine these random conditions, we assume that z_i is normally distributed random

Table 2: Fixed parameters for all TBAM data generation simulations

Parameter	**Description**	**Value**
`n-events?`	binary: chooses between one event or n-events	FALSE
`event-sources`	sets number of events. Only valid if n-events is true	1
`eight-mode?`	binary: chooses between spreading event to both diagonal and adjacent patches or only to adjacent patches	true
`twitter-network`	binary: chooses between Erdos-Renyii and random network	Erdos-Renyii
`num-links`	number of links in the random network	-NA-
`people`	number of people microblogging	1000
`probability`	probability of a link being created between two people in the Erdos-Renyii Network	0.45
`step`	distance between each layer	7
`num-clusters`	number of clusters of people	9
`cluster?`	binary: choose to cluster people (1) or distribute people uniformly (0)	1
`percentage-clustering`	percentage of people in clusters	0.75
`tweet-threshold`	used to generate random number that a user will tweet	0.7
`user-interest`	duration over which people remain interested about a tweet	5
`event-interest`	duration over which people remain interested about tweets related to an event	5
`night-mode`	to consider periodicity in tweets and separate users microblogging at day and night	True

Table 3: Variable parameters for different TBAM data generation simulations

Parameter	**Description**	**VIRG**	**STEM**	**GAR**
`tweet-chance`	probability of a person sending out a tweet	0.33	0.29	0.22
`event-duration`	length of time that event remains active	31	48	8
`event-tweet-chance`	probability of a person sending out a tweet about an event	0.49	0.55	0.67
`night-tweet-chance`	probability of sending out tweets during low Twitter activity	0.17	0.16	0.12
`night-duration`	the duration of low Twitter activity	8	8	8
`ndist`	scaling factor effecting decay for q_i	0.07	0.12	2

number of mean `tweet-threshold` and variance 0.2. The scrapped twitter data also follows a rough normal distribution which is why we chose the random conditions to be normally distributed. Before an event occurs a user will only send out a routine tweet. A user will only tweet if $z_i <$ `tweet-chance` where z_i is a random number generated for user i and `tweet-chance` is probability of sending out a standard tweet (from Table 3).

At a specific time (tick) and location (patch) the event occurs and with each tick spreads across the world. The rumor spreading model has been used as a basis to simulate spreading of an event influence (or information) [67]. There may be other models that can be used to simulate spreading of an event, like the susceptible-infected-refractory (SIR) model [71]. Similar to the rumor spreading model, immediately after an event occurs, the event influence starts spreading to all the neighboring patches (shown by the red colored patches in Figure 11). However, the rate at which the event influence spreads to its adjacent neighbors may not be uniform and may vary with time. Initially, as soon as the event occurs, the event influence immediately spreads to all patches within a fixed radius. The influence then spreads to adjacent neighboring patches with decreasing rate as more time elapses. This assumption is based on the observations made from Twitter data collected that shows a sharp rise in the immediately after an event. One parameter that effects the spreading of event is the `eightmode?`. Setting the `eightmode?` parameter to true causes the event to spread to its

diagonals and its adjacent neighbours but setting the `eightmode?` parameter to false causes the event to only spread to its adjacent neighbours. When the `eightmode?` is true, then the event spreads outwards more quickly.

Once an event occurs, a user can send out either an event related tweet or a routine tweet. A user will choose to send out a tweet about an event if $z_i < \frac{q_i}{(q_i + \texttt{tweet-chance})}$ where z_i is a random number generated for user i as described previously, q_i is probability a user i will tweet about an event and `tweet-chance` is from Table 3. Once a user chooses to tweet about an event, then a user will only send out tweet about an event if they are on a patch where an event has spread to and $z_i < q_i$ where z_i is a random number generated for user i and q_i is probability a user i will tweet about an event. There are multiple methods of determining q_i. q_i could be fixed or vary with time and distance from event. For the model we create a hybrid approach. In the immediate vicinity of the event $q_i =$ `event-tweet-chance` where `event-tweet-chance` is one of the parameters from Table 3. But as the distance and time from event increases, then q_i decreases according to Equation 3.14.

$$q_i = event - tweet - chance * [(t - t_{event})^{-\texttt{ndist}/\alpha} * (d_{event})^{-\texttt{ndist}/\beta}] \tag{3.14}$$

t is current time tick measured after the event occurs, t_{event} is the time tick at which event occurred, d_{event} is the distance of the user from the event, `ndist`$/\alpha$ and `ndist`$/\beta$ is a scaling factor. A high `ndist` values means that `event-tweet-chance` decays less rapidly with changing time and distance respectively. For our model we keep α fixed at 1 and β fixed at 20. Since we are considering local events and users would generally be in close proximity to the event, therefore, the decay of `event-tweet-chance` with distance should reduce less rapidly than decay due to time. Hence, we choose a larger value of β than for α. It should be noted that there are many different functions that could be used to simulate the decay of probability of microblogging about an event. Previous literature ([49], [52]) have considered exponential distribution for tweets which can also be observed from the collected twitter data. Hence, we choose an exponential function to simulate how probability of microblogging about an event decays with time and distance. Figure 12 is a plot of the function showing how q_i changes as distance and time from event changes when $\alpha = 1$ and $\beta = 20$.

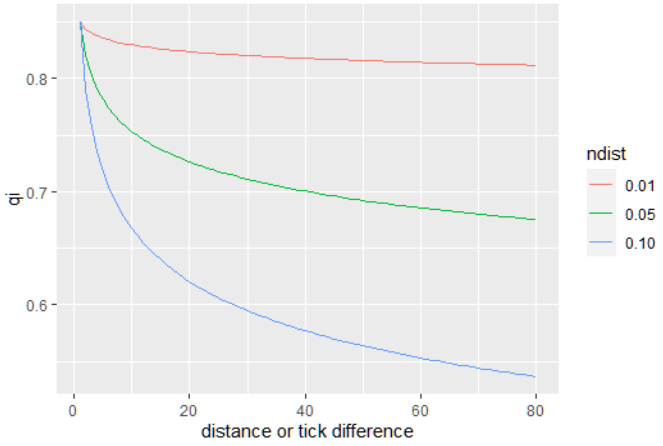


Figure 12: Changing q_i with changing distance or ticks (with $\alpha = 1$ and $\beta = 20$)

During the simulation phase, users can also send out retweets. A user will send out a standard retweet if $z_i <$ `tweet-chance` and there is a link with another user who has sent out a standard tweet. Consequently, a user will only send out an event related retweet, if $z_i < q_i$ and there is a link with another user who has sent out an event related tweet. We define the parameters `event-interest` and `user-interest` as the tick duration over which users will keep on talking about an event or a routine tweet. These parameters quantify the importance of standard or event related tweets and the higher these parameters are the larger will be the number of retweets sent out. For our simulations, we keep these values constant.

The event ends when `event-duration` ticks have elapsed. Once the event ends, then just like the rumor spreading model, the patches lose the influence of the the events. No new tweets relating to an event are generated and the event related tweets gradually decrease until they eventually stop. Higher `event-duration` value signifies that users will continue to generate new tweets about an event for longer periods of time.

There is also the option of choosing multiple events which is done by setting `n-events?` true. If there are multiple events then `event-sources` sets the number of event sources. For this model we use one event source. In short, `event-duration`, `event-tweet-chance` and

`event-interest` determine how significant an event is. If these values are set high then it indicates an event that has a high impact on users' lives and they will tweet and retweet more about the event and remain interested in the event for longer duration. These parameters can be changed to incorporate different types of events.

The data collected from Twitter reveals periods of time with very few tweets being sent out. To incorporate such behavior we introduce the parameter `night-mode` that enables or disables consideration of time when there is low Twitter activity. If `night-mode` is enabled, then there are two parameters that affect the low Twitter activity. One parameter `night-duration` effects how long the low activity period lasts. The other parameter `night-tweet-chance` is a measure of the probability of a user microblogging during the low Twitter time period. A user sends out tweets during this time only when $z_i <$ `night-tweet-chance`. Usually `night-tweet-chance` would be less than `tweet-chance` which in turn is usually less than `event-tweet-chance`.

3.4 Bottom-up data generation

Our bottom-up approach generates data by learning from the real world data directly without input of any parameters. In this thesis we use generative adversarial network (GAN) as a bottom-up approach to generate data. As we discussed in Section 2.2, GAN has been used to generate data to augment and enhance the real data. As we are also attempting to augment our data, using GAN is a logical option.

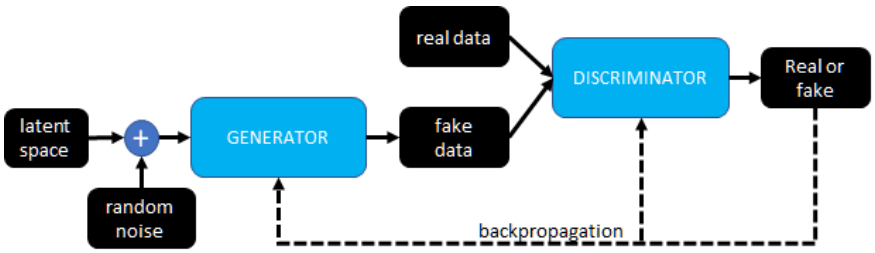


Figure 13: Generative Adversarial Network (GAN) Architecture

GAN is a deep learning framework invented in 2014 [23]. The GAN architecture can be seen in Figure 13. GAN generates new data based on the same statistics as the training set. GAN has been used to generate new photographs that can look very similar to the photographs on which GAN was trained on.

GAN consists of two agents: the generator and the discriminator. The generator and the discriminator work against each other (almost like game theory). The generative network generates realistic looking fake data with the objective to increase the error rate of the discriminator. The discriminator would attempt to catch the deceptive data. The generator trains on data that successfully fools the discriminator while the discriminator trains on the known training data. The generator is seeded with randomized input that is selected from a latent space of a specific distribution (like normal distribution). Back propagation allows the generator to produces better data and the discriminator becomes more proficient at catching fake data.

4.0 Data sets Used for Localization

In this chapter, we introduce the data on which we apply the enrichment strategy and use the data for localization. We perform the following tasks in this chapter:

- Introduce three different data sets
- Show how TBAM data can be generated using heuristics
- Define metrics that can be used to validate the TBAM generated data
- Show how the metrics are affected when different TBAM parameters are changed
- Compare TBAM and GAN and their affect on the metrics
- Apply filters on TBAM and real data and show how the patterns are affected in the two data sets

4.1 Introduction to the data sets

In this section we first describe the data that we use for analysis in the subsequent sections. We explored three different types of data sets to examine how different parameters effect the performance of disaggregation, reconstruction, and localization. The first data set is based an epidemiological series based on weekly trend of New York measles cases from 1928 to 1936 obtained from Tycho [63]. This was used previously ([38], [72]), and although not in the spatial domain, provides a baseline for the proposed approach. We examine step size, filter threshold and tolerance and this epidemiological data set seems most viable as it has dynamics similar to Twitter data [32].

The second type of data set uses raw tweets from different events that are represented in a way similar to Section 3.1.2. The 'TwitteR' package in R [20] allows tweet count to be collected with in a specific radius around a central coordinate. The data set with their details are summarized in Table 4. However, this data is **aggregated** and there is no ground truth to assess the disaggregated data.

Table 4: Summary of Real Data

Event Name	**Ref Name**	**Event Date**	**Event Location (latitude, longitude)**
FIFA World Cup Final	FIFA	03-15-2018	55.715989, 37.553758
London Bridge Attacks	LON	06-03-2017 10:16pm	51.508056, -0.085717
STEM School Shootings	STEM	05-07-2019 1:53pm	39.556, -104.9979
Virginia Beach Shootings	VIRG	05-31-2019 4:44pm	36.7509, -76.0575
Garlic Festival Shootings	GAR	07-28-2019 5:40pm	36.997778, -121.585278
El-Paso Shootings	ELP	09-03-2019 10:45am	31.7771, -106.3843
Santa Clarita Shootings	SANTA	11-14-2019 7:38am	34.4419, -118.5177

The third type of data sets are generated using different modeling methods TBAM and GAN. The implementation of data generated using GAN is described later in Section 5.2.

To generate data using TBAM, we use the parameter values described in Table 2 and Table 3. Table 2 shows the parameters that are kept fixed for all the simulations. Table 3 shows the parameters that are changed according to the Twitter data set they are meant to match. The parameters in Table 3 were estimated by inspection of the Twitter data. Table 5 summarizes how we estimated the different parameters from the Twitter data. The `event-duration` was estimated as the duration over which the number of tweets sent after the event were higher than tweets sent before the event. For example, in Figure 14(b) the number of tweets in the *Real* data return to the value before the event after 48 ticks, hence, the TBAM `event-duration` parameter was set to 48 to generate the data in that figure.

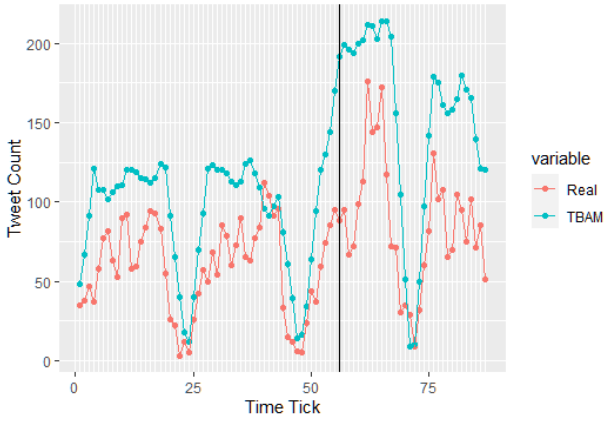
Similarly, from the real data we calculate the values for $tweets_{night}$, $tweets_{pre-event}$ and $tweets_{post-event}$. Then using these values we estimate the probabilities for the TBAM parameters of `tweet-chance`, `event-tweet-chance` and `night-tweet-chance` which are then used to generate TBAM data.

The heuristic analysis of the data from Twitter reveals how the different parameters vary for different areas and events. The difference in these parameters could be due to

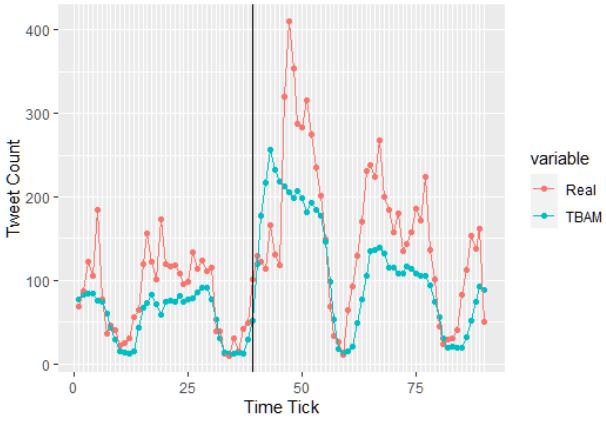
Table 5: Determining the probabilities from Twitter data for TBAM

Parameter	**Description**
$tweets_{night}$	mean number of tweets sent in low Twitter activity hours
$tweets_{pre-event}$	mean number of tweets sent before the event (excluding low Twitter activity tweets)
$tweets_{post-event}$	mean number of tweets sent after the event (excluding low Twitter activity hour tweets)
`event-duration`	duration over which number of tweets sent after event are remain higher than number of tweets sent before event
`tweet-chance`	$\frac{tweets_{pre-event}}{(tweets_{night})+tweets_{pre-event}+tweets_{post-event}}$
`event-tweet-chance`	$\frac{tweets_{post-event}}{(tweets_{night})+tweets_{pre-event}+tweets_{post-event}}$
`night-tweet-chance`	$\frac{tweets_{night}}{(tweets_{night})+tweets_{pre-event}+tweets_{post-event}}$

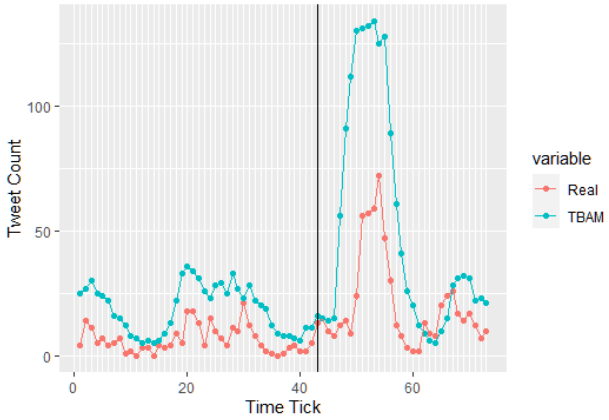
the difference in demographics, Twitter usage and density of the Twitter network. For all three data sets we considered a similar event. VIRG and STEM had roughly similar parameters but GAR has very different parameters. This is because GAR refers to two events combined as one. GAR event is different from the other two events as it started off as a different event which was a festival but ended up as a shooting event. As a result there were more Twitter users compared to usual days and the parameters `event-tweet-chance`, `event-duration` and `ndist` are significantly different than the other two events. The parameter of `event-duration` is significantly shorter due to Twitter users leaving the event location and hence, the scaling factor is high to account for the high outflow of Twitter users. Figures 14(a), 14(b) and 14(c) represents the plots of the data. *Real* indicates data obtained from Twitter and *TBAM* indicates data generated through TBAM. Each tick represents the number of tweets sent out in an hour. The occurrence of an event is indicated by the vertical line.



(a) VIRG Twitter Data vs TBAM generated data



(b) STEM Twitter Data vs TBAM generated data



(c) GAR Twitter Data vs TBAM generated data

Figure 14: Comparison of Twitter data with TBAM generated data

4.1.1 Performance Metrics for TBAM analysis

For comparing the peaks in the TBAM data with the real data we define a set of metrics. The metrics are summarized in Table 6. The accuracy, penalty and quality measure how well the peaks in the data from Twitter compare to the TBAM generated data.

The metrics are based on some essential parameters. The first parameter is *distance (`d`)*. The *distance (d)* is defined as the distance of peak in the TBAM data to the nearest peak in data collected from Twitter. Lower the distance better is the identification of the event signature.

The next parameter is *number of mismatches (`m`)*. This is defined as the difference in the number of peaks in the data collected from Twitter and the TBAM generated data. The parameters can be explained through the following example. Let the position of peaks in data collected from Twitter be $\texttt{A} = [3, 7, 9, 10, 13]$ and the position of peaks in TBAM generated data be $\texttt{B} = [3, 6, 9, 12]$. Then we define a matrix (Equation 4.1) `AB` which is the difference of vectors `A` and `B`. Then `d` would be the mean of the minimum distance values in each of the rows. In case of our example $\texttt{d} = mean(0, 0, 0, 1) = 0.25$. Consequently, $\texttt{m} = |A| - |B| = 6 - 5 = 1$.

$$AB = \begin{bmatrix} \mathbf{0} & -4 & -6 & -7 & -10 \\ 4 & \mathbf{0} & -2 & -3 & -6 \\ 6 & 2 & \mathbf{0} & -1 & -4 \\ 9 & 5 & 3 & 2 & \mathbf{1} \end{bmatrix} \tag{4.1}$$

The first metric is *accuracy (`acc`)* which is defined as: $\texttt{acc} = 1 - \frac{d}{t_s - 1}$. The second metric is *event detection penalty (`pen`)* which is defined as $\frac{m}{t_s}$. The third metric is *event detection quality (`qual`)* which is defined as $1 - \frac{(d+m)}{(2t_s - 1)}$. In the equations for the metrics t_s is the length of data collected from Twitter. Ideally `acc` and `qual` should be high and `pen` should be low.

In addition to the parameters above, we also use the *cross-correlation function (`ccf`)* [57]. The cross-correlation function between two time series x_t and y_t is given by:

$$\rho_{xy}(s,t) = \frac{\gamma_{xy}(s,t)}{\sqrt{\gamma_x(s,s)\gamma_y(t,t)}} \tag{4.2}$$

$$\gamma_{xy}(s,t) = cov(x_s, y_t) = E[(x_s - \mu_{xs})(y_s - \mu_{yt})] \tag{4.3}$$

μ_{xs} is the mean of time series x_s and μ_{yt} is the mean of time series y_t.

The cross-correlation measures the dependence between two points on different time series observed at different times. In other words, `ccf` measures the linear predictability of the series at time s, say x_s, using only the value y_t. In our TBAM data we are trying to see if the trends and patterns in the original data match. Hence, *ccf* will be a suitable metric and provides a measure of similarity between the two data sets.

4.1.2 Determining Optimum TBAM Parameters

In the previous section we generated our data using heuristics. Previous works have looked at using similarity [37] or a surrogate approach [75] to optimize and calibrate agent based model and its parameters. In this section, we will look at how the different parameters effect the metrics `maximum ccf, ccf at` $lag = 0$ and `root mean square error (RMSE)`. By measuring these metrics, we can determine the optimum set of parameters for generating the data using TBAM. We focus on the STEM and VIRG data sets as the basis for the TBAM parameters and to generate the GAN data set. We also compare our data generation technique with generative adversarial network (GAN) which is a popular data augmentation technique. There are four different simulations that look at the different parameters. In each simulation some parameters were kept fixed while others were changed. The simulations are summarized in Table 7.

Figure 15(a) and Figure 15(b) show the result of our simulation. Ideally, `rmse` should be low and `ccf at` $lag = 0$ should be near 1. Figure 15(a) shows that `ccf at` $lag = 0$ and `maximum ccf` are the same which is the ideal case. This indicates that the data generated using TBAM and the data collected from Twitter have patterns that are very close to each other. We also observe that (visually) GAN generated data is much more scattered. This is because GAN data is generated by minimizing the RMSE. However, this might not be ideal

as low RMSE can also have low `ccf` which indicates less similarity between the generated data and the real data. This observation is also made in Figure 15(b) where GAN has low RMSE but very low `ccf at` $lag = 0$. **This shows that GAN might not always be able to capture the trends and patterns of data collected from Twitter**. Ideally, in Figure 15(b), the data points must lie at the top left; i.e., have low RMSE and high `ccf at` $lag = 0$. Most of the data generated using TBAM is in the top left which is a good result. The same result is obtained when the simulations are repeated for the VIRG data set (Figures 16(a) and 16(b)). It should be noted that for the VIRG data set, the standard + event Tweets simulation was not run as the STEM simulation showed that the other three simulations were able to cover the parameters of this simulation.

Figures 17(a), 17(b) and 17(c) show the boxplots for the `ccf at lag=0`, maximum `ccf` and RMSE and how these values vary for the different simulations for the STEM data set. It can be seen that RMSE value for GAN has very low variance. The `ccf` for GAN is low. On the other hand, agent-based modelling generates data independent of RMSE and CCF value. Hence, it is more likely that GAN may generate data that is very close in value to the actual data but may not be able to capture the trends and patterns found in the real data. Therefore, we might conclude that agent-based modelling might be a better option for learning event signatures and patterns for localization. The same observation is made when we look at the boxplots for VIRG (Figures 18(a), 18(b) and 18(c)). However, in this case the RMSE is higher and maximum CCF is lower for GAN generated data making it less suitable than TBAM data.

By looking at the different parameters that give the maximum value of `maximum ccf` or `ccf at` $lag = 0$ or the minimum value of `RMSE`, the optimum set of parameters to generate data using TBAM can be determined. Figures 19(a), 19(b) and 19(c) show the highest values of `ccf at` $lag = 0$ for the different simulations. From our simulations, we have observed that the parameters determined using heuristics were in the set of parameters that had the best value of `ccf at` $lag = 0$ (highlighted in the tables). Hence, heuristics can also be used to generate data that best matches real world data.

4.2 TBAM Model Validation

In order to measure the accuracy of the TBAM generated with the data collected from TBAM we use the *cross-correlation function (ccf)* [57]. Other works have focused on using root mean square (RMSE) to compare data generated through models and real data set. However, the focus of our work is to observe microblogging behavioral changes and to capture the trends and patterns in the data. Hence, *ccf* will be a better metric and give an overview of the similarity between the two data sets.

Figures 20(a), 20(b) and 20(c) shows the *ccf* between VIRG, STEM and GAR data sets and the TBAM generated data respectively. It can be seen that the TBAM generated data and the real data have a very similar pattern as the *ccf* is higher than the threshold for most lags. Another important observation is that the correlation is also high at $lag = 0$. High correlation at $lag = 0$ indicates a strong statistically significance.

For comparison to show that our method does generate accurate data, we measure the *ccf* between data from Twitter and uniformly randomly generated data. The random numbers were generated between 1 and the maximum value in the number of tweets of the specific data set. Figures 21(a), 21(b) and 21(c) shows the *ccf* between VIRG, STEM and GAR data sets and uniformly randomly generated data. It is clearly seen from the plots that there is very low correlation at all lags. This shows that the data generated using TBAM is significantly better.

As shown above, TBAM reasonably reproduces microblogging behavior. Unlike real tweets, here we can (a) identify, control, and tune parameters which impact the tweet counts and microblogging behavioral patterns (b) separate event related tweets from standard tweets without looking at semantics (c) create aggregates in space or time (which we have not done here, but it is straightforward) which can reflect real world settings and provides different space and time granularity (d) intentionally, randomly, or using other models, introduce gaps or noise (e) add additional demographics - groups that tweet more or less (f) add system-wide or regional variations - all in a controlled manner. In this way, there is more control over the delivery slate and reliability of the microblogging data for the purpose of understanding localization of events.

4.2.1 A Comparison of the Affect of Filters on TBAM and Data from Twitter

In Section 3.2.1.1 we considered using low pass filters to remove peaks not related to an event. In this section, we explore different low pass filters to remove peaks not related to events and compare the peaks in the actual and filtered data for the data obtained from Twitter and TBAM generated data and see how well we can recover *significant peaks*. Figures 22(a), 22(b) and 22(c) shows the comparison between the real and TBAM data after it is passed through a simple low pass filter and how the peaks change. The corresponding *ccf* are shown in Figures 23(a), 23(b) and 23(c). It can be observed that the *ccf* after filtering is much better than *ccf* before filtering. Hence, a major potential application for TBAM data can be for training and validation of event detection models, especially that use the notion of peaks as manifestation of events.

To further investigate the effect of filter threshold on peaks in both data collected from Twitter and TBAM generated data, we look at how the different metrics and average `ccf` changes with different filter threshold values for different types of filters. We look at low pass filter, butterworth filter and moving average filter. In addition to filter threshold, the peaks detected are also effected by the peak threshold. Peak threshold is how much a value is above its adjacent values. We look at peak threshold value of 0.5 and 0.02. At low peaks threshold, there would be more peaks detected. Hence, as threshold increases `ccf` decreases as there are more peaks generated and more likelihood of mismatch. After a certain threshold value, the `ccf` becomes constant. The butterworth filter with the sharp cut-off frequencies produces a more smoother plot with less variation in `ccf` values as compared to the low pass-pass filter. The moving average filter works opposite to the butterworth and low pass filter and has an increase in average `ccf` values with increasing filter threshold. Figures 24(a), 24(b) and 24(c) shows the variation of average `ccf` with changing filter threshold.

Figures 25(a), 25(b) and 25(c) shows changing `acc`, `pen` and `qual` for changing filter threshold for butterworth filter at a peak threshold of 0.5. Similarly, we also look at changing filter threshold at a different filter threshold of 0.02 as seen in Figures 26(a), 26(b) and 26(c). At low peak threshold, there are more peaks that are detected and hence, the `acc` is generally lower as there is more likelihood of mismatch in detected peaks. Another point to note is

that at specific threshold value, the accuracy drops drastically. This is might be because the filters remove the event related peaks. This is also the same point after which `ccf` becomes constant and does not change too much.

It is also observed from the results that as the threshold is increased the value of `acc` becomes constant after a certain threshold value and does not vary too much. This means that after a specific filter threshold, the peaks in both TBAM generated data and data from Twitter will match. However, `pen` keeps on increasing with increasing filter threshold. Consequently, this results in `qual` decreasing with increasing filter threshold.

This information can allow us to choose the best filter threshold value. We could choose the point at which `qual` drops the most. That point will give us a threshold value that best preserves the peaks between the TBAM generated data and data from Twitter.

The same observation is made when we look at Figures 27(a), 27(b) and 27(c) which show changing `acc`, `pen` and `qual` for different filter thresholds for low pass filter at a peak threshold of 0.5.

For moving average filter, the affect is the opposite. Again we see a sharp drop in `acc` value at the same point after which `ccf` starts to become constant. Figures 28(a), 28(b) and 28(c) shows changing `acc`, `pen` and `qual` for different filter thresholds for moving average filter at a peak threshold of 0.5.

Table 6: Summary of Metrics

Metric	Ref	Description
Distance	`d`	distance of peak in the TBAM data to the nearest peak in data collected from Twitter
No. of mismatches	`m`	difference in the number of peaks in the data collected from Twitter and the TBAM generated data
Accuracy	`acc`	$acc = 1 - \frac{d}{t_s - 1}$
Penalty	`pen`	$\frac{m}{t_s}$
Quality	`qual`	$1 - \frac{(d+m)}{(2t_s - 1)}$
Cross-correlation function	`ccf`	dependence between two points on time series

Table 7: Summary of Simulations for determining optimum TBAM Parameters

Simulation Name	Parameters Changed
standard + event Tweets	`people`, `event-tweet-chance`, `tweet-chance`
standard Tweets	`people`, `tweet-chance`, `user-interest`
night Tweets	`people`, `night-tweet-chance`, `night-duration`
event Tweets	`people`, `ndist`, `event-tweet-chance`, `event-duration`, `event-interest`

(a) CCF at lag=0 & max CCF

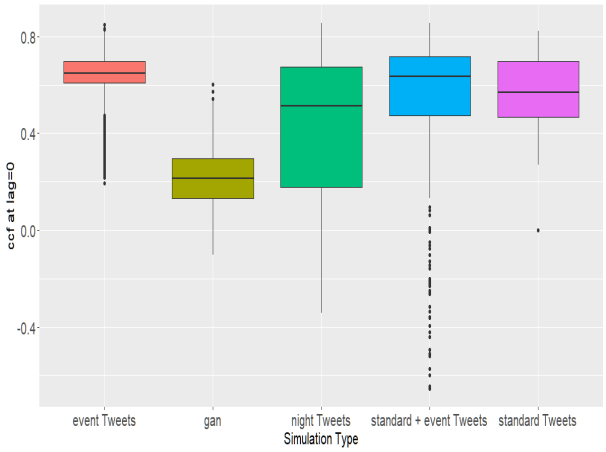
(b) CCF at lag=0 & RMSE

Figure 15: Effect of changing TBAM parameters & GAN on metrics using STEM data set

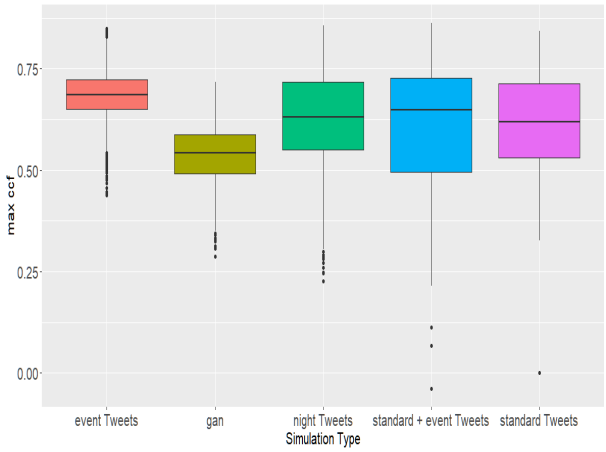
(a) CCF at lag=0 & max CCF

(b) CCF at lag=0 & RMSE

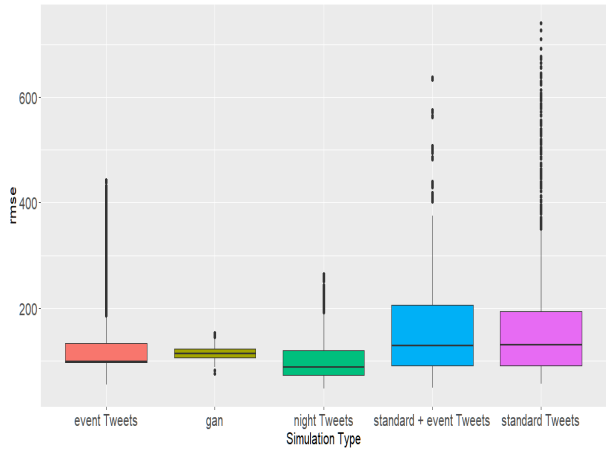
Figure 16: Effect of changing TBAM parameters & GAN on metrics using VIRG data set



(a) Maximum `CCF`

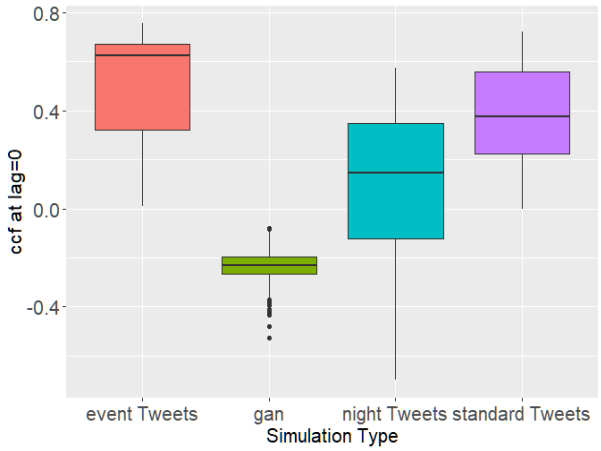


(b) `CCF` at lag=0

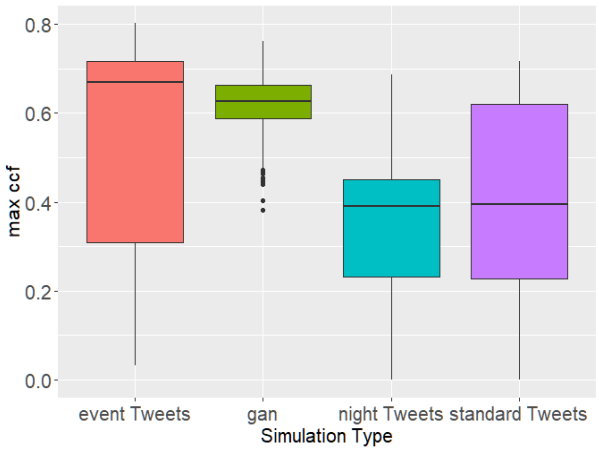


(c) RMSE

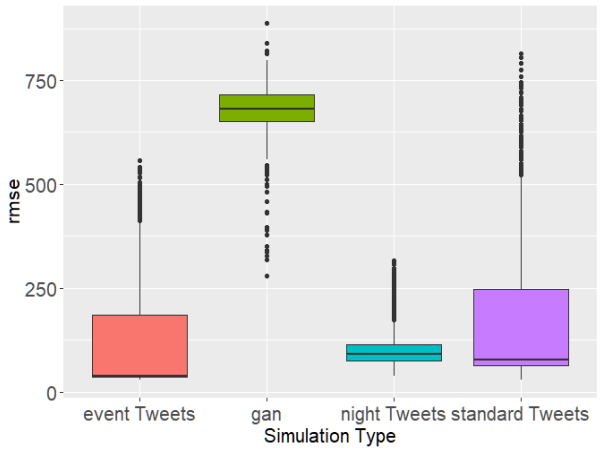
Figure 17: Boxplots for different simulations using STEM data set



(a) Maximum CCF



(b) CCF at lag=0



(c) RMSE

Figure 18: Boxplots for different simulations using VIRG data set

people	ndist	event_tweet_chance	event_dur	event_interest	ccf_max	ccf_0	rmse
3196	1000	0.1	0.5	30	5	0.849222	0.849222	59.97305
3197	1000	0.1	0.5	35	5	0.840904	0.833486	55.61235
3777	1000	0.13	0.75	35	5	0.830988	0.830988	111.3709
4267	1000	0.18	0.95	35	5	0.823742	0.823742	146.4016
3430	1000	0.11	0.6	50	5	0.836531	0.823321	61.6335
5194	1000	0.12	0.4	20	9	0.840402	0.822805	81.34412
3660	1000	0.12	0.7	50	5	0.837283	0.821319	101.1114
3986	1000	0.12	0.85	30	5	0.820766	0.820766	168.8248
7392	1000	0.12	0.4	10	13	0.831808	0.818836	75.56961
3440	1000	0.12	0.6	50	5	0.828143	0.81857	68.30796

(a) Event Tweet Parameters

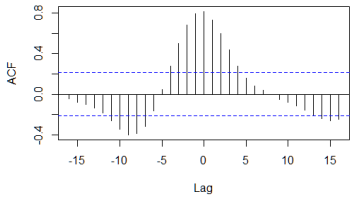
people	user_interest	tweet_chance	ccf_max	ccf_0	rmse
346	500	5	0.3	0.821497	0.821497	89.00843
185	300	5	0.25	0.808008	0.808008	114.456
143	200	13	0.15	0.804912	0.804912	125.4586
64	100	13	0.2	0.804747	0.804747	136.8518
25	100	5	0.25	0.802178	0.802178	138.3301
525	700	9	0.25	0.81583	0.796669	66.23326
27	100	5	0.35	0.842458	0.795966	135.4157
344	500	5	0.2	0.792436	0.792436	100.8386
746	1000	5	0.3	0.7972	0.791881	56.44575

(b) Standard Tweet Parameters

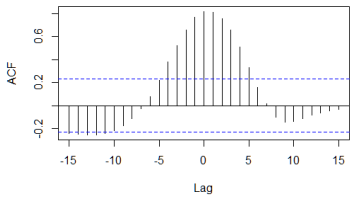
people	night_tweet_chance	night_dur	ccf_max	ccf_0	rmse
79	1000	0.04	7	0.856552	0.856552	47.13904
318	1000	0.14	6	0.836444	0.836444	47.21958
175	1000	0.08	7	0.834322	0.834322	52.04998
367	1000	0.16	7	0.832732	0.832732	49.90658
7	1000	0.01	7	0.83146	0.83146	52.54575
270	1000	0.12	6	0.827394	0.827394	49.39354
31	1000	0.02	7	0.822237	0.822237	52.77742
32	1000	0.02	8	0.840031	0.822026	55.51386
56	1000	0.03	8	0.822228	0.820007	55.69041
391	1000	0.17	7	0.816941	0.816941	51.56851

(c) Night Tweet Parameters

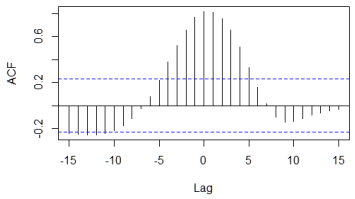
Figure 19: Top values of parameters by `ccf at` $lag = 0$ for STEM



(a) Cross Correlation between VIRG Twitter Data and TBAM data

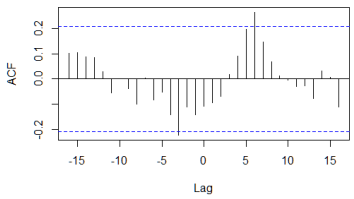


(b) Cross Correlation between STEM Twitter Data and TBAM data

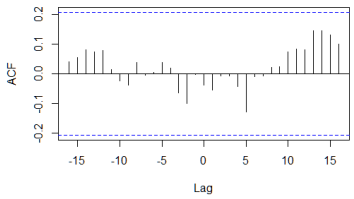


(c) Cross Correlation between GAR Twitter Data and TBAM data

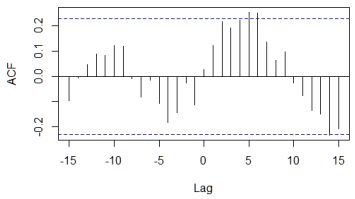
Figure 20: CCF of Real Twitter data with TBAM generated data



(a) Cross Correlation between VIRG Twitter Data and Uniform Random Data

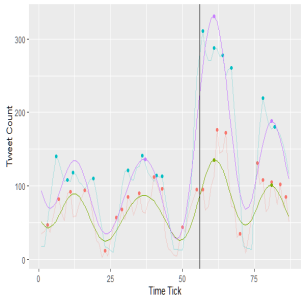


(b) Cross Correlation between STEM Twitter Data and Uniform Random Data

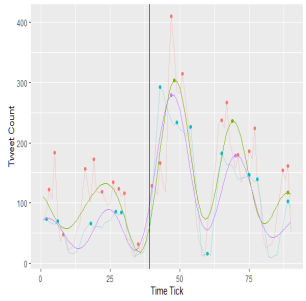


(c) Cross Correlation between GAR Twitter Data and Uniform Random Data

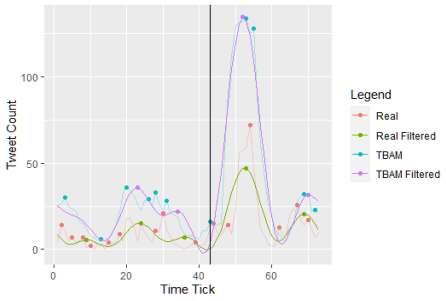
Figure 21: CCF of Real Twitter data with randomly generated data



(a) Plot of VIRG Twitter Data vs TBAM generated data

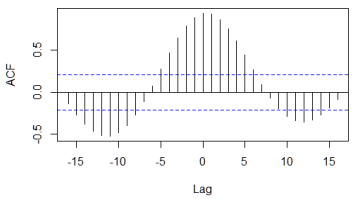


(b) Plot of STEM Twitter Data vs TBAM generated data

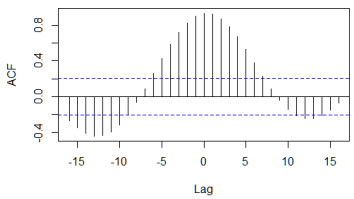


(c) Plot of Garlic Festival Twitter Data vs TBAM generated data

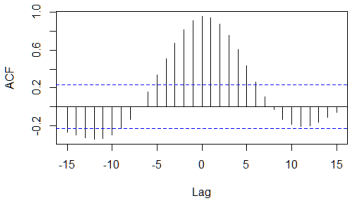
Figure 22: Comparison of Filtered and Unfiltered Real Twitter data with TBAM data



(a) Cross Correlation between VIRG Twitter Data and TBAM data

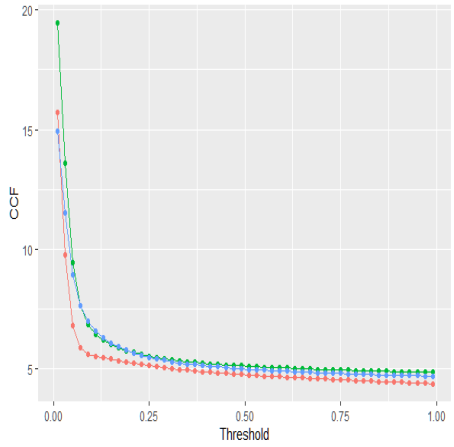


(b) Cross Correlation between STEM Twitter Data and TBAM data

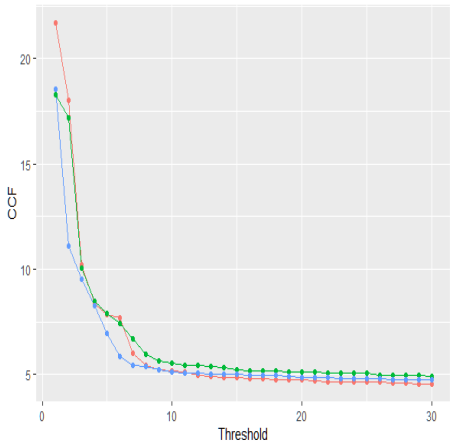


(c) Cross Correlation between GAR Twitter Data and TBAM data

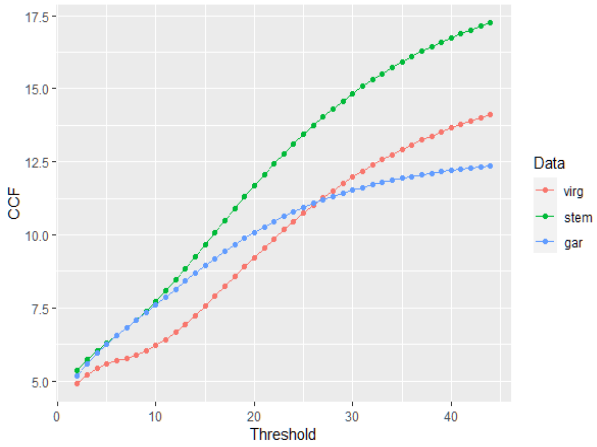
Figure 23: CCF of Filtered Real Twitter data with TBAM generated data



(a) Average `ccf` for Butterworth Filter

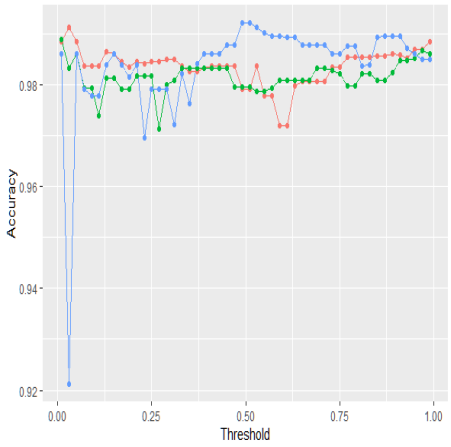


(b) Average `ccf` for Low Pass Filter

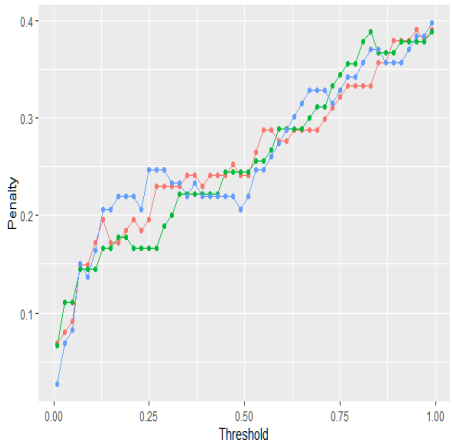


(c) Average `ccf` for Moving Average Filter

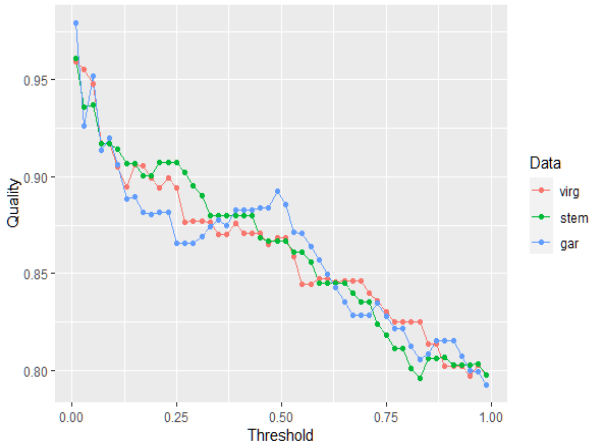
Figure 24: Average `ccf` for Different Filters (Peak Threshold=0.5)



(a) Butterworth Filter Accuracy

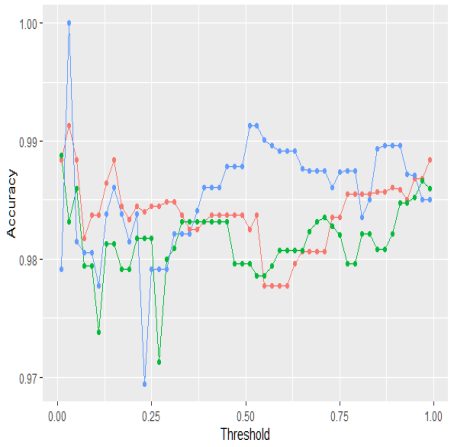


(b) Butterworth Filter Penalty

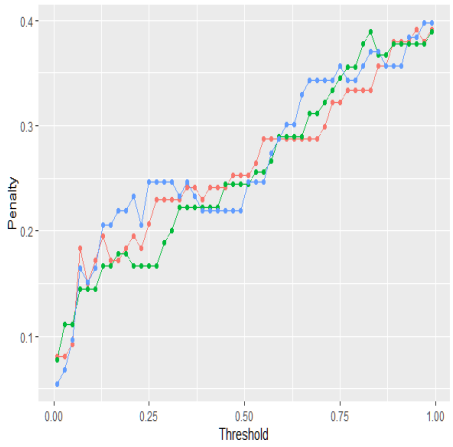


(c) Butterworth Filter Quality

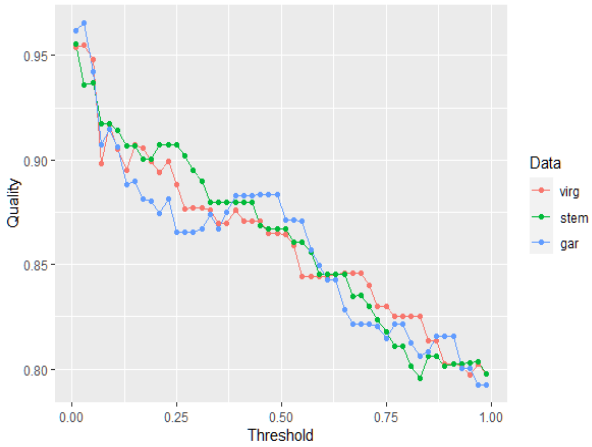
Figure 25: Changing Filter Threshold for Butterworth Filter (Peak Threshold=0.5)



(a) Butterworth Filter Accuracy

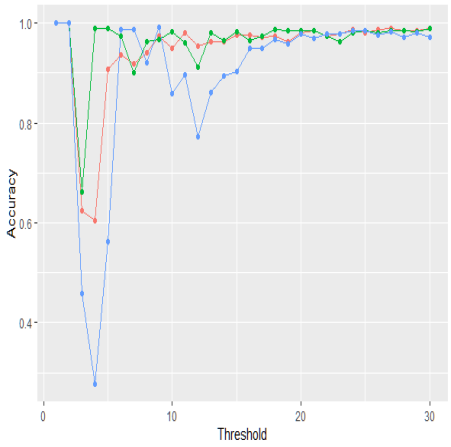


(b) Butterworth Filter Penalty

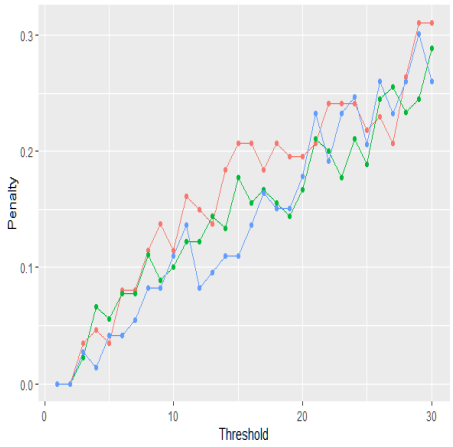


(c) Butterworth Filter Quality

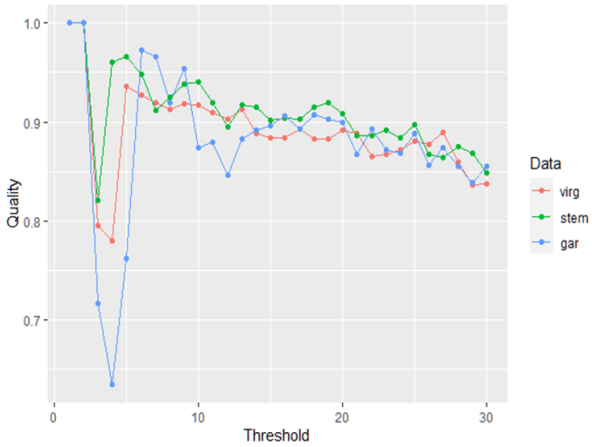
Figure 26: Changing Filter Threshold for Butterworth Filter (Peak Threshold=0.02)



(a) Low Pass Filter Accuracy

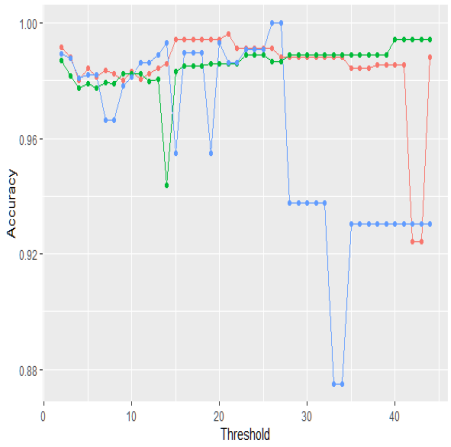


(b) Low Pass Filter Penalty

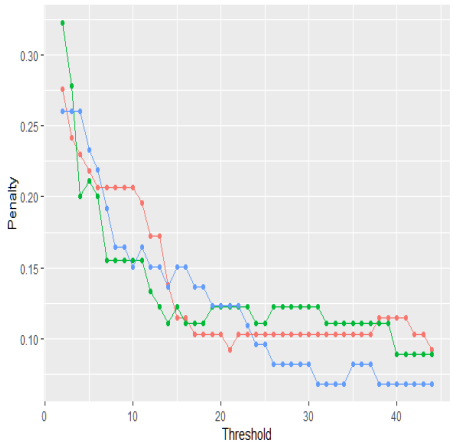


(c) Low Pass Filter Quality

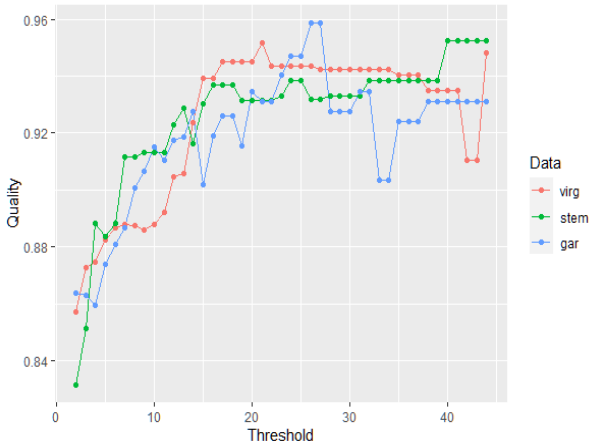
Figure 27: Changing Filter Threshold for Low Pass Filter (Peak Threshold=0.5)



(a) Moving Average Filter Accuracy



(b) Moving Average Filter Penalty



(c) Moving Average Filter Quality

Figure 28: Changing Filter Threshold for Moving Average Filter (Peak Threshold=0.5)

5.0 Event Pattern Detection

In this chapter we aim to answer research question 2, i.e. we show how the enriched data can be used to find patterns that can be used for event localization.

5.1 Using Explicit Patterns

The focus of this section is to find events using explicit patterns in a sequence of number of tweets. We use our definition of peaks to localize an event. We define metrics to measure how well events are detected using explicit signatures.

5.1.1 Measuring disaggregation (reconstruction) quality

The Root Mean Square Error (RMSE) has been used as a metric in previous literature to measure reconstruction quality [38]. Since we define a peak as an event, then RMSE as a metric might not be best able to capture how well reconstructed data is able to best determine the information about an event. Consequently, we introduce area under the curve (AUC) as a metric measuring how well events were identified from disaggregated data (here we know the fine-grained sequence – ground truth – and use an aggregated sequence to test our approach). AUC is obtained from receiver operating characteristic (ROC) curve which is a plot of true positive rate and the false positive rate.

If the position of the peaks in actual and reconstructed data match then $AUC = 1$, which is desirable. As the position of the peaks in actual and reconstructed data changes then AUC would decrease. An AUC of 0.5 indicates random guess.

In some cases people are slow to respond to an event and there might be delay in tweets being sent out regarding an event. To improve AUC in such cases, we introduce a parameter that we call *tolerance (tol)*. It is used in the comparison of the peaks in the original data and predicted peaks which are the peaks in the reconstructed data. When $tolerance > 0$

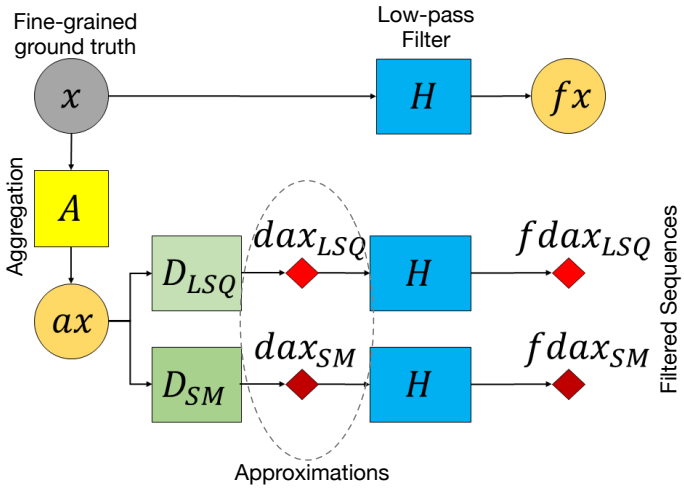


Figure 29: Complete filter/unfiltered framework

then there is more acceptance to errors in peak position in the predicted peaks. If the peaks in the predicted peaks lie within the tolerance value then they are considered to correspond to the actual peak.

5.1.2 Data with ground truth and effect on parameters

In this section we aim to show how the parameters of step, filter threshold and tolerance effect AUC. Our main aim here is to see how well we can reconstruct the peaks in the original data after it is disaggregated and filtered using the different parameters and we use AUC as the metric to measure of how well peaks in the disaggregated data match the original data.

Figure 29 explains our approach in using Tycho data and how we use the reconstructed data to calculate AUC. Here, `x` is the fine-grained input sequence which is the ground truth. The fine-grained data is passed through a low-pass filter `H` to obtain `fx`. The input signal is aggregated using `A` to obtain ax based on Equation 3.4, which is similar in nature to the data obtained from Twitter. `ax` is disaggregated using LSQ D_{LSQ} or LSQ with smoothness constraint D_{SM} to obtain dax_{LSQ} and dax_{SM} respectively. dax_{LSQ} and dax_{SM} are the approximations of the original input series `x`. $fdax_{LSQ}$ and $fdax_{SM}$ are the filtered versions of

dax_{LSQ} and dax_{SM} respectively. This information is summarized in Table 8. The parameters are varied to construct different approximates of `x` and the corresponding AUC values are calculated. A comparison in position of the peaks is made for `x` and dax_{LSQ} denoted as `unFiltered LSQ` and `x` and dax_{SM} denoted as `unFiltered SM` in the experiments. We also compare peak position `fx` and $fdax_{LSQ}$ to indicate `Filtered LSQ` and `fx` and $fdax_{SM}$ to indicate `Filtered SM` in the experiments. Figure 30 shows the effect of changing tolerance, filter threshold and step on AUC.

From the figures we can make the following observations.

- As tolerance increases, AUC also increases. This behavior is predictable as the peak position in the reconstructed data has more allowance and would be considered equal to the actual data peak position even if the exact positions do not match.
- At low filter threshold values, tolerance has no effect on the AUC value (since peaks are removed).
- If we compare LSQ and LSQ with smoothness constraint, then it is observed that LSQ in general is better able to match the peak positions. This can be observed through the high tolerance value required for `unFiltered SM` to have AUC = 1.
- Increasing filter threshold decreases AUC. This happens because as the filter threshold increases, then more peaks are generated in the data due to filtering errors. Hence, there are more chances of mismatch in the position of the peaks. Here again LSQ has generally higher AUC values compared to smoothness which means that disaggregation using LSQ with smoothness constraint results in more mismatch of peak position.
- There is a filtering threshold range within which `Filtered LSQ` has an AUC value less than that with `unFiltered LSQ`. This could be because of peaks being generated due to filtering errors. But in case of LSQ with smoothness, AUC never drops below AUC of `unFiltered SM` which means that LSQ with smoothness is less susceptible to filtering errors.
- Increasing "step" results in a lot of variation in the value of AUC. This is because if a row corresponding to a peak is removed, then there will be a high mismatch in the position of the peaks and a large drop in AUC occurs. In this case LSQ with smoothness has more variation in the AUC values but in some cases has better AUC than LSQ.

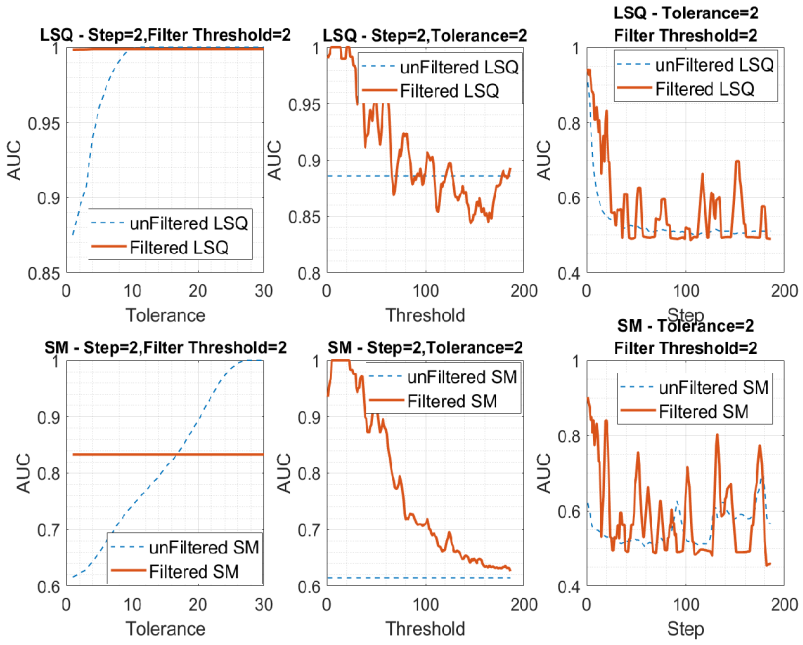


Figure 30: Effect of changing tolerance, filter threshold and step on AUC

These experiments give a good general idea on what effect the different parameters have on the position of the peaks and show that AUC can be a good measure for the quality of disaggregation. The results also give a comparison between LSQ and LSQ with smoothness constraint. Even though LSQ generally has higher AUC value, there may be certain conditions under which LSQ with smoothness performs better. Understanding these conditions and what combination of parameters to choose to get the most accurate results are part of future research. For the dissertation, we simply examine both approaches.

Table 8: Experimental Setup Summary

Term	Description
`x`	input series (spatial or temporal)
`H`	filter
`fx`	Aggregation
`ax`	Aggregated input series x
D_{LSQ}	Disaggregation using LSQ
D_{SM}	Disaggregation using LSQ with smoothness constraint
dax_{LSQ}	Disaggregated `fx` after passing through D_{LSQ}
dax_{SM}	Disaggregated `fx` after passing through D_{SM}
$fdax_{LSQ}$	dax_{LSQ} after applying H
$fdax_{SM}$	dax_{SM} after applying H

5.1.3 AUC as an pattern detection metric

The previous section show how the parameters effect the AUC value. In this section we look at how AUC can be used to detect an event related peak when the parameters are varied. For this purpose we use the FIFA Twitter data. This data set is aggregated and hence, there is no ground truth. The number of tweets are collected for different radii. It should be noted that the FIFA event would be considered as a known event as there is prior knowledge about event information. Figure 4 shows the disaggregated data set obtained with LSQ. The fine-grained granularity is obtained with step = 2 and coarse-grained with step = 7. The reference point $\mathcal{C}_i$ is at a point 2-miles from the actual event. Hence, our ground-truth is an assumed vector where there is a peak at 2-miles point.

From Figure 4 we observe that the event can be seen at the 2.1 mile mark. With a higher step size which equates to coarser granularity the uncertainty in detecting the location of an event is higher. Furthermore, there is another smaller peak at the 2.5 mile marker, which is not indicative of an event. Hence, this peak should be removed for an accurate event detection. The objective here is to see if the smaller peak can be removed by changing the parameters and what effect does the changing parameters have on the AUC. First we change the step and keep filter threshold constant. This allows us to observe how changing granularity effects event detection. Next we change filter threshold and keep step at the finest granularity. The filter threshold is varied to see if the smaller peak is removed and only the significant peak corresponding to an event is left. The corresponding AUC values are then calculated by comparing how close we are able to get to the peak at 2-miles.

Figure 31 shows how step and filter threshold affects AUC. For LSQ, AUC remains at a high value until after a certain step size when it drops to AUC = 0.5 which indicates random behavior. This may be because there are no more peaks at high granularity. However, in case of smoothness there is more fluctuation in the AUC value with changing step value. This shows that AUC is very sensitive to changing step value in case of smoothness. Furthermore, as before, high tolerance value have on average higher AUC then at low tolerance values.

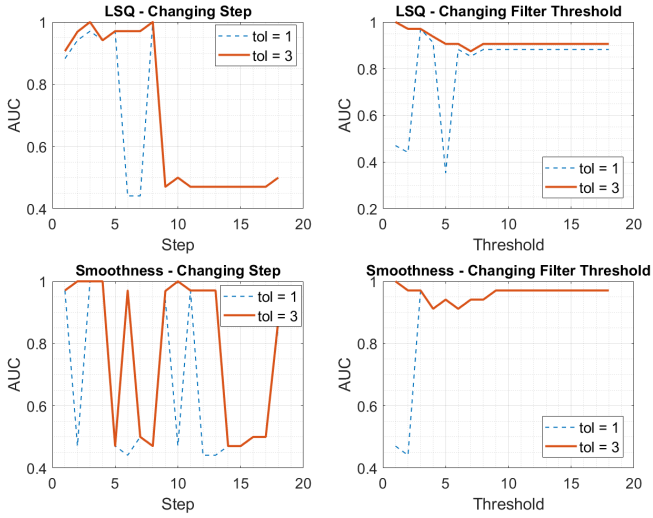


Figure 31: Effect of changing tolerance, filter threshold and step on AUC for FIFA

In case of changing filter threshold, the trend is straight-forward. As filter threshold increases, AUC decreases. This is because at low filter threshold the non-event related peak is removed and only the event-related peak (or the significant peak) remains. As filter threshold is increased, AUC decreases as filtering error is introduced. At a certain filter threshold, the filtered signal no longer changes and that is why there is a constant AUC value. As before from Section 5.1.2 smoothness has higher AUC compared to LSQ which means that the smoothness constraint may improve event detection from disaggregated data.

5.1.4 Using SDF for event-related patterns

The SDF was introduced in Section 3.2.1.2 as a way to improve upon low pass filter in removing peaks not related to an event. It is based on the idea that tweets in event-related peaks are more similar than in non-event related peaks.

The first objective of this section is to verify the first hypothesis enounced in Section 3.2.1.2; namely to confirm if the decay in the aggregated number of similarities (Equation 3.12) in an event-related peak is lower than the decay in non-event peaks. To do so, we compare such decay on three different events described in the data subsection. We used the STEM, VIRG and LON data sets. At each location, we compare the decay obtained on the

Table 9: Decay in number of similarity

Data	λ pre-event	λ post-event
STEM	8.45	1.97
LON	3.91	3.21
VIRG	4.63	2.52

cluster of tweets shared in the 11 hours before the event and the 11 hours following the event for STEM, in the 16 hours before the event and the 16 hours following the event for the VIRG, and in the 11 hours before the event and the 11 hours following the event for LON. The duration over which we made the measurements was predicated by the duration of the event peak i.e., the time frames correspond to the length of the event-related peaks.

Figures 32(a) to 32(c) show the aggregated number of similar tweets (Equation 3.11) for each event at each location for a similarity threshold a_k going from 0 to 0.9. The aggregated number of similar tweets was normalized to values between 0 and 1 for the sake of presentability and comparability. For example, at the first location in the STEM (Figure 32(a)), the number of aggregated similar tweets went down before the event from 1558 for a threshold of 0 to 0 for a threshold of 0.9. After the event, it went down for 33656 for a threshold of 0 to 226 for a threshold of 0.9. During the Virginia shooting, the number of aggregated similar tweets went down before the event from 338028 for a threshold of 0 to 8 for a threshold of 0.9. After the event, it went down for 993755 for a threshold of 0 to 1437 for a threshold of 0.9. On average, the number of aggregated similar tweets at the 0.9 threshold is at 23.66 before the event, and at 478.75 after the event. **This result indicated that the number of nearly identical tweets is 20 fold higher in an event-related peak.**

Table 9 compares the decay (Equation 3.12) in the number of aggregated similar tweets as the threshold increases pre and post-event (The decay of the curves in Figures 32(a) to 32(c)). As the table shows, for all the data we tested, the decay is, for all but one location in the London attacks, lower during event-related peaks. On average, the decay is 3.02 for

Table 10: Event time according to the significant peak and the peak in SDF

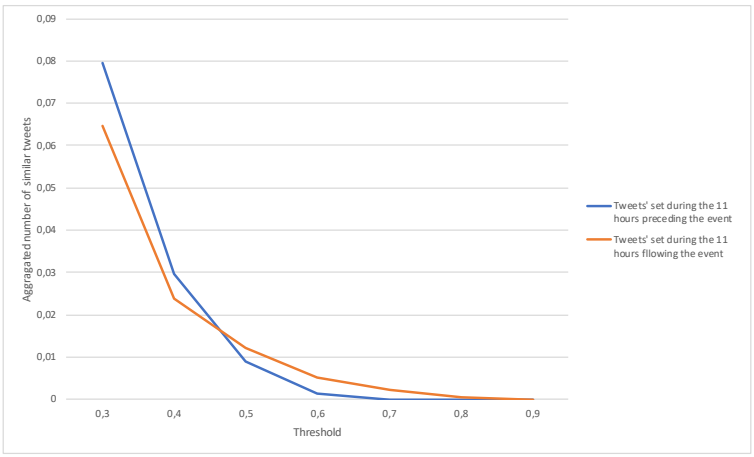
Event	Event time	Significant peak time	SDF peak time	Difference between peaks
STEM	13h53	2h03	2h03	0h00
LON	22h06	19h40	22h40	21h00
VIRG	16h04	23h31	23h31	0h00

event-related peaks and 4,99 pre-event, which represents a 1.65 fold decrease. This result confirms our hypotheses that decay in the aggregated number of similarities in an event-related peak is lower than the decay in non-event related peaks.

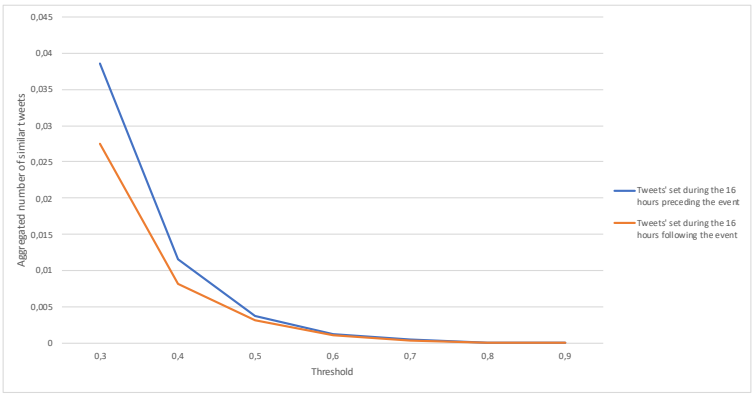
Figures 33(a) to 33(c) divide the time over which the data collection was made into windows t of length 1 hours (Figure 33(c) and 33(b)), and 5 hours (Figure 33(a)). We choose those windows for each data set based on the limitation of the required minimum number of tweets (no less than 2) present in each time window. The figures show the number of tweets (blue line) together with the decay (Equation 3.12) in the number of similar tweets (orange line) in each time window. The figures show a depression of the decay coinciding with an event-related peak. It is important to note that this depression in the decay does not occur around other peaks in the number of tweets outside the event-related peak. However, the figures show regular depressions in the decay occur every day in the early morning (between 5 and 7 am). This depression coincides with a low traffic time on Twitter at which a minimum amount of tweets is shared in the areas of data collection (less than ten tweets). Upon examination of those tweets, we noticed that the majority are automated and identical weather reports, thus the low decay. We mentioned the drawback of only using the decay as a filter at the end of the Section 3.2.1.2 which led us to define the function SDF.

Figures 34(a), 34(b) and 34(c) show the function SDF as a function of time for the considered event. For all figures, we chose the parameters $\alpha = 1$ and $\beta = -1$ in Equation 3.13. Those parameters grantee that SDF peaks around the event-related peak. As the Figures show, SDF reaches a value higher than zero only around the event-related peak for all the events and thus can be viewed as an indicator. Table 10 compares the real-time of

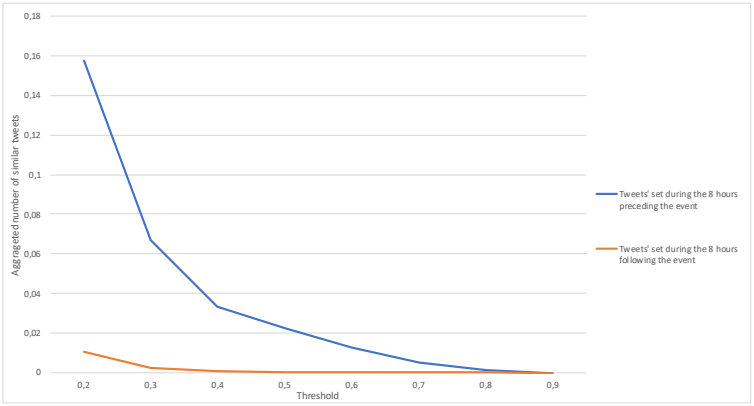
the event with the time at which the significant peaks occur (see significant peak detection section) and SDF peaks. Note that, since we use a 5 hour time window for STEM, for example, the time of the peaks is the end of the time window (2h03 for STEM). However, the tweets were taken between 21h33 and 2h03. The Table 10 also shows, in the case of LON where the event-related peak is submerged by other peak, if we follow the significant peak approach, the event-related peak would be located 12h after the event. However, the SDF can accurately detect the cluster of tweets related to the event, and peaks 36 minutes after the event.



(a) STEM

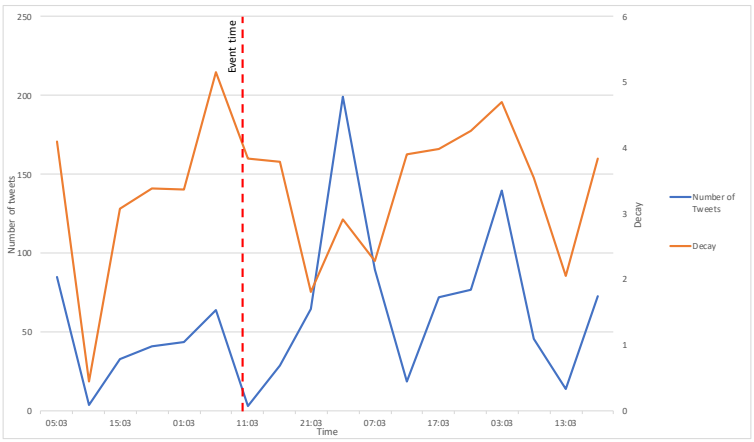


(b) VIRG

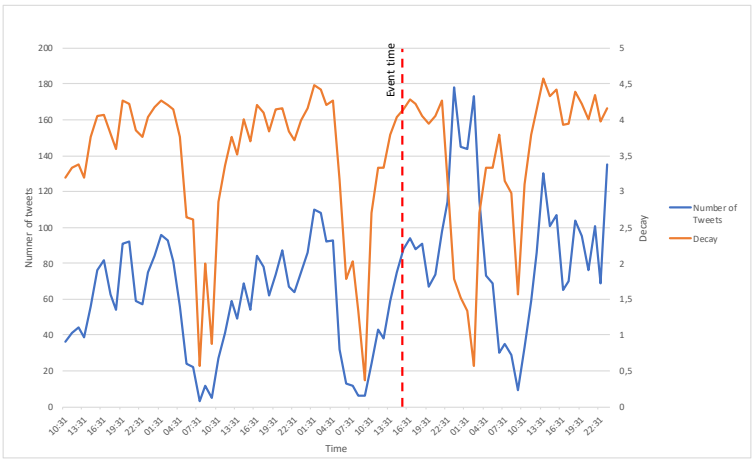


(c) LON

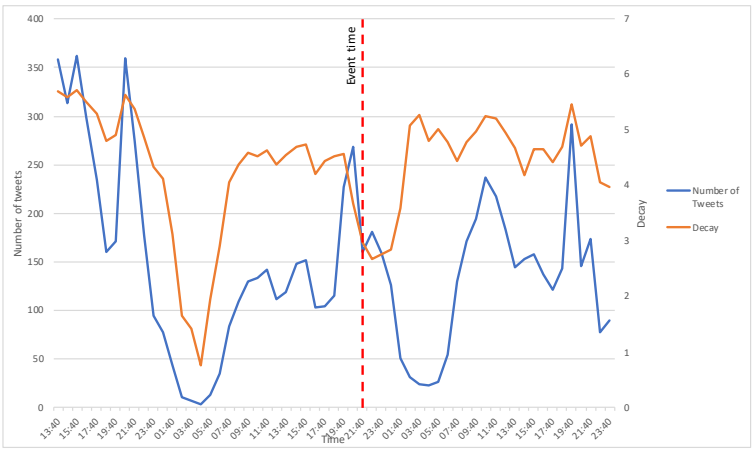
Figure 32: Normalized aggregated number of similar tweets as a function of the threshold



(a) STEM

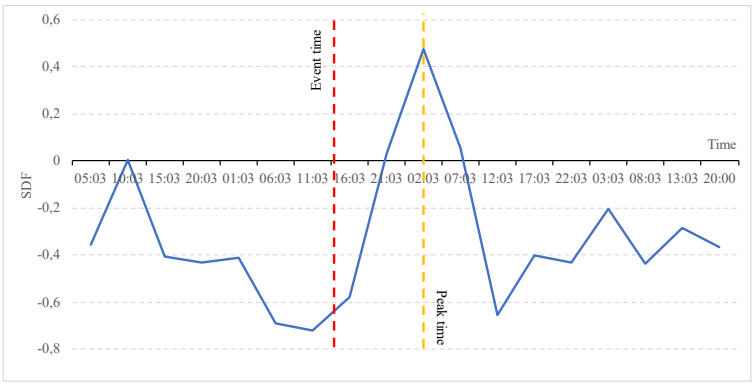


(b) VIRG

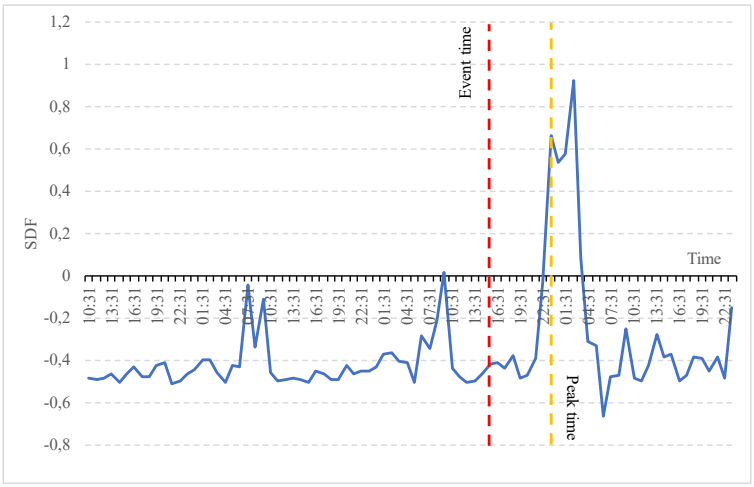


(c) LON

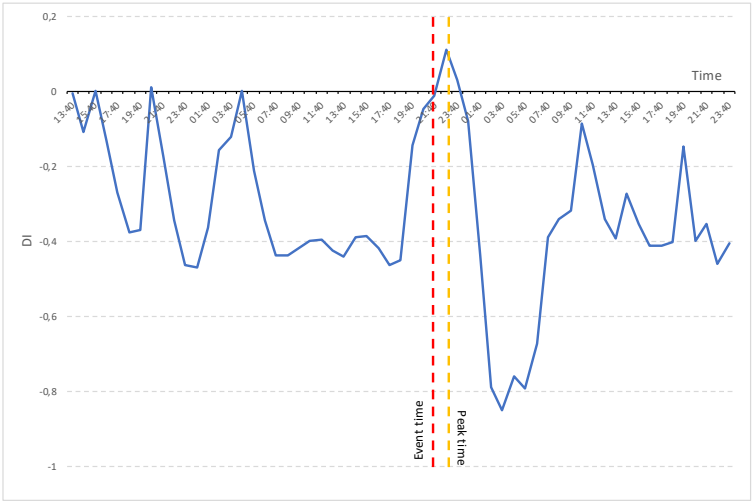
Figure 33: Decay as a function of time for STEM, VIRG and LON



(a) STEM



(b) VIRG



(c) LON

Figure 34: SDF as a function of time for STEM, VIRG and LON

5.2 Using Latent Patterns

In this section we use GAN to generate data and describe our methodology for finding event signatures using LSTM from GAN generated data. In order to effectively discover latent patterns, we create a `spatial-temporal grid (STG)`. The `STG` is shown in Equation 5.1. `STG` represents the disaggregated number of tweets in each layer, along the column, within a specific time window, represented along each row.

$$STG = \begin{bmatrix} s_{11} & \cdots & s_{1j} \\ \vdots & \ddots & \vdots \\ s_{i1} & \cdots & s_{ij} \end{bmatrix} \tag{5.1}$$

Each element s_{ij} represents the number of tweets within a time window in a specific layer.

5.2.1 Latent pattern methodology

The inspiration for our methodology comes from image processing where images are presented as 2-dimensional matrices and then augmented using GAN. The augmented images are then used to train machine learning methods so the trained models can be used for image recognition [26].

Our methodology to learn latent patterns can be summarized in Figure 35. The Twitter data is converted into an STG. Then GAN is used to augment the original data which is then used to train Bi-LSTM. The `STG` copies generated using GAN are denoted $\hat{STG}_i$ where i is from 1 to n and n are the total number of copies generated.

The augmented data is then used to train the Bi-LSTM model. But before training the augmented data needs to be assigned a label. Since the data collected from Twitter is unlabeled, many strategies can be adopted to assign labels, like assign label by manually inspecting every Tweet and assigning a label based on its content.

For this paper, we develop a strategy called `Pre-Post Labeling (PPL)` which is shown in Figure 36. PPl is a simple labelling strategy that would not depend on the content of the Tweets. According to `PPL` each $\hat{STG}_i$ is first divided into pre-event and post-event matrices.

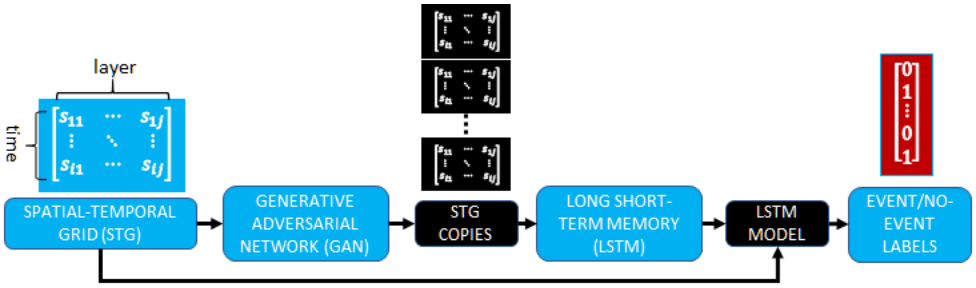


Figure 35: Finding Event Patterns

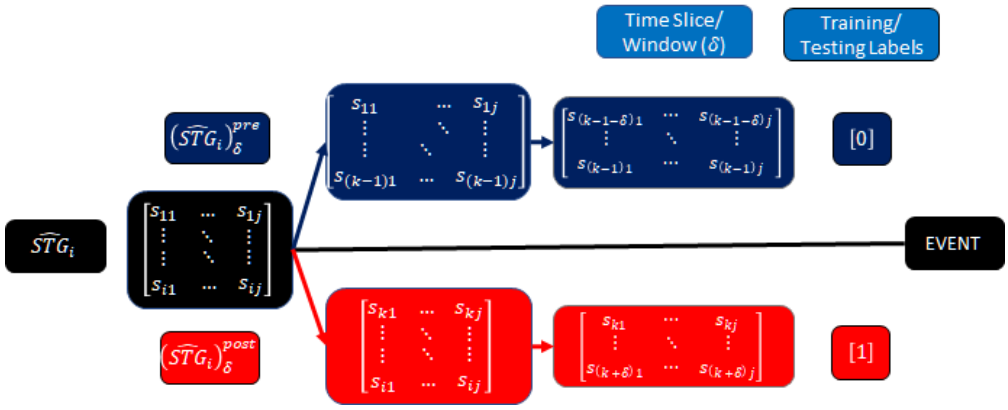


Figure 36: Methodology for Assigning Training Labels - `Pre-Post Labeling (PPL)`

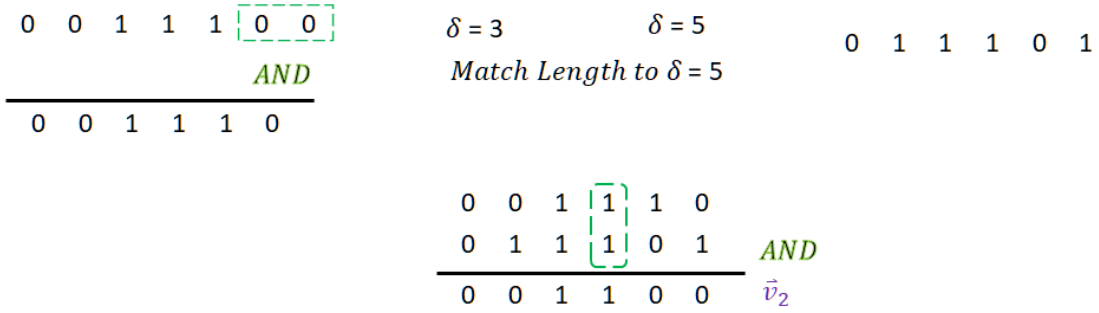


Figure 37: `AND Combine (AC)` Example with 2 different δ

Then we extract δ time slice before the event and δ time slice after the event for each $\hat{STG}_i$. Each time slice before event is denoted $(\hat{STG}_i)_\delta^{pre}$ and time slice after the event is denoted $(\hat{STG}_i)_\delta^{post}$. The time slice before event is given a label of 0 and time slice after event is given a label of 1. We repeat the above method for multiple δ values. The Bi-LSTM model is trained on data for different δ values.

The trained Bi-LSTM model can then be used to find patterns and signatures in the STG from real Twitter data. For testing, the STG is divided into sliding windows of size δ. These δ values represent time slices (in hours) before and after the event. Figure 39 shows how the STG is divided into time slice of size δ. For each time slice of size δ, the model assigns a label of 1 if an event signature is detected or 0 is no event signatures are detected. In this way a label vector $\hat{v}_\delta$ is obtained. We use different δ values to divide the STG and tested on Bi-LSTM trained using different values of δ to obtain different $\hat{v}_\delta$ label vectors.

The different $\hat{v}_\delta$ are combined to obtain. a more accurate label vector. We use the following techniques for combining $\hat{v}_\delta$:

- The first method, called `AND Combine (AC)`, takes the logical `AND` of all $\hat{v}_\delta$ to obtain a label vector $\vec{v}_k$, where k are the number of δ values to generate $\hat{v}_\delta$. For example if $k = 2$, then the label vector v_2 was created using two different $\hat{v}_\delta$. If there is a difference in length, then the length is matched by taking `AND` of all the elements that are in the longer length vector. Figure 37 shows an example of how `AC` is implemented when $k = 2$.

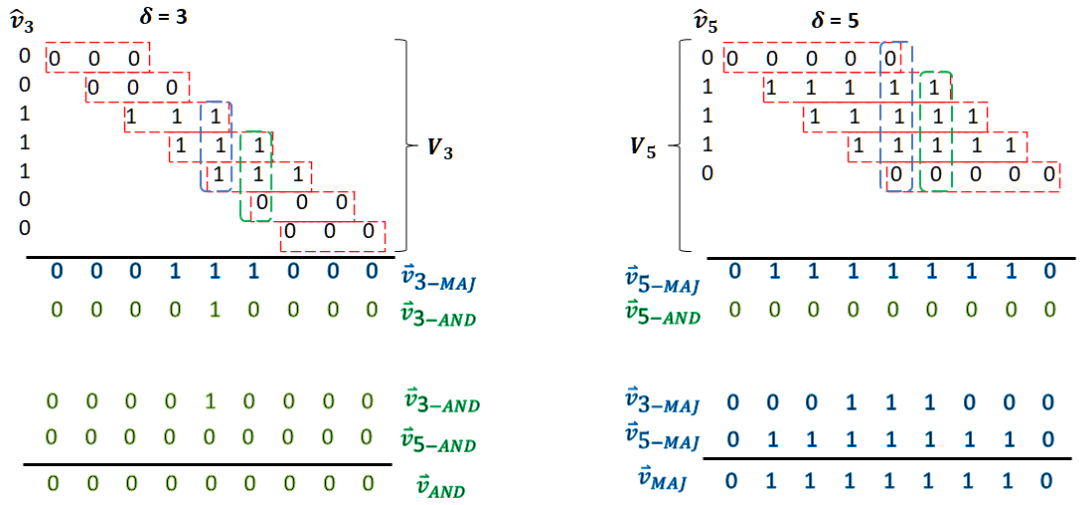


Figure 38: `Matrix Combine (MC)` Example with 2 different δ

We implement `AC` for different combinations of $\hat{v}_\delta$. These methods are denoted `AC-2WINS` for combining 2 different $\hat{v}_\delta$, `AC-5WINS` for 5 different $\hat{v}_\delta$ and `AC-7WINS` for 7 different $\hat{v}_\delta$.

- In the second method, called `Matrix Combine (MC)`, a new matrix ,$\boldsymbol{V_\delta}$ is created by taking δ repetitions of each element in $\hat{v}_\delta$ shifted by 1. Figure 38 shows an example for the MC method using two δ values. It should be noted that the number of rows of $\boldsymbol{V_\delta}$ is equal to the length of $\hat{v}_\delta$. Depending on how we combine $\hat{v}_\delta$, `MC` splits in the following two methods:
  - `MC-AND`: *AND* of each element in column in $\boldsymbol{V_\delta}$ to generate $\vec{v}_{\delta-AND}$. The method is repeated for different δ values to generate multiple $\vec{v}_{\delta-AND}$. We take the logical `AND` of all the $\vec{v}_{\delta-AND}$ to obtain $\vec{v}_{AND}$.
  - `MC-MAJ`: In this method we first count the number of $0s$ and $1s$ in the columns of $\boldsymbol{V_\delta}$ to create a label vector $\vec{v}_{\delta-MAJ}$. If there are more $0s$ than $1s$ in the column then a 0 is assigned and if there are more $1s$ than $0s$ then a 1 is assigned to that position in $\vec{v}_{\delta-MAJ}$. The method is repeated for different δ values to obtain multiple $\vec{v}_{\delta-MAJ}$. We implement majority voting again on all the $\vec{v}_{\delta-MAJ}$ to obtain $\vec{v}_{MAJ}$.

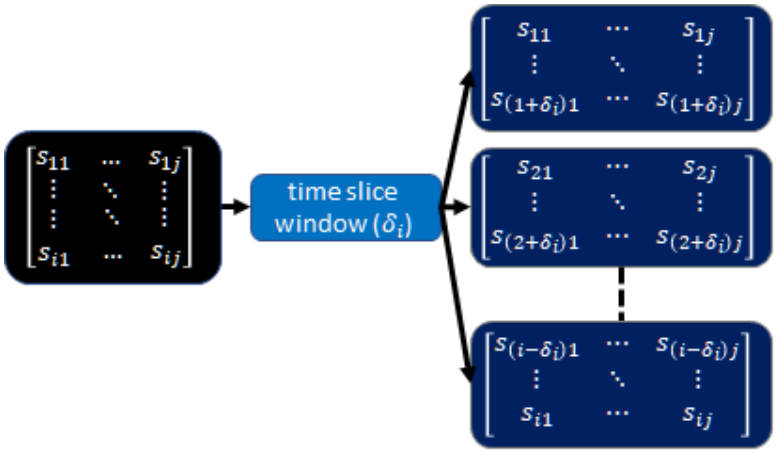


Figure 39: Dividing STG into time slice (window) of size δ

In this way there are three different binary label vectors $\vec{v}_k$, $\vec{v}_{AND}$ and $\vec{v}_{MAJ}$. Next we introduce different metrics to asses how effective the Bi-LSTM is in assigning event labels to the test data.

5.2.2 Latent pattern analysis

In this section, we look at the efficacy of our framework in finding event signatures by conducting different experiments. We follow the methodology shown in Figure 35. We first augment the data collected from Twitter by using GAN to generate multiple copies of the data. Next the augmented data is divided into different time slices of size δ as shown in Figure 36. We use 7 different δ values which are $1, 5, 7, 10, 13, 15$ & 20. The data serves as training data for the Bi-LSTM model. In addition, we use different training data sizes to train the Bi-LSTM model. The training data sizes are `1 dataset, 2 data set, 3 dataset` and `complete data set`. `1 dataset` refers to using GAN augmented Twitter data collected from a single reference point for a single event only to train the Bi-LSTM model. `Complete datas et` refers to using the GAN augmented Twitter data collected for all the events in Table 4 for training the Bi-LSTM model. Next we find the event or non-event signatures in the STG generated from Twitter data.

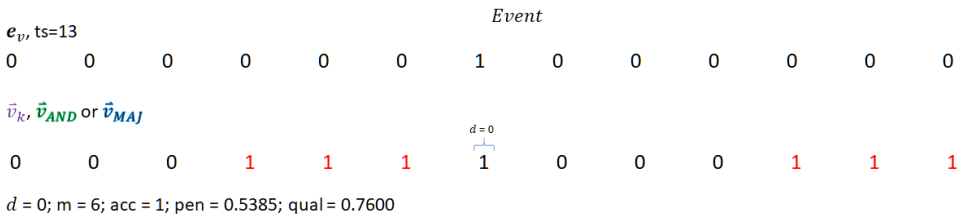


Figure 40: Using Metrics for Latent Pattern Analysis

The STG for each reference coordinate in the event is divided into δ slices as described before. The data serves as testing data for the LSTM model. In this way we generate multiple $\hat{v}_\delta$.

We use the two different methods with different variations that are described in Section 5.2.1 to combine the different $\hat{v}_\delta$ and analyze how effectively the events are identified using Bi-LSTM. The different methods are summarized in Table 11.

In `AC-2WINS`, `AC` method is used to combine two different $\hat{v}_\delta$ ($\hat{v}_1$ and $\hat{v}_{20}$) to create $\vec{v}_2$. Similarly, `AC-5WINS` denoted by $\vec{v}_5$ is generated by combining $\hat{v}_1$, $\hat{v}_5$, $\hat{v}_{10}$, $\hat{v}_{15}$ and $\hat{v}_{20}$. `AC-7WINS` denotes vectors $\vec{v}_7$ created by combining $\hat{v}_1$, $\hat{v}_5$, $\hat{v}_7$, $\hat{v}_{10}$, $\hat{v}_{13}$, $\hat{v}_{15}$ and $\hat{v}_{20}$. `MC-AND` and `MC-MAJ` denote vectors $\vec{v}_{AND}$ and $\vec{v}_{MAJ}$ created by combining all the different values of δ.

The `PKS` creates a binary vector by placing a 1 at the position of peak in STG and places a 0 everywhere else. The `PKS` method serves as a baseline for assessing performance of our proposed method. We used peaks as a baseline because a peak driven approach has been used in previous literature as an indication of event signature ([34], [5]).

The label vectors generated from the above methods are compared with the event vector (e_v) and different values of `acc`, `pen` and `qual` are calculated. However, the definition of these metrics was slightly modified and the labels are represented as vectors and not matrices. For this section we define *distance (d)* is defined as the distance of nearest 1 in $\vec{v}_k$, $\vec{v}_{AND}$ or $\vec{v}_{MAJ}$ from the 1 in e_v and *number of mismatches (m)* is defined as the number of mismatches between e_v and $\vec{v}_k$, $\vec{v}_{AND}$ or $\vec{v}_{MAJ}$. Based on these parameters, we measure `acc`, `pen` and

`qual` as explained in Table 6. Figure 40 shows how the parameters and the metrics are calculated. Finally, we calculate the average value of `acc`, `pen` and `qual` across all the different events.

The results of our experiments for different data sizes are show in Figures 41(a). For this experiment we use the STEM data set and attempt to find event signature in STEM data. The results show that increasing data size does not effect accuracy too much but it does reduce penalty substantially which in turn improves the quality of event detection. Hence, we might conclude that using a large data size that is collected from a different set of reference points for multiple events can substantially improve the detection of event signatures.

We also compare how labels are generated for other machine learning methods which are log regression (LOG), bayes method (BAYES) and k-nearest neighbor (KNN). For training the machine learning methods we used the `Complete data set`, i.e. the model training is done using the augmented data from all the events. The results of our analysis are summarized in Figure 41(b). The figure shows that our method is able to achieve high accuracy which means that we are able to always detect the event signature. The penalty value indicates how often do we wrongly classify an event as a non-event or a non-event as an event. For the Bi-LSTM model the penalty value is generally low. The figure reveals that LOG method performs the worst and PKS method performs the best. Our Bi-LSTM method is close second with `AC-7WINS` having the highest quality among Bi-LSTM methods.

Additionally, for improving the result we generate a new event vector (e_{v1}) which places a 0 for all positions before an event and 1 for all positions after an event. We used this vector instead of e_v for calculating the accuracy, penalty and quality. The result of this event vector is shown in Figures 42(a) and 42(b). The results may seem to be slightly different than for e_v. For `Complete data set` the quality is generally high. For the rest of the data set sizes the quality varies from low to high. This again verifies that large data set size can lead to better identification of event signatures. In case of comparing different machine learning methods, $MC - MAJ$ method has the highest quality indicating that this method is best at identifying at event signatures.

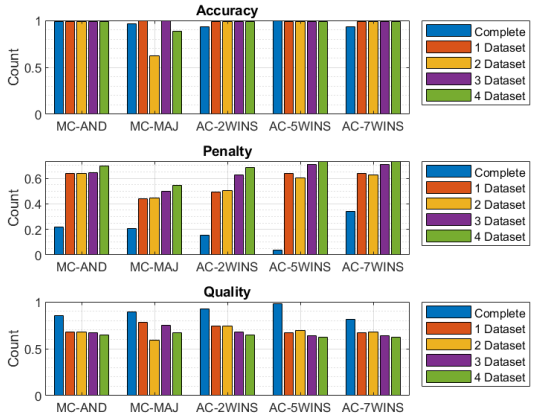
We also compare how labels are generated for other machine learning methods which are

Table 11: Summary of Experimental Setup

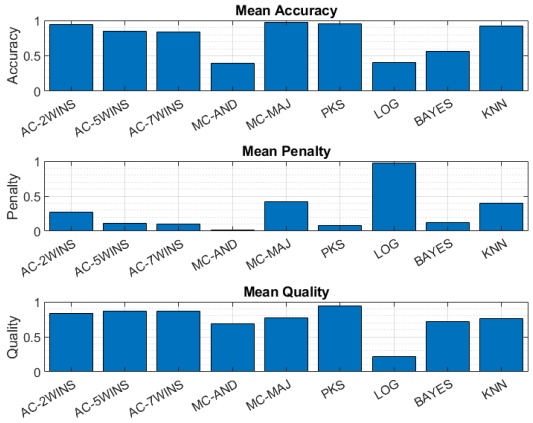
Symbol	**Description**
`AC-2WINS`	2 δ ($\hat{v}_1$ and $\hat{v}_{20}$) values used to create $\vec{v}_2$
`AC-5WINS`	5 δ ($\hat{v}_1$, $\hat{v}_5$, $\hat{v}_{10}$, $\hat{v}_{15}$ and $\hat{v}_{20}$) values used to create $\vec{v}_5$
`AC-7WINS`	7 δ ($\hat{v}_1$, $\hat{v}_5$, $\hat{v}_7$, $\hat{v}_{10}$, $\hat{v}_{13}$, $\hat{v}_{15}$ and $\hat{v}_{20}$) values used to create $\vec{v}_7$
`MC-AND`	$\vec{v}_{AND}$
`MC-MAJ`	$\vec{v}_{MAJ}$
`PKS`	Peak value in STG

log regression (LOG), Bayes method (BAYES) and K-Nearest Neighbor (KNN). The results of our analysis are summarized in Figure 41(b). The figure shows that our method is able to achieve high accuracy which means that we are able to always detect the event signature. The penalty value indicates how often is do we wrongly classify an event as a non-event or a non-event as an event. Finally, quality is a combination of accuracy and penalty. The figure reveals that LOG method performs the worst and PKS method performs the best. However, our Bi-LSTM method has high quality indicating that this method can correctly identify event signatures.

Our method can be easily extended to find the location of an event too. Instead of taking time windows we can create space window slices for different radii (e.g. by taking transpose of STG and using that as input for training and testing Bi-LSTM model). Using the same method but taking slices along the spatial domain, we obtain different labels. The spatial labels are combined using `MC-AND` and `MC-MAJ` methods. The spatial label vectors and the temporal label vectors are multiplied to create a binary label STG_l. In the STG_l, the time and radius or layer at which the event occurred is assigned a 1 and 0 where there are no

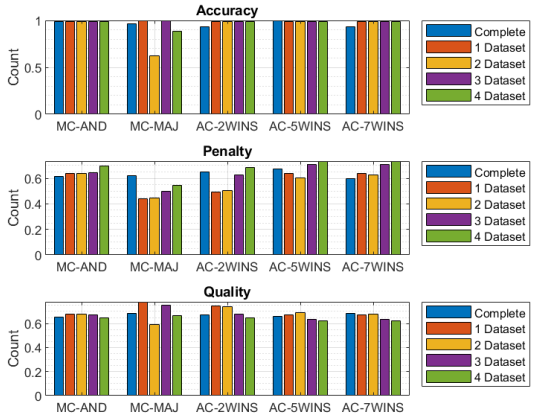


(a) Different Data sizes

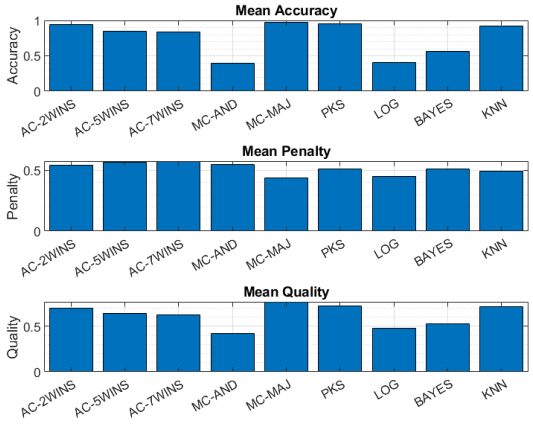


(b) Different Machine Learning Methods

Figure 41: Comparison when using event vector e_v



(a) Different Data sizes



(b) Different Machine Learning Methods

Figure 42: Comparison when using event vector e_{v1}

event signatures. To calculate `acc`, `pen` and `qual` a new event matrix is created. This event matrix ($\boldsymbol{E}$) is created by vector multiplication of temporal event vector (e_v) and spatial temporal vector (e_d). The spatial event vector has 1 at the layer at which event occurred and 0 everywhere else. We add a tolerance parameter which adds event signature (i.e. 1) to time window and layer $\pm$ tolerance. The results are shown in Figure 48(a). Our results reveal that `MC-MAJ` has highest `qual`.

In summary, we used GAN generated data to train LSTM and other machine learning models. Our results showed that LSTM performed reasonably well in determining latent event signatures. It even out-performed KNN (a method used in previous literature). We looked at different data sizes and observed that increasing data size improves `qual` by decreasing `pen`. We also observed that `MC-MAJ` labelling method had the highest `qual`. It was also observed that using different event vectors for measuring `acc`, `pen` and `qual` had a profound affect on these metrics and would require detailed study on how to choose the event vector.

6.0 Localization and Other Applications

Using explicit and latent event signatures we aim to locate an event with greater accuracy. For this purpose, we combine the location of the peaks from multiple reference points and use trilateration to localize an event.

6.0.1 Trilateration calculation

Event localization is the task of finding the location of an event. In Section 3.2.1 we identified a peak as indicative of an event. Let us suppose the peak is located at a distance r_i from $\mathcal{C}_i$. A single $\mathcal{C}_i$ will only allow us to estimate the event as being located in a circle of radius r_i, but not the exact latitude and longitude.

Figure 4(b) demonstrates how multiple reference coordinates ($\mathcal{C}_i$) can be used for locating an event. The reference coordinates are analogous to sensors measuring the number of social sensors sending out tweets. There are multiple methods, like triangulation and trilateration, that can be used to find the location of an event [12]. In our methodology we use trilateration [12] to localize an event with multiple reference coordinates. Trilateration has been used widely in sensor localization and in GPS systems to find the location of a person from 3 or more different satellites. We follow the same concepts of GPS location to find the coordinate of an event. Next we describe how the trilateration calculation is done to find the location of an event.

There are n reference coordinates, $\mathcal{C}_i$ where $i = 1....n$ (also called anchors nodes), whose coordinates are represented in the 2D Cartesian plane as (x_i, y_i) where $i = 1....n$. The unknown coordinate (which is the possible event location) is represented by coordinates $\mathbf{x} = (x, y)$. The distance between the approximate event location and the reference coordinates is the layer at which the significant peak lies. It is denoted r_i for reference coordinate $\mathcal{C}_i$ respectively. The relationship between sensor nodes, approximate event and distances is

represented as a two dimensional matrix:

$$\begin{bmatrix} (x_1 - x)^2 + (y_1 - y)^2 \\ (x_2 - x)^2 + (y_1 - y)^2 \\ \vdots \\ (x_n - x)^2 + (y_n - y)^2 \end{bmatrix} = \begin{bmatrix} {r_1}^2 \\ {r_2}^2 \\ \vdots \\ {r_n}^2 \end{bmatrix} \tag{6.1}$$

This can be represented as $A\mathbf{x} = b$ with:

$$A = \begin{bmatrix} 2(x_n - x_1) & 2(y_n - y_1) \\ 2(x_n - x_2) & 2(y_n - y_2) \\ \vdots & \vdots \\ 2(x_n - x_{n-1}) & 2(y_n - y_{n-1}) \end{bmatrix}$$

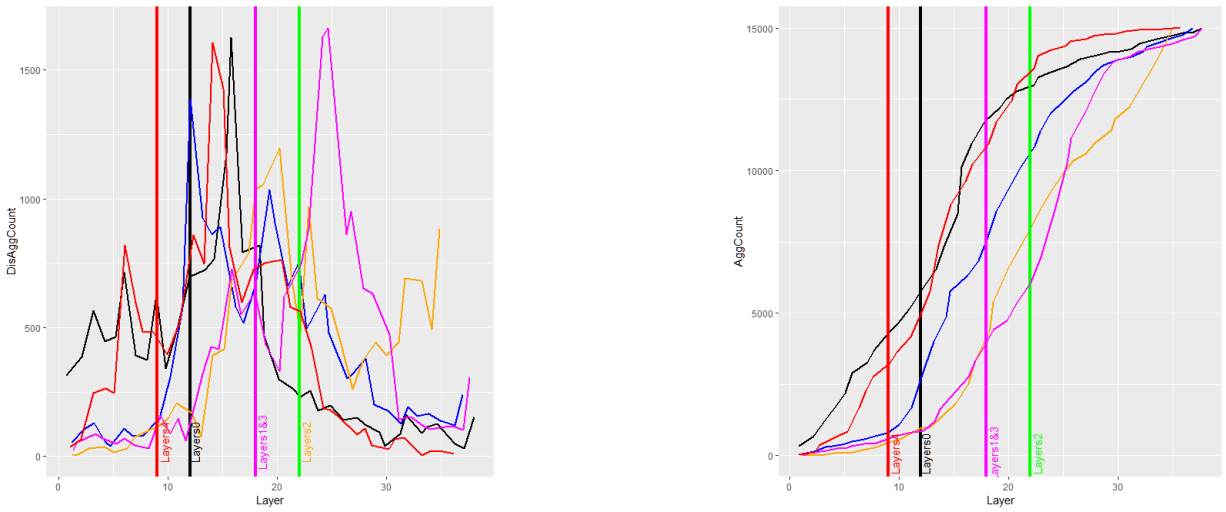
$$b = \begin{bmatrix} {r_1}^2 - {r_n}^2 - {x_1}^2 - {y_1}^2 + {x_n}^2 + {y_n}^2 \\ {r_2}^2 - {r_n}^2 - {x_2}^2 - {y_2}^2 + {x_n}^2 + {y_n}^2 \\ \vdots \\ {r_{n-1}}^2 - {r_n}^2 - {x_{n-1}}^2 - {y_{n-1}}^2 + {x_n}^2 + {y_n}^2 \end{bmatrix}$$

Using least square estimation $\mathbf{x}$ can then be found using $\mathbf{x} = (A^T A)^{-1} A^T b$.

6.0.2 Trilateration with TBAM generated data

In this we attempt to find the location of an event. The event location from the FIFA data set is a radius of 2.1 mile. The error in location of the event location varies from a minimum of 0.1 to a maximum of 4.1 miles. In order to get a more precise location of the event (co-ordinates, not just radius), it would require us to perform the trilateration method. To implement trilateration, it would require multiple $\mathcal{C}_i$. The event location layer from each $\mathcal{C}_i$ is combined to get the event location.

We use TBAM generated data to demonstrate the implementation of the trilateration methodology. First we used random parameters to generate TBAM data. The data was generated with 3, 4 and 5 reference coordinates. The plot of aggregated and disaggregated number of tweets at each layer for 5 reference coordinates is shown in Figure 43(a) and Figure



(a) Disaggregated TBAM Data

(b) Aggregated TBAM Data

Figure 43: Distribution of tweet count with layers

43(b) respectively. The vertical lines labels *layerX* where X is from 0 to 5 indicates the layer where the event occurred, e.g. *layer0* indicates the event layer for reference coordinate 0.

An event was injected at a known location of (7,7). The highest number of tweets recorded was the assumed to be the event peak. The results of the trilateration methodology are summarized in Table 12 which shows approximate event coordinates obtained through triangulation. For localization using triangulation, we choose the layer with the highest peak from Figure 43(a). The results show that localization is effected by number of sensors and localization error increases as number of sensors increase. Nevertheless, it demonstrates the use of TBAM generated data in event localization and a potential avenue for future research for accurate localization of an event.

6.0.3 Trilateration with real data

We focus on the STEM, VIRG and GAR data sets. Figure 4(b) shows how the number of tweets were collected for the STEM data set when using multiple $\mathcal{C}_i$. The $\mathcal{C}_i$ are at known latitude and longitude around the event. The collected tweets were already aggregated

Table 12: Triangulation using TBAM

Number of sensors	**Reference Co-ordinates**	**Approximated Event Coordinates**	**Relative Error**	**Error (miles)**
3 Sensors	(25,25),(25,-25),(-25,25)	(2.56,-1.12)	9.2546 units	0.52535
4 Sensors	(25,25),(25,-25),(-25,25),(-25,-25)	(3.31,-1.87)	9.6087 units	0.5434
5 Sensors	(25,25),(25,-25),(-25,25),(-25,-25),(12,12)	(7.64,-6.20)	13.2105 units	0.7473

and were disaggregated in a similar way to previous sections. For simplicity, the step and tolerance were kept at 1, that is we assume the data is at finest granularity. The filter was not used as we wanted to see to see the actual disaggregated data. Figure 44, Figure 45 and Figure 46 are the plots of disaggregated number of tweets for STEM, VIRG and GAR data sets respectively for 2 different time windows.

The disaggregated data shows multiple peaks which reveals the issue on how to identify the significant peak corresponding to an event. Compared to the FIFA data set, there was one peak with a much larger magnitude than the other peak. The smaller peak could easily be eliminated using filtering leading to higher AUC at low filter thresholds resulting in only a single significant peak. But for STEM, VIRG and GAR data sets there are peaks with similar magnitudes which means that filtering would result in multiple significant peaks. Hence, there is uncertainty in the exact location of the peak corresponding to an event. The plots also consider the distribution of number of tweets over a time window before and after the event (in this case the time windows used are 3 hrs and 12 hrs). It can be clearly seen that before the event and after the event the magnitude of the peaks is much higher. But in some cases the magnitude of the peak before the event is higher than after the event. This

could be because people still have not noticed the event. Nevertheless, these figures clearly show that an event may be located by finding peaks in a stream of tweets. By comparing the position of the peaks before the event and position of the peaks after the event, we can eliminate the peaks to find the significant peak. At this moment we do not delve into how all the peaks can be removed and only choose the ones that are closest in within the 3 hour time window and lying closest to the 2 mile radius which is the actual event location. This problem needs further work.

The parameters used and the results obtained from the trilateration formula are summarized in Table 13. We are indeed able to find the approximate coordinates for the event. By comparing these with the actual coordinates of the event the error in location estimation can be found. The estimated error with no trilateration is the minimum and maximum error when only a single $\mathcal{C}_i$ is used for collecting the aggregated numbers of tweets.

We also compared our trilateration method with one of the methods used in literature for location estimation which is *kmeans* clustering. We use this method as out baseline method. We clustered using the coordinates of the tweets that were sent after the event and used the center of the cluster as the location of the event. Localization of the event using trilateration was more accurate than clustering method.

Furthermore, we used TBAM to generate data in a similar way to the real data. The parameters defined in Table 2 and Table 3 were used to generate the data. Figures 47(a), 47(b) and 47(c) shows the plots of the data generated using the VIRG, STEM and GAR parameters. The plots shows the disaggregated (or the fine-grained) number of tweets in each layer over a fixed time window. Similar to real data, there are 4 reference coordinates, labeled `C1, C2, C3` and `C4` respectively, that collect the number of tweets. The vertical lines show the event layer with the name of the corresponding reference coordinate, e.g. `C1&2` denotes the event layer for reference coordinate `C1` and `C2`. The estimated event location was the position of the peak closest to the actual event location. The reference coordinates and the results of trilateration are summarized in Table 14. It can be observed that the trilateration error is comparable to the trilateration error from the Table 13.

6.0.4 Trilateration with latent patterns

Our method for latent patterns (Section 5.2) can be easily extended to find the location of an event too. Instead of taking time windows we can create space window slices for different radii (e.g. by taking transpose of STG and using that as input for training and testing Bi-LSTM model). Using the same method but taking slices along the spatial domain, we obtain different labels. The spatial labels are combined using `MC-AND` and `MC-MAJ` methods. The spatial label vectors and the temporal label vectors are multiplied to create a binary label STG_l. In the STG_l, the time and radius or layer at which the event occurred is assigned a 1 and 0 where there are no event signatures. To calculate accuracy, penalty and quality a new event matrix is created. This event matrix ($\boldsymbol{E}$) is created by vector multiplication of temporal event vector (e_v) and spatial temporal vector (e_d). The spatial event vector has 1 at the layer at which event occurred and 0 everywhere else. We add a tolerance parameter which adds event signature (i.e. 1) to time window and layer $\pm$tolerance. The results are shown in Figure 48(a). Our results reveal a higher quality for the `MC-MAJ` than `MC-AND`.

The reference coordinates, that monitor the counts of tweets to observe changes in event patterns to identify event signatures, are analogous to sensors measuring the number of social sensors sending out tweets. The position at which the event signature is identified in the labels for spatial data would be radius at which the event was detected at a specific reference coordinate. By combining the positions from multiple reference coordinates, a more precise location of an event can be obtained. There are multiple methods, like triangulation and trilateration, that can be used to find the precise location of an event by combining multiple reference coordinates [12]. Hence, the labels obtained using our method can be used for localizing an event. This demonstrates another possible application for the Bi-LSTM model. We use trilateration [12] to localize an event with multiple reference coordinates. Trilateration has been used widely in sensor localization and in GPS systems to find the location of a person from 3 or more different satellites. We follow the same concepts of GPS location to find the coordinate of an event. The results of our analysis are shown in Figure 48(b). The layer at which the event occurred was found using the `MC-MAJ`. The figure shows the difference between the actual event geographic coordinate and the geographic coordinate

identified using trilateration in miles. The results show that the error results are significantly low which means that the GAN generated data can be used to train Bi-LSTM model and eventually find the location of events.

6.1 Potential Applications

Potential application of this research is development of a stream processing infrastructure. The reference coordinates collect a continuous stream of underdeveloped data that is fed into a data enrichment box. The box should be scalable and monitors the data to look for event patterns. The model described in this thesis can also be extended to other domains as part of the stream processing infrastructure. The methodology can be used with modification in the sensor domain to monitor for "events". This could give rise to exploration of other definitions of events which are not restricted by spatial or temporal dimensions.

6.1.1 Application to more developed data

In the previous sections, we look at underdeveloped microblogging data. We introduced strategies that could localize an event using underdeveloped data. In this section, we look at data that is not completely underdeveloped. We present this data as potential application where we can use such data for discovering event signatures for event detection. We explore the data obtained from human rights documents. The placements of the human rights documents according to the dimensions of underdeveloped data is shown in Figure 48.

The human rights violation documents, unlike microblogging data, is more reliable, semantically more rich and with a more regular delivery schedule. The documents are collected from local authorities and are less subjective. Even though the humans are still sources, the documents are reviewed by expert authorities which makes their information more reliable when compared to microblogging data.

The human rights violation documents contain information about human rights violation aspects, perpetrators committing the crime and the victims of the human rights violation

spanning over various years over different countries. The change in patterns of aspects, perpetrators and victims can be used to learn about behavior of countries. For example, the change in aspect, perpetrators and victims can be used to determine how a country behaves before and after a war. We consider this as part of future work to study these patterns in greater detail.

Using PULSAR [47], a human rights text parser, structured information is extracted from the streams of raw textual information authored and published by hundreds of human rights organizations daily. From these human rights documents, perpetrators and victims, and the human right violation is identified. Figures 50 and 51 shows the count of the aspect feature for different years, extracted using PULSAR for two countries Ethiopia and Tanzania respectively. The aspect feature described the type of human right aspect that was violated. Consequently, Figure 49 is the frequency bar plot of changing aspect for the two countries over the span of six years. The plots clearly show a pattern in the data that can be indicative of an event. For example, in Ethiopia there was a war that lasted from May 1998 to June 2000 and the change in aspect before and after war can be seen.

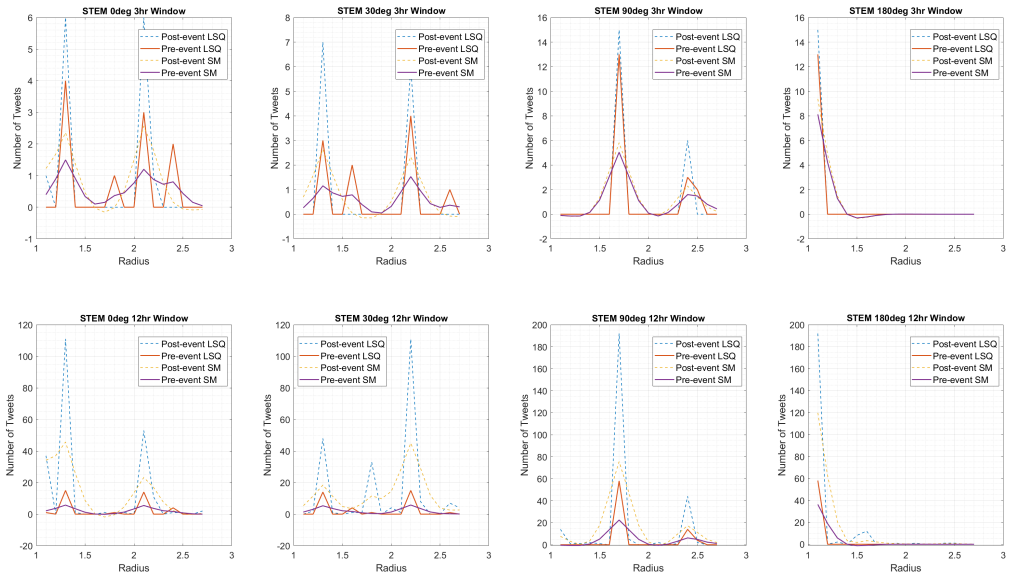


Figure 44: STEM disaggregated with 3 and 12 hr time windows

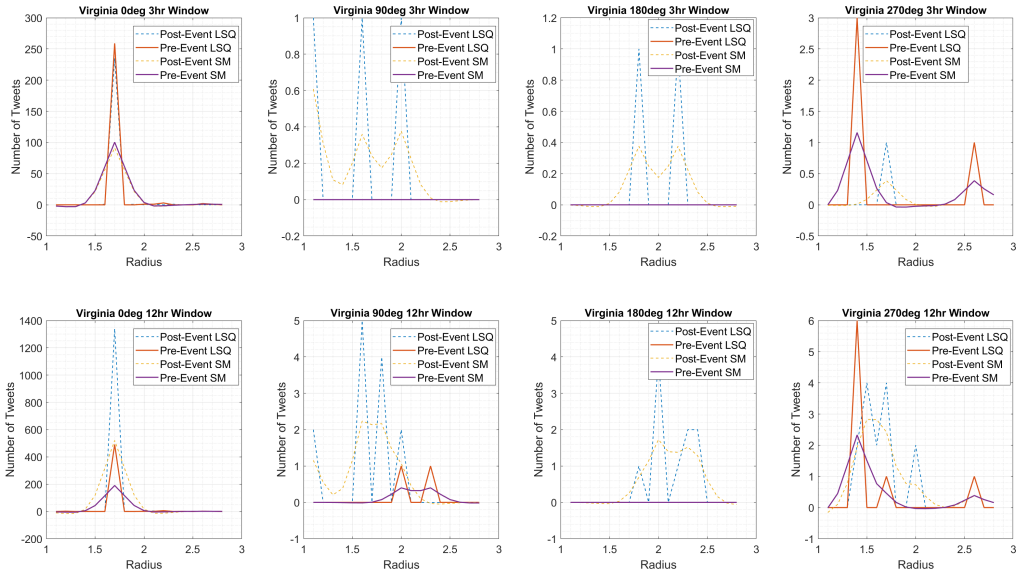


Figure 45: Virginia disaggregated with 3 and 12 hr time windows

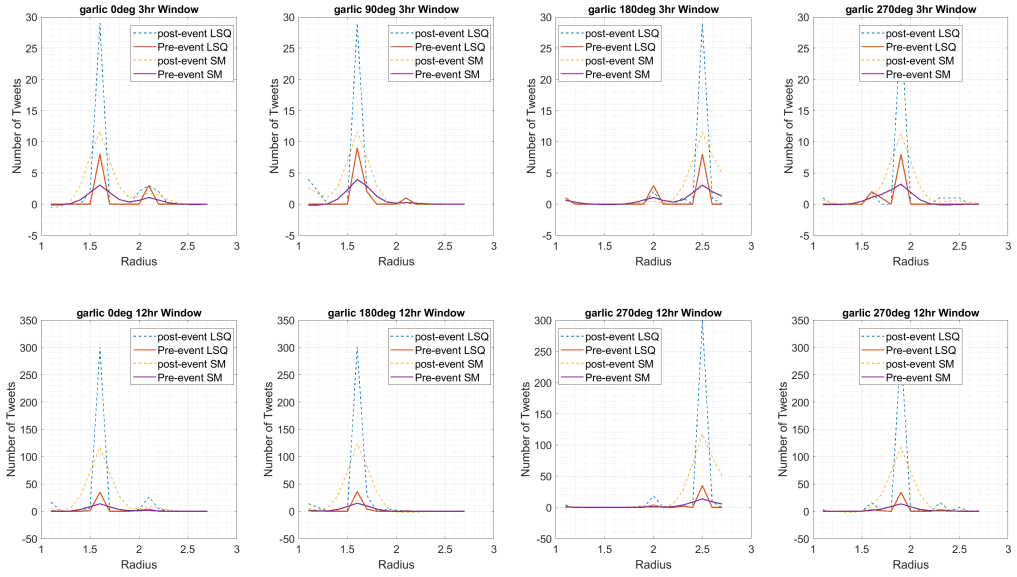
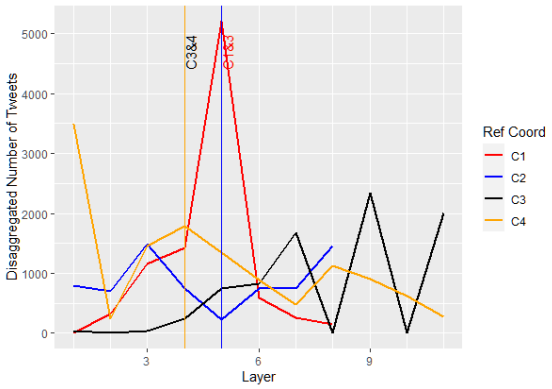


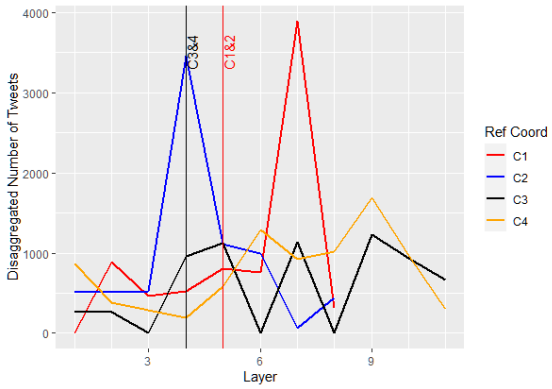
Figure 46: Garlic Festival disaggregated with 3 and 12 hr time windows

Table 13: Trilateration Parameters and Results

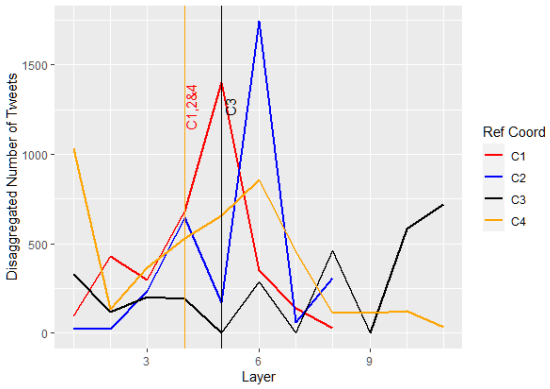
Data Set	**Reference Coordinate (Latitude, Longitude)**	**Significant Peak Location (miles)**	**No Trilateration Error (miles)**	**Trilateration Error (miles)**	**Baseline Error (miles)**
FIFA World Cup Final	$55.73410, 37.57873$	2.1	$0.1 - 4.1$	NA	NA
STEM School Shootings	$39.58482, -104.99790$	2.2	$0.2 - 4.2$	**0.63943**	*1.29801*
	$39.58096, -104.97928$	2.2	$0.2 - 4.2$		
	$39.55599, -104.96067$	1.7	$0.3 - 4.3$		
	$39.53438, -104.99790$	1.6	$0.1 - 3.0$		
Virginia Beach Shootings	$36.75089, -76.02167$	1.7	$0.3 - 4.3$	**0.28902**	*0.53208*
	$36.75089, -76.02167$	2.0	$0.0 - 4.0$		
	$36.72206, -76.05750$	2.1	$0.1 - 4.1$		
	$36.75089, -76.09333$	1.4	$0.6 - 4.6$		
Garlic Festival Shootings	$37.02661, -121.58528$	1.6	$0.4 - 4.4$	**0.47435**	*0.84882*
	$36.99777, -121.54933$	1.6	$0.4 - 4.4$		
	$36.96894, -121.58528$	2.5	$0.5 - 4.5$		
	$36.99777, -121.62123$	1.9	$0.1 - 4.1$		



(a) VIRG parameters



(b) STEM parameters

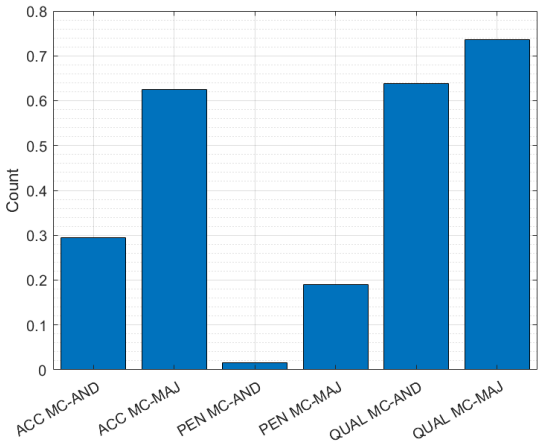


(c) GAR parameters

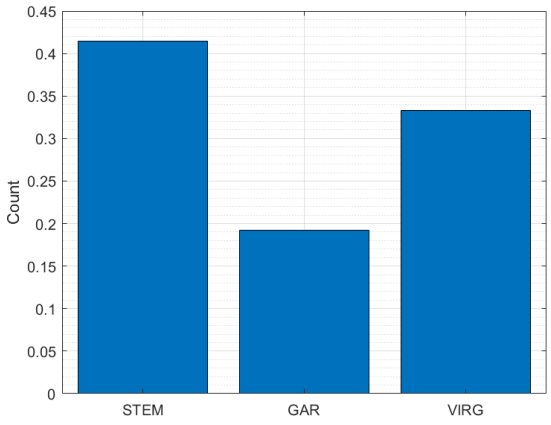
Figure 47: Plots of TBAM generated data along spatial dimension

Table 14: Triangulation using Parameters from Real Data

Parameter	**Reference Coordinates**	**Relative Error**	**Error (miles)**
STEM	(25,1),(-25,1),(1,25),(-1,-25)	1.485 units	0.1187
VIRG	(25,1),(-25,1),(1,25),(-1,-25)	1.993 units	0.1593
GAR	(25,1),(-25,1),(1,25),(-1,-25)	2.173 units	0.1737



(a) Accuracy, Penalty & Quality for STG_l



(b) Trilateration Error (in miles)

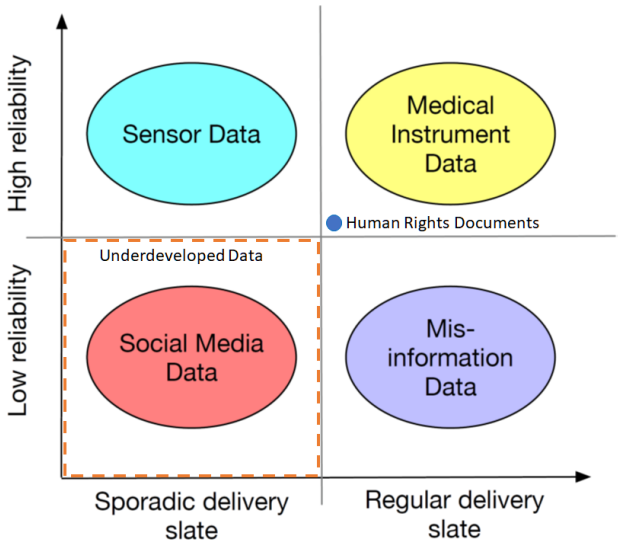


Figure 48: Human Rights Documents as more Developed Data

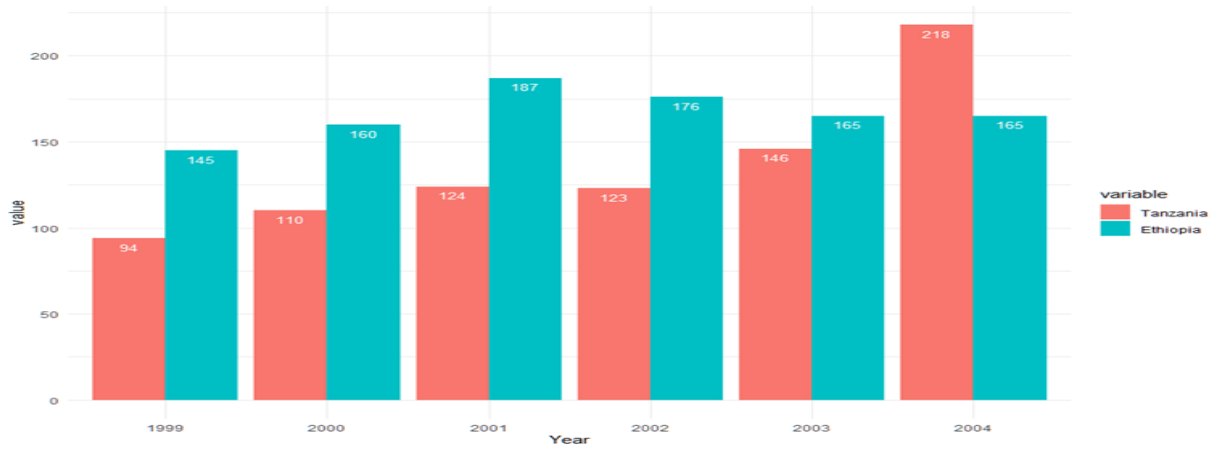


Figure 49: Aspect Change by Year

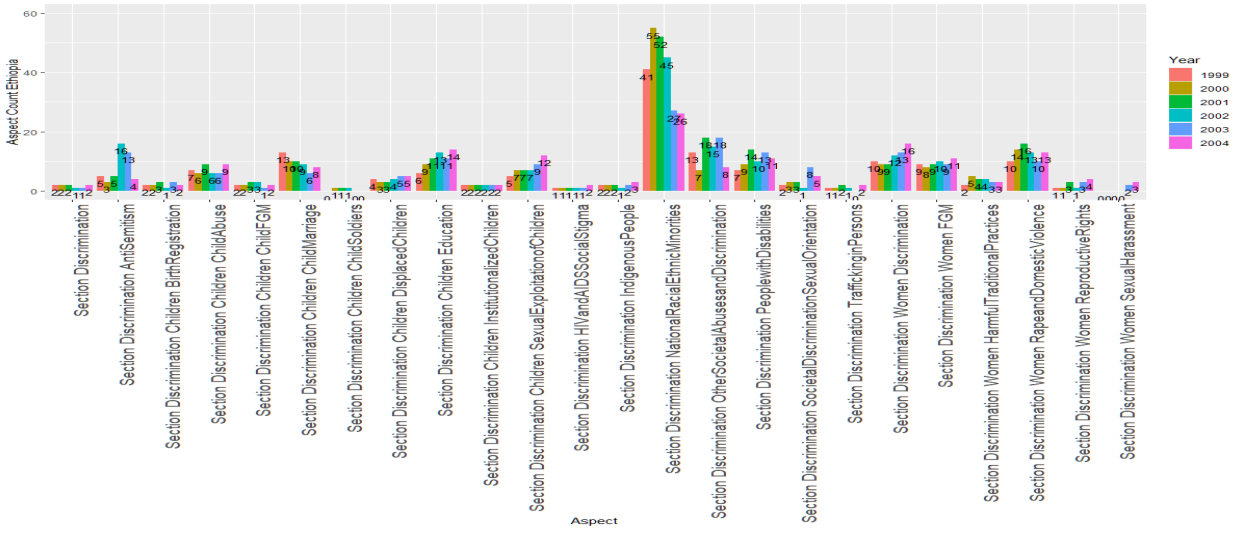


Figure 50: Ethiopia Aspect Count

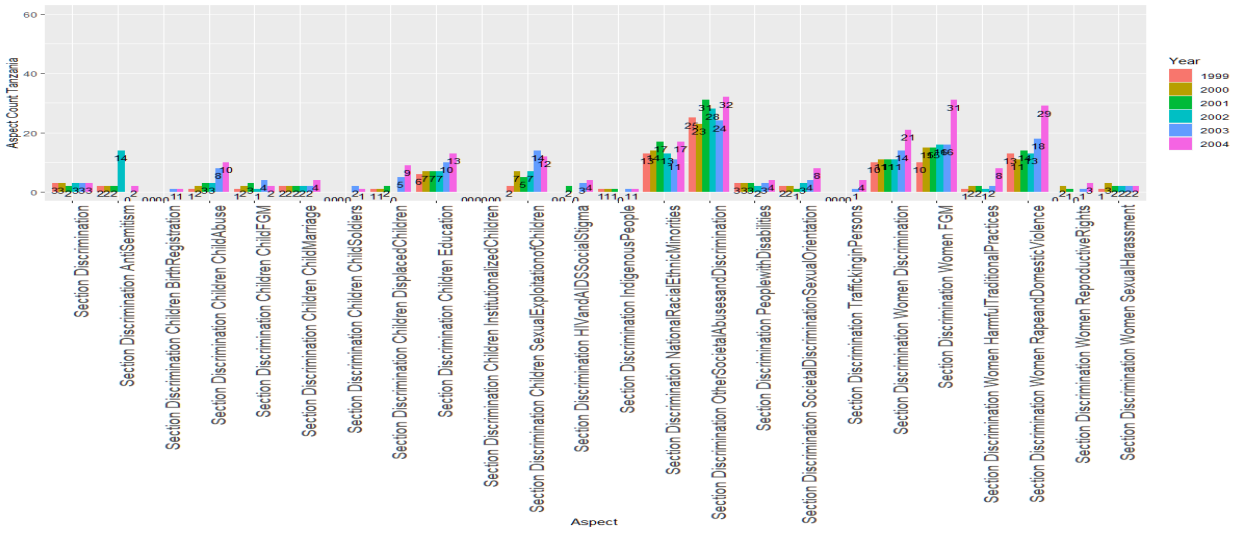


Figure 51: Tanzania Aspect Count

7.0 Conclusion

In this section we discuss our findings and explore some of the future research potential for our research.

7.1 Primary Conclusions of Result

Table 15: Trilateration Summary

Data	**Baseline**	**Explicit Patterns-Real**	**Explicit Patterns-TBAM**	**Latent Patterns-GAN**
STEM	1.29801	0.6394	0.1187	0.42
VIRG	0.53208	0.28902	0.1593	0.34
GAR	0.84882	0.4744	0.1737	0.19

The results of trilateration are summarized in Table 15. Our results show that localization using explicit patterns found in TBAM data has the lowest trilateration error. Using GAN generated data to train LSTM and then discover latent event patterns from the LSTM model for localization has the second lowest trilateration error. Localization with event patterns discovered from disaggregated real data has the third lowest trilateration error. Finally, our baseline method of using center of cluster as event location has lowest localization accuracy. Hence, our enrichment and augmentation methods do improve localization accuracy.

In this dissertation, we only consider a single significant event that occur within a specific time window and geographic location.

7.2 Discussion and Future Work

In this section we explore the different aspects of our work that have not been addressed and are beyond the scope of the research.

In Section 3.1.2 on spatial disaggregation we presented it as a system of linear equations. It should be noted that as the step size increases (or we move to lower granularity), the accuracy of disaggregation decreases. Furthermore, if the step size becomes greater than the number of tweets, then it is likely that there would be no tweets estimated in each annular ring. That is not desirable and can be addressed by changing the domain constraint, such as using a distance-based constraint, which we leave as part of future work.

One question that arises is if there are not enough tweets or what if the event is not "significant" enough for people to tweet about. In that case, there might not be deviation from normal behavior and an event pattern may not manifest itself. Under such circumstances, latent event patterns might be able to identify an event occurrence. But a more quantitative approach to under what condition an event signature may not be identified is beyond the scope of this work.

In the agent-based model described in Section 3.3, we did not consider changing demographics and changing tweeting behavior of a city. Considering these factors could have resulted in data generating using TBAM matching the actual data from Twitter more closely.

In Section 5.2.2 we looked into the training data used for training machine learning models. We showed through Figure 41(a) and Figure 42(a) how size of data effects localization accuracy of event detection. But it would be interesting to see how data from different events and different locations would effect training of machine learning methods and in turn effect event pattern detection. We made assumptions about assigning labels, more specifically the ground truth label. How the different labelling methods compare to each other is left for future work.

In the thesis we have defined multiple parameters, like AUC and CCF, that can effect the accuracy of localization. However, we did not focus on the optimization of the parameters or how to best choose the parameters to obtain the best localization results. Our focus was more on how the different parameters effect the metrics and localization accuracy.

For example in Section 5.1, we looked into how different filter threshold values effects removal of peaks and the AUC metric. But we did not delve into the details of choosing the best threshold value for removing non-event related peaks. By determining the optimum frequency threshold, event patterns (peaks) can be accurately defined, which would in turn improve the localization accuracy.

We also did not focus on optimizing the GAN and LSTM. The primary thesis objective was to show the applicability of how data generated from GAN can be used to train machine learning models so that event patterns may be identified. We did not investigate in depth the selection of optimum machine learning parameters, like number of layers, etc. The optimization of machine learning methods will also be considered as part of future work.

In summary the tasks that we would like to have achieved but leave for future are:

- Exploration of other constraint matrices for disaggregation and other graph models in TBAM
- Optimization of parameters like filter threshold and machine learning parameters
- Understanding the minimum number of tweets that would be required to accurately localize an event (and hence, population density/tweeting behavior)
- Augment models for microblogging with population demographics and evolution of microblogging users over time
- Application specific event pattern location in sequences such as fusing data from traffic (e.g. images) and other microblogging sources (like Flickr, Instagram, etc.)
- Exploration of other labeling techniques and definition of event vector for machine learning algorithms
- Fusing of multiple data sources, not just microblogging data for event detection
- More detailed implementation of pattern recognition and event detection in the other data domains such as, like sensors, human rights document, etc.

7.3 Conclusion

In this dissertation we envisioned a scenario where there would be multiple reference coordinates that would monitor the number of tweets. The data collected from these reference coordinates are underdeveloped. The purpose of this dissertation is to localized an event using underdeveloped data. We showed how microblogging data is underdeveloped and implemented different enrichment techniques. These enrichment techniques include disaggregation and augmentation by generation of data using GAN and ABM. We created metrics to measure how well the data was disaggregated.

Bibliography

[1] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. Eventweet: Online localized event detection from twitter. *Proceedings of the VLDB Endowment*, 6(12):1326–1329, 2013.

[2] Robert John Allan. *Survey of agent based modelling and simulation tools*. Science & Technology Facilities Council, 2010.

[3] Faisal M Almutairi, Fan Yang, Hyun Ah Song, Christos Faloutsos, Nicholas Sidiropoulos, and Vladimir Zadorozhny. Homerun: scalable sparse-spectrum reconstruction of aggregated historical data. *Proceedings of the VLDB Endowment*, 11(11):1496–1508, 2018.

[4] Farzindar Atefeh and Wael Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164, 2015.

[5] Mehdi Ben Lazreg, Usman Anjum, Vladimir Zadorozhny, and Morten Goodwin. Semantic decay filter for event detection. In *17th ISCRAM Conference Proceedings. Blacksburg, VA (USA): Virginia Tech.*, pages 14–26. ISCRAM, 2020.

[6] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[7] Lars-Erik Cederman, Nils B Weidmann, and Nils-Christian Bormann. Triangulating horizontal inequality: Toward improved conflict analysis. *Journal of Peace Research*, 52(6):806–821, 2015.

[8] Carmela Comito, Deborah Falcone, and Domenico Talia. A peak detection method to uncover events from social media. In *Data Science and Advanced Analytics (DSAA), 2017 IEEE International Conference on*, pages 459–467. IEEE, 2017.

[9] Mário Cordeiro and João Gama. Online social networks event detection: a survey. In *Solving Large Scale Learning Tasks. Challenges and Algorithms*, pages 1–41. Springer, 2016.

[10] Kainan Cui, Xiaolong Zheng, Daniel Dajun Zeng, Zhu Zhang, Chuan Luo, and Saike He. An empirical study of information diffusion in micro-blogging systems during emergency events. In *International Conference on Web-Age Information Management*, pages 140–151. Springer, 2013.

[11] Xiang Dai and Heike Adel. An analysis of simple data augmentation for named entity recognition. *arXiv preprint arXiv:2010.11683*, 2020.

[12] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[13] Mark A Davenport, Marco F Duarte, Yonina C Eldar, and Gitta Kutyniok. Introduction to compressed sensing. *preprint*, 93(1):2, 2011.

[14] Xiaowen Dong, Dimitrios Mavroeidis, Francesco Calabrese, and Pascal Frossard. Multiscale event detection in social media. *Data Mining and Knowledge Discovery*, 29(5):1374–1405, 2015.

[15] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[16] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018.

[17] Jason Gale. Googling for gut symptoms predicts covid hot spots, study finds. `https://www.msn.com/en-us/health/medical/googling-for-gut-symptoms-predicts-covid-hot-spots-study-finds/ar-BB18Yw1Y?ocid=uxbndlbing`.

[18] Muskan Garg and Mukesh Kumar. Review on event detection techniques in social multimedia. *Online Information Review*, 40(3):347–361, 2016.

[19] Maíra Gatti, Paulo Cavalin, Samuel Barbosa Neto, Claudio Pinhanez, Cícero dos Santos, Daniel Gribel, and Ana Paula Appel. Large-scale multi-agent-based modeling and simulation of microblogging-based online social network. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 17–33. Springer, 2013.

[20] Jeff Gentry. *twitteR: R Based Twitter Client*, 2015. R package version 1.1.9.

[21] Prasanna Giridhar, Tarek Abdelzaher, Jemin George, and Lance Kaplan. Event localization and visualization in social networks. In *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 35–36. IEEE, 2015.

[22] Gene H Golub, Per Christian Hansen, and Dianne P O'Leary. Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications*, 21(1):185–194, 1999.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[24] Mark W Groch. Radioactive decay. *Radiographics*, 18(5):1247–1256, 1998.

[25] David L Hall and James Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997.

[26] Changhee Han, Hideaki Hayashi, Leonardo Rundo, Ryosuke Araki, Wataru Shimoda, Shinichi Muramatsu, Yujiro Furukawa, Giancarlo Mauri, and Hideki Nakayama. Gan-based synthetic brain mr image generation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 734–738. IEEE, 2018.

[27] David C Harrison, Winston KG Seah, and Ramesh Rayudu. Rare event detection and propagation in wireless sensor networks. *ACM Computing Surveys (CSUR)*, 48(4):1–22, 2016.

[28] Mahmud Hasan, Mehmet A Orgun, and Rolf Schwitter. A survey on real-time event detection from the twitter data stream. *Journal of Information Science*, 44(4):443–463, 2018.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[30] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)*, 47(4):67, 2015.

[31] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *arXiv preprint arXiv:2007.15951*, 2020.

[32] Fang Jin, Edward Dougherty, Parang Saraf, Yang Cao, and Naren Ramakrishnan. Epidemiological modeling of news and rumors on twitter. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, page 8. ACM, 2013.

[33] Jaeyoon Kim, Donghyun Tae, and Junhee Seok. A survey of missing data imputation using generative adversarial networks. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 454–456. IEEE, 2020.

[34] John Krumm and Eric Horvitz. Eyewitness: Identifying local events via space-time signals in twitter feeds. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 20. ACM, 2015.

[35] Mehdi Ben Lazreg, Morten Goodwin, and Ole-Christoffer Granmo. Combining a context aware neural network with a denoising autoencoder for measuring string similarities. *Computer Speech & Language*, page 101028, 2019.

[36] Ryong Lee and Kazutoshi Sumiya. Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In *Proceedings of the 2nd ACM SIGSPATIAL international workshop on location based social networks*, pages 1–10. ACM, 2010.

[37] Zhengchun Liu, Dolores Rexachs, Francisco Epelde, and Emilio Luque. A simulation and optimization based method for calibrating agent-based emergency department models under data scarcity. *Computers & Industrial Engineering*, 103:300–309, 2017.

[38] Zongge Liu, Hyun Ah Song, Vladimir Zadorozhny, Christos Faloutsos, and Nicholas Sidiropoulos. H-fuse: Efficient fusion of aggregated historical data. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 786–794. SIAM, 2017.

[39] Edgar Alonso Lopez-Rojas, Ahmad Elmir, and Stefan Axelsson. Paysim: A financial mobile money simulator for fraud detection. 09 2016.

[40] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, et al. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1596–1607, 2018.

[41] Charles M Macal and Michael J North. Tutorial on agent-based modeling and simulation. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 14–pp. IEEE, 2005.

[42] Brian McNair. *Fake news: Falsehood, fabrication and fantasy in journalism.* Routledge, 2017.

[43] M Zuhair Nashed. *Generalized Inverses and Applications: Proceedings of an Advanced Seminar Sponsored by the Mathematics Research Center, the University of Wisconsin—Madison, October 8-10, 1973.* Number 32. Elsevier, 2014.

[44] Ozer Ozdikis, Halit Oguztuzun, and Pinar Karagoz. Evidential location estimation for events detected in twitter. In *Proceedings of the 7th Workshop on Geographic Information Retrieval*, pages 9–16. ACM, 2013.

[45] Ozer Ozdikis, Halit Oğuztüzün, and Pinar Karagoz. Evidential estimation of event locations in microblogs using the dempster–shafer theory. *Information Processing & Management*, 52(6):1227–1246, 2016.

[46] Ozer Ozdikis, Halit Oğuztüzün, and Pinar Karagoz. A survey on location estimation techniques for events detected in twitter. *Knowledge and Information Systems*, 52(2):291–339, 2017.

[47] Baekkwan Park, Michael Colaresi, and Kevin Greene. Beyond a bag of words: Using pulsar to extract judgments on specific human rights at scale. *Peace Economics, Peace Science and Public Policy*, 24(4), 2018.

[48] Baekkwan Park, Kevin Greene, and Michael Colaresi. Human rights are (increasingly) plural: Learning the changing taxonomy of human rights from large-scale text reveals information effects. *American Political Science Review*, 114(3):888–910, 2020.

[49] Fabio Pezzoni, Jisun An, Andrea Passarella, Jon Crowcroft, and Marco Conti. Why do i retweet it? an information propagation model for microblogs. In *International Conference on Social Informatics*, pages 360–369. Springer, 2013.

[50] Samira Pouyanfar, Yudong Tao, Saad Sadiq, Haiman Tian, Yuexuan Tu, Tianyi Wang, Shu-Ching Chen, and Mei-Ling Shyu. Unconstrained flood event detection using adversarial data augmentation. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 155–159. IEEE, 2019.

[51] Adrianna Rodriguez. Trump administration considering 'pool testing' for coronavirus, fauci says. here's what that means. https://www.usatoday.com/story/news/health/2020/06/26/what-covid-19-pool-testing-faqs-faucis-proposal/3262870001/.

[52] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[53] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[54] Zahra Khodabandeh Shahraki, Afsaneh Fatemi, and Hadi Tabatabaee Malazi. Evidential fine-grained event localization using twitter. *Information Processing & Management*, 56(6):102045, 2019.

[55] Minglai Shao, Jianxin Li, Feng Chen, Hongyi Huang, Shuai Zhang, and Xunxun Chen. An efficient approach to event detection and forecasting in dynamic multivariate social media networks. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1631–1639. International World Wide Web Conferences Steering Committee, 2017.

[56] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[57] Robert H Shumway, David S Stoffer, and David S Stoffer. *Time series analysis and its applications*, volume 3. Springer, 2000.

[58] Kelsey Simpkins. How sampling campus wastewater aims to keep covid-19 in check. https://www.colorado.edu/today/2020/08/27/how-sampling-campus-wastewater-aims-keep-covid-19-check.

[59] Duncan Smith and Sameer Singh. Approaches to multisensor data fusion in target tracking: A survey. *IEEE transactions on knowledge and data engineering*, 18(12):1696–1710, 2006.

[60] Hyun Ah Song, Fan Yang, Zongge Liu, Wilbert van Panhuis, Nicholas Sidiropoulos, Christos Faloutsos, and Vladimir Zadorozhny. Gb-r: A fast and effective gray-box reconstruction of cascade time-series. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pages 494–501. IEEE, 2017.

[61] Avinash Srinivasan and Jie Wu. A survey on secure localization in wireless sensor networks. *Encyclopedia of Wireless and Mobile communications*, page 126, 2007.

[62] Enrico Steiger, Joao Porto De Albuquerque, and Alexander Zipf. An advanced systematic literature review on spatiotemporal analyses of t witter data. *Transactions in GIS*, 19(6):809–834, 2015.

[63] Tycho. Tycho epidimeological dataset. `https://www.tycho.pitt.edu`.

[64] Xiang Wang, Kai Wang, and Shiguo Lian. A survey on face data augmentation for the training of deep neural networks. *Neural Computing and Applications*, pages 1–29, 2020.

[65] Gary M Weiss and Haym Hirsh. Learning to predict extremely rare events. In *AAAI workshop on learning from imbalanced data sets*, pages 64–68. AAAI Press, 2000.

[66] Qingsong Wen, Liang Sun, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*, 2020.

[67] U. Wilensky. Netlogo rumor mill model, 1997. `http://ccl.northwestern.edu/netlogo/models/RumorMill`.

[68] U. Wilensky. Netlogo, 1999. `http://ccl.northwestern.edu/netlogo/`.

[69] Uri Wilensky and William Rand. *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. MIT Press, 2015.

[70] Zack Winn. Real-time data for a better response to disease outbreaks. `https://news.mit.edu/2020/kinsa-health-0821`.

[71] Fei Xiong, Yun Liu, Zhen-jiang Zhang, Jiang Zhu, and Ying Zhang. An information diffusion model based on retweeting mechanism for online social media. *Physics Letters A*, 376(30-31):2103–2108, 2012.

[72] Fan Yang, Hyun Ah Song, Zongge Liu, Christos Faloutsos, Vladimir Zadorozhny, and Nicholas Sidiropoulos. Ares: Automatic disaggregation of historical data. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 65–76. IEEE, 2018.

[73] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

[74] Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. Gain: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*, 2018.

[75] Yi Zhang, Zhe Li, Yongchao Zhang, et al. Validation and calibration of an agent-based model: A surrogate approach. *Discrete Dynamics in Nature and Society*, 2020:1–9, 2020.