Real-Time Event Detection from the Twitter Data Stream

By

Mahmud Hasan

BSc, Islamic University of Technology, Bangladesh
MSc, University of Reading, United Kingdom

A thesis submitted to Macquarie University
for the degree of
Doctor of Philosophy
Department of Computing
May 2018

Typeset in $\LaTeX\,2_\varepsilon$.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Mahmud Hasan

Dedication

Dedicated to the collective human effort for a better world.

Acknowledgements

I would like to thank my principal supervisor Professor Mehmet A. Orgun for his continuous support and guidance at every step of my PhD journey. I would also like to thank my associate supervisor Dr. Rolf Schwitter for being an integral part in guiding my research and providing insightful feedback. I am grateful to have had the opportunity to work with my supervisors. This thesis would not have been possible without them.

I am thankful to be a part of Macquarie University and truly appreciate the support I have been provided with during my postgraduate research.

I would like to thank my parents, Md. Abdur Rab and Nilufar Banu, for their unconditional love and wisdom that made me who I am today. I am eternally in your debt for everything you have done and all the sacrifices you have made to raise your children with an incomprehensible disregard to your own comfort.

Last but not the least, I would like to thank my wife, Tasmia Ekram, for the tremendous support and the unwavering patience she has shown to accommodate my work. Your gracious and loving presence in my life has been a sanctuary for me in difficult times. Thank you for being you and being with me.

List of Publications

- M. Hasan, M. A. Orgun, and R. Schwitter. *Real-time event detection from the Twitter data stream using the TwitterNews+ Framework*. Journal of Information Processing & Management, (2018). Elsevier. `https://doi.org/10.1016/j.ipm.2018.03.001`.
- M. Hasan, M. A. Orgun, and R. Schwitter. *A survey on real-time event detection from the Twitter data stream*. Journal of Information Science, (2017). SAGE Publications. `https://dx.doi.org/10.1177/0165551517698564`.
- M. Hasan, M. A. Orgun, and R. Schwitter. *TwitterNews+: a framework for real time event detection from the Twitter data stream*. Proceedings of the 8th International Conference on Social Informatics (SocInfo '16), Bellevue, WA, USA, pp.224-239, (2016). `https://doi.org/10.1007/978-3-319-47880-7_14`
- M. Hasan, M. A. Orgun, and R. Schwitter. *TwitterNews: real time event detection from the Twitter data stream*. PeerJ PrePrints vol.4, pp.e2297v1, (2016). `https://doi.org/10.7287/peerj.preprints.2297v1`

Abstract

Detecting events in real-time from the Twitter data stream has gained substantial attention in recent years from researchers around the world. We have performed a survey on different event detection systems and identified that one of the major challenges faced, while designing these systems, is the high volume of tweets in the Twitter stream which can incur a computationally prohibitive cost to detect events in real-time. As a solution to the problem, we have designed an end-to-end event detection framework, *TwitterNews+*, which incorporates a novel variant of an incremental clustering approach to provide a low computational cost and a scalable solution to detect newsworthy events in real-time from the Twitter data stream.

We have conducted a parameter sensitivity analysis to fine-tune the parameters used in *TwitterNews+* in order to improve its performance in detecting newsworthy events. We then performed an experimental evaluation of the effectiveness of *TwitterNews+* against five state-of-the-art baselines that cover a wide range of event detection techniques. The results of the evaluation, performed on a publicly available tweet corpus, show that *TwitterNews+* outperforms the baselines by achieving the highest recall and precision in detecting newsworthy events. Our experiments revealed that the *number-of-tweets/second* processing capability of *TwitterNews+* is sufficiently high and thus, allows our system to achieve real-time event detection capability and scalability.

Finally, we have incorporated a novel component in *TwitterNews+*, which can provide a set of context tweets for an event by extracting relevant additional information using the Twitter Search API. A probabilistic-feedback-based approach has been taken to maximize the relevancy of the context tweets associated with an event. The modular nature of the context providing component in *TwitterNews+* allows it to be used by any event detection system to supplement the limited information often contained in an event.

Contents

List of Figures

List of Tables

1

Introduction

The proliferation of social networking platforms has resulted in a rapid increase of their user base, spanning most parts of the world. These online platforms are continuously gaining popularity as effective means of communicating information and allow their users to post public statuses which collectively accumulate into a massive untapped source of dynamic and real-time information on diverse topics. The real-time nature of the content produced through these social networking platforms can provide a timely insight on the current state of affairs.

The microblogging service Twitter has become a focal point of recent research endeavors to investigate different approaches to detect events in real-time, based on the available public statuses provided through the Twitter Streaming API [1, 2]. Twitter allows its users to post and read short text messages, known as tweets. Due to the informal nature of the tweets, and the ease with which they can be posted, Twitter users can be faster in reporting an event than the traditional news media. With around 310 million monthly active Twitter users[1] producing content from all over the world, Twitter essentially has become a host of sensors for events as they happen.

An event, in the context of social media, can be regarded as something of interest that occurs at a specific point in time in the real world and instigates a discussion about associated topics by social media users. Dou et al. [3] defined an event as: "An occurrence causing change in the volume of text data that discusses the associated topic at a specific time. This occurrence is characterized by topic and time, and often associated with entities such as people and location". In the collection on Topic Detection and Tracking (TDT) [4], an event is defined as: "Something that happens at specific time and place along with all necessary conditions and unavoidable consequences". Becker et al. [5] defined an event as a real world occurrence e with 1) an associated time period T_e and 2) a time-ordered stream of Twitter messages M_e of substantial volume, discussing the occurrence and published during time T_e.

According to the aforementioned definitions, most of the event detection systems focus on detecting only major events that instigate a large volume of tweets that discuss the associated topics at specific times. Petrovic et al. [6], who analyzed both major and minor events reported within the duration of two months in the newswire and the Twitter streams, found that major events are equally covered by both the traditional newswire providers and Twitter, whereas Twitter has better coverage on events related to sports, unpredictable high-impact phenomena, and minor or local events which fall outside the radar of the newswire sources. Petrovic et al. also noted [6] that, in some cases, Twitter leads on reporting events related to politics and business.

Osborne and Dredze [7] compared the coverage and latency of breaking news reported on Facebook, Google Plus, and Twitter, based on the major events[2] identified from Wikipedia which occured between the 10^{th} and 31^{st} of December, 2013. Osborne and Dredze [7] found Twitter to be faster in reporting breaking news than other social media, but was still outperformed by the newswires.

---

[1] `https://about.twitter.com/company`

[2] `http://en.wikipedia.org/wiki/December2013`

The findings by Petrovic et al. [6] and Osborne and Dredze [7] confirm the utility of an efficient Twitter-centric event detection system, capable of detecting both major and minor events, tracking event-related updates, and providing meaningful summaries of the detected events. Hence, we have modified the definition of an event provided by Becker et al. [5] which only accounts for major events and define an event in the context of social media as follows:

An event is a real world occurrence e with 1) an associated time period T_e *and 2) a time-ordered stream of messages* M_e *of any volume, discussing the occurrence and published during time* T_e*.*

In addition to the major events that instigate a high level of discussion on Twitter, our definition of an event also considers the real world occurrences, that instigate a low level of discussion, as events. The challenge in doing so, however, is the limited context provided by tweets resulting from the length restriction of 140 characters imposed on a tweet. On top of that, the majority of the information propagated on Twitter is irrelevant for the event detection task, and the noise generated from spammers and the use of an informal language, coupled with spelling and grammatical errors, adversely affect the event detection process. One of the major challenges faced in detecting events from the Twitter data stream in real-time is to minimize the computational cost which is incurred due to the high volume of tweets encountered in a streaming setting.

1.1 A General Event Detection Framework

Based on the literature on event detection systems [1, 2], we observe that a general event detection system framework (Figure 1.1) contains a number of components to deal with the different stages of operations in event detection. A *pre-processing* component receives tweets from a streaming endpoint using a Twitter streaming API and processes them to be used in the subsequent stages by different components. Part-of-Speech (POS) tagging, slang word conversion, and Named Entity Recognition (NER) are all examples of pre-processing performed on tweets. After the pre-processing stage, the *Detecting Events* component employs an event detection technique to detect events from the processed

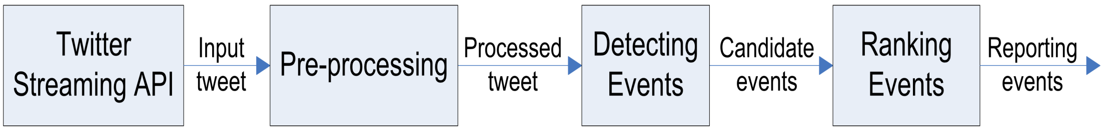


Figure 1.1: A general framework for event detection systems

tweets. The event detection technique employed by this component belongs to any one of the general event detection approaches based on term interestingness, topic modelling, and incremental clustering. There are also miscellaneous event detection techniques found in some studies in the literature which do not directly belong to the above-mentioned three general approaches, but are used in the *Detecting Events* component [1]. The *Detecting Events* component is responsible for grouping the tweets that are related and each group/cluster of tweets corresponds to a candidate event.

Once the candidate events are detected, the *Ranking Events* component ranks them based on some criteria to determine the events that are of interest in the real world. In some cases, a threshold value or a supervised method is used to filter out the trivial events by the *Ranking Events* component. Finally, the newsworthy events detected by an event detection system are rendered by a desktop or web-based application.

Applications of an event detection system include journalism, disaster management, stock market analysis, election polling, etc. As an example, a sample event from the streaming Twitter data is displayed in Table 1.1.

Table 1.1: A sample event: the 2012 Nobel prize winner in literature

Date/Time	Event related tweet
4:50 PM - 11 Oct 2012	Nobel literature jury to announce 2012 winner: After the announcements of three science awards, the Nobel Prizes... http://bit.ly/WUc1EHÃĆ
4:51 PM - 11 Oct 2012	Chinese author Mo Yan wins Nobel Prize for Literature 2012 (first ever Chinese author to win it) ... says he’s “overjoyed & scared” :)
4:51 PM - 11 Oct 2012	Chinese writer ‘Don’t Speak’ Mo Yan has been named the winner of the Nobel Prize in literature http://ti.me/Q0FYhhÃĆ MT @TIME
4:52 PM - 11 Oct 2012	Nobel Prize for literature awarded - Mo Yan of China won the prize for his novel “Frog", which explores the traditio... http://ow.ly/2sCWeyÃĆ

1.2 Main Contributions

In this thesis, we have proposed and implemented a novel end-to-end Twitter-centric event detection framework, *TwitterNews+*, which takes as input a time-ordered stream of tweets and generates clusters of tweets in real-time where each cluster represents a major or minor newsworthy event. The main contributions in this thesis are as follows:

- A survey on the literature on various Twitter-centric event detection methods is performed, where we classify these methods based on the common traits they share (i.e., using probabilistic topic modelling, identifying interesting properties in a tweet's keywords/terms, and using incremental clustering). This categorization will allow the readers to gain a perspective on each of the general research directions that are taken to solve the problem of event detection and an understanding of the finer details that separate one event detection method from the other. The survey is intended to introduce its readers to the noteworthy literature on Twitter-centric event detection and has been published in the Journal of Information Science (2017), Sage Publications. Refer to the list of publications provided in this thesis for further details.

- A novel and efficient solution, to the problem of event detection from the Twitter data stream in real-time, is proposed and implemented as the event detection framework, *TwitterNews+*. There are two major components in *TwitterNews+* for event detection purposes: 1) the Search Module, and 2) the EventCluster Module. The Search Module provides a solution to determine the novelty of an input tweet which aids in the event detection process and the EventCluster Module provides a solution to cluster the tweets that are related to an event. The low computational cost approaches used in these two modules allow *TwitterNews+* to have a real-time processing capability despite the high volume of tweets encountered in the Twitter stream. A part of the work done in *TwitterNews+* has been published in the Lecture Notes in Computer Science (LNCS) book series (2016), Springer.

- We have performed a parameter sensitivity analysis on the different parameters of *TwitterNews+*, which gave us the optimal parameter settings for our system to improve its performance in terms of recall and precision. An experimental evaluation of *TwitterNews+* is performed by utilizing a publicly available corpus along with five state-of-the-art event detection systems which are used as baselines. Our experiments revealed that *TwitterNews+* outperforms the baselines by achieving the highest recall and precision in detecting newsworthy events. An extended version of the work done in *TwitterNews+* has been published in a special issue of the Journal of Information Processing & Management (2018), Elsevier.

- A third major component of *TwitterNews+*, EvenContext Module, which is integrated as part of *TwitterNews+*, is implemented based on a novel approach that provides a context to an event which is detected by any Twitter-centric event detection system. A context is provided by retrieving additional tweets that are related to an event in order to supplement the information contained in the event.

1.3 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 contains a discussion on the survey performed on various event detection methods. In Chapter 3, we discuss our initial proposed system, *TwitterNews*, which combines the Locality Sensitive Hashing [8] scheme with a random-indexing-based term vector model [9] to provide a novel solution to detect events from the Twitter data stream. Based on the experience gathered from developing *TwitterNews*, we were able to design an efficient solution to the problem of event detection which resulted in the implementation of *TwitterNews+*. The various aspects of *TwitterNews+* (i.e., architecture, parameter sensitivity analysis, experimental setup, and evaluation) are discussed in Chapter 4. We then discuss the process of providing a context to an event, which is detected by an event detection system, in Chapter 5. Finally, we conclude the thesis and sketch future research challenges in Chapter 6.

2

Literature Review

2.1 Introduction

An early study on event detection techniques by Atefeh and Khreich [2] mostly focused on the literature before the year 2012, but there has since been rapid and wide-ranging development in this research area. The work done by Li et al. [10], Gaglio et al. [11], Stilo and Veraldi [12], Xie et al. [13], Zhou et al. [14], McMinn and Jose [15], De Boom et al. [16], and Guille and Favre [17] are examples of contemporary developments in this area, employing a wide variety of techniques.

In contrast to our study on the Twitter-centric event detection techniques which is discussed in this chapter, the organization of the content in the study conducted by Atefeh and Khreich [2] is based on three major categories: 1) the type of the event being detected by an event detection system (i.e., specified events, and unspecified events), 2) detection task (i.e., New Event Detection, and Retrospective Event Detection), and 3) event detection methods (i.e., supervised, unsupervised, and hybrid). As the focus of our study is on event detection methods, we believe that a different organization of content is required instead of simply dividing the event detection methods based on whether they are supervised, unsupervised or hybrid.

In this chapter, we classify various event detection methods based on the common traits they share (i.e., using probabilistic topic modelling, identifying interesting properties in a tweet's keywords/terms, and using incremental clustering). This categorization will allow the readers to gain a perspective on each of the general research directions, along with the finer details that separate one event detection method from the other. In the study by Atefeh and Khreich [2], a total of 16 different research articles were discussed in detail. Our focus in this study is to have a wider range of coverage on different event detection methods, as necessitated by the large number of recent studies. We intend to introduce the readers to the recent literature, with a self-contained summary of each of the surveyed works and a comparison of the different proposed methods. This chapter does not contain an exhaustive review of the Twitter-centric event detection literature; however, major representative methods are discussed from various fields, such as information extraction and retrieval, machine learning, data mining, and natural language processing.

The remainder of this chapter is organized as follows: Sections 2.2, 2.3, 2.4, and 2.5 contain discussion of the event detection techniques, categorized into approaches based on term interestingness, topic modelling, incremental clustering, and miscellaneous approaches, respectively. We then describe the different methods employed in the literature for system performance evaluation in Section 2.6. Pre-processing techniques are discussed in Section 2.7. A number of general observations on different event detection approaches are presented in Section 2.8. Finally, Section 2.9 concludes the chapter.

2.2 Term-interestingness-based Approaches

This section discusses the event detection methods which rely on tracking the terms (from the Twitter data stream) that are likely to be related to an event. These methods are summarized in Table 2.1, focusing on the approaches taken to determine term interestingness, clustering techniques used to group the tweets related to an event, and the techniques employed to rank the events that were detected by an event detection system.

The event detection system Twevent [10] initially extracts continuous and non overlapping word segments (single words or phrases) from each tweet. Statistical information obtained from the Microsoft Web N-gram Service[1] and the Wikipedia is used to detect nontrivial word segments. The top-k bursty event segments within a fixed time window are then calculated from the frequency of bursty segments, in conjunction with the user frequency of the bursty segments. Finally, a variant of the Jarvis-Patrick clustering algorithm [18] is used to group related event segments, by exploiting the content of their associated tweets and the frequency pattern of the segments within the specified time window. The events are then filtered based on the newsworthiness score $\mu(e)$, which is calculated using the Wikipedia as a knowledge base for an event e containing a set of event segments $e_s = \{s\}$ in the following manner:

$$\mu(e) = \frac{\sum_{s \in e_s} \mu(s)}{|e_s|} \cdot \frac{\sum_{g \in E_e} sim(g)}{|e_s|} \tag{2.1}$$

where, E_e denotes the set of edges obtained after applying the clustering algorithm and each edge representing the similarity between the event segment nodes they connect; $sim(g)$ is the similarity score of an edge, $g \in E_e$, between the two event segments, calculated based on the associated tweets and their temporal frequency patterns. The newsworthiness score $\mu(s)$ of a segment s is calculated based on the prior probability of a sub-phrase l of segment s, appearing as an anchor text in the Wikipedia articles containing l:

$$\mu(s) = \max_{l \in s} e^{Q(l)} - 1 \tag{2.2}$$

---

[1] `http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx`

Table 2.1: A summary of the term-interestingness-based approaches

Article	Term-interestingness	Clustering	Ranking Event
Li et al. [10] (Twevent)	Top-k bursty word segments are identified from the word segment frequency and the user frequency, within a specific time window.	Jarvis-Patrick clustering algorithm [18].	Events are ranked using newsworthiness score, calculated exploiting Wikipedia.
Marcus et al. [19] (TwitInfo)	User provided keywords are used to identify temporal peaks in tweet frequency utilizing a weighted moving average and variance.	N/A	N/A
Mathioudakis and Koudas [20] (TwitterMonitor)	Detects high frequency terms within a specific time window.	Tweets are grouped based on term co-occurrence, following a greedy strategy.	N/A
Alvanaki et al. [21] (enBlogue)	Shift in correlation values of tag pairs are identified.	Grouping is performed based on co-occurrence of tag pairs in a minimum number of documents.	Events are ranked based on average burstiness of a topic and average number of documents containing all tags of a topic.
Gaglio et al. [11] (TLDF)	A specialized term scoring measure is utilized to retrieve top-K terms from a dynamic temporal window.	A modified SFPM algorithm [22, 23] is used to group related terms in a topic.	N/A

Table 2.1. continued.

Article	Term-interestingness	Clustering	Ranking Event
Cataldi et al. [24]	Emergent terms are identified leveraging the user authority and the usage of the terms in previous time intervals.	Strongly connected components containing emerging terms are identified, from a graph constructed using a term correlation vector.	Average usage of emergent terms in a topic is utilized to rank an event.
Stilo and Veraldi [12] (SAX*)	Temporal series of terms within a specific time window are made discrete using Symbolic Aggregate ApproXimation (SAX) [25] and a regular expression learned from Wikipedia Events is applied to remove non-event terms.	Bottom up hierarchical clustering with complete linkage.	N/A
Parikh and Karlapalem [26] (ET)	Increase in frequency and appearance pattern of terms are used to determine important terms.	Agglomerative hierarchical clustering algorithm.	Number of frequent terms in a cluster is used to rank an event.
Weng and Lee [27] (EDCoW)	Discrete wavelet signals built from individual terms are leveraged to filter out trivial words.	Modularity-based graph partitioning.	Non-trivial events are detected based on the number of words and the cross-correlation among the words related to an event.
Zhang et al. [28]	Bursty words are identified by detecting increase in augmented normalized term frequency [29] and user authority.	A word relation graph is utilized to cluster bursty words.	Event popularity is computed based on a linear spread prediction function.

Events detected by Twevent [10] are highly dependent on the Microsoft Web N-gram Service and the Wikipedia. This dependency can yield a set of events that are influenced by the service itself. Moreover, events that are not yet reported in the Wikipedia might not be detected.

TwitInfo [19] allows a user to input event related keywords to track an event. The system starts logging the tweets that match the user specified keywords, and detects spikes in tweet data as sub-events and automatically labels them with frequently occurring meaningful terms from the tweets. The approach taken by TwitInfo to detect sub-events based on the peak in conversation about a topic, is inspired by the online algorithm used in TCP congestion control to detect unusual delays in packet transmission [30]. Tweets that arrive within a fixed time window are binned, and the frequency of the tweets within the bins is measured to detect the spikes in the tweet frequency with respect to time. Analogous to the approach used by TCP to determine an unusual delay in the transmitted packet being acknowledged, TwitInfo uses a weighted moving average and variance to determine an unusually large number of tweets in a bin. A significant increase in the tweet arrival rate, with respect to the weighted average of the historical mean tweet rate, is registered as a spike by the system. The local maximum of the spike is identified using a hill climbing algorithm to detect the time window for the sub-event. The top ranking *tf-idf* weighted terms are used to provide meaningful labels for the sub-event, where the *tf-idf* value for a term is obtained by multiplying the term-frequency with the inverse document frequency.

TwitInfo allows aggregate sentiment visualization using a Naïve Bayes classifier for sentiment analysis and employs a recall-normalized approach to provide an overview on the sentiment associated with an event. TwitInfo also allows users to drill down through a graphical interface into the sub-events of an event, but it is limited by the fact that, the system is intended to track only the events specified by the users and cannot distinguish between overlapping events.

TwitterMonitor [20] detects emergent topics by utilizing an elementary queuing model to identify the high frequency (bursty) terms from the tweets within a small time window.

The bursty terms detected by the system co-occurring in a large number of tweets are placed in the same group where each group qualifies as a trend. A greedy search strategy is used to generate groupings, in order to avoid the high computational cost to enumerate all possible groups. To provide an accurate description of the detected trends, TwitterMonitor uses context extraction algorithms [31] to find terms that are not necessarily bursty in nature, but correlated to the trend.

enBlogue [21] detects emergent topics from blogs and Twitter data, by computing statistical values for hashtag pairs within a given time window and monitoring unusual shifts in their correlations. The strength of these shifts in hashtag pairs is used to rank emergent topics, and the top-k ranked topics are returned by the system. There are three major stages to the system operation in enBlogue: seed hashtag selection, correlation tracking, and shift detection.

enBlogue reduces the computational cost by considering only the popular hashtags as a seed. The popular hashtags are determined based on the sliding-window average of elements found in the document stream. The extracted named entities are also included as part of the seed. Any combination of the hashtags that contain at least one of the seed hashtags are further considered for the calculation of correlation. The correlation between the hashtag pairs in a given time window are measured based on the average number of documents containing both the hashtags and the similarity between these documents. The hashtag pairs are then monitored to detect a shift in their correlation, and the strength of the shift is measured using exponential smoothing as a forecasting technique. The exponential smoothing equation computes the score for possible emergent topics, by giving higher weights to the most recently observed correlation values and lower weights to the observations in the distant past. Finally, a post-processing phase is employed to group hashtag pairs that refer to the same event. To minimize multiple hashtag pairs referring to the same event, two hashtag pairs that co-exist in at least 80% of tweets are grouped in the same event.

Twitter Live Detection Framework (TLDF) [11] adapts the Soft Frequent Pattern Mining (SFPM) algorithm [22], to detect relevant topics within a generic macro event while addressing the dynamic nature of the Twitter data stream. Unlike enBlogue [21] and TwitterMonitor [20], TLDF uses a sigmoid function dependent dynamic temporal window size, to detect events based on term co-occurrence in real-time. The dynamic temporal window allows TLDF to adapt its event detection behaviour based on the actual volume of tweets related to an event. To reduce the number of terms to be considered, the term selection method in the modified SFPM generates top-K terms, from the set of tweets within the current time window.

Each term is weighted based on a value which is decided depending on the term being recognized as a named entity (i.e., persons, organizations, and locations) by the NER [32], its $tf-idf$ score, and the highest ratio of the likelihood of appearance for a term in the current time widow and the reference corpus of randomly collected tweets. The combination of the likelihood ratio of appearance and the $tf-idf$ score of a term filters out the common terms, and retains the terms which are relevant in the current time window and also in the past topics. Moreover, during the term weighting process, named entities are boosted by a factor of 1.5 to account for their importance in event detection.

The event detection scheme proposed by Cataldi et al. [24] initially creates a term vector representation from the tweets within a given time window. The term weights are represented as augmented normalized term frequency [29] to reduce noise. At the next step, the system determines the authority of a user based on the number of followers of the user and the authority of the followers. The approach taken to determine the user authority is similar to the PageRank algorithm [33]. A life-cycle-based content model is applied within a given time interval to determine the life cycle of each of the detected terms. The life cycle of a term is modelled by calculating the energy of the term. Energy is computed by leveraging the user authority and the usage of the term in previous time intervals. Afterwards, a threshold is dynamically set at each time interval, and the terms having a higher energy value than the threshold are considered as emergent terms for that interval. Then, a correlation vector for each of the terms is created. It represents the semantic association of a term with all the other terms in the corpus. By leveraging this correlation vector, a directed and edge-weighted graph is constructed. This allows topic detection from the graph by identifying strongly connected sub-graphs containing an emergent term.

Stilo and Veraldi [12] use temporal co-occurrence of terms instead of contextual co-occurrence to detect events. The temporal co-occurrence-based approach has been adopted to overcome the limitation of the limited context provided in individual tweets. The authors [12] have utilized the Symbolic Aggregate ApproXimation (SAX) [25] technique to provide a temporal characterization of events within a specific temporal window W, by converting the time series associated with terms into a sequence of symbols. Anomalous behaviors are detected by learning a regular expression, on a subset of the Wikipedia Events[2], to reduce the numbers of terms to be considered in the subsequent clustering phase. A bottom-up hierarchical clustering technique with complete linkage is applied on the remaining terms to group the tokens that share similar string representations within window W. The overall computational complexity of the event detection system is $O(Dt + L + L^{'}W + (L^{'} - 1) + (W^2L^{'}) + K)$, where D is the number of tweets in W, t is the average tweet length, L is the vocabulary dimension in W, $L^{'}$ is the vocabulary dimension after pruning, and K is the number of discovered events.

The event detection system ET, proposed by Parikh and Karlapalem [26], extracts event representative terms based on the increased frequency of terms in consecutive time intervals. A list of intervals are associated with each term in which they are frequent. A combination of a frequency increase and appearance pattern of terms is utilized to filter out terms that are bursty in nature but are not related to any event. ET uses bigrams as candidate terms to reduce the computational cost incurred by a large number of bursty unigrams. Similar terms (bigrams) are clustered using an agglomerative hierarchical clustering technique that does not require the number of clusters to be specified in advance. The clustering algorithm utilizes a similarity matrix that contains the similarity score between each pair of terms and starts with merging the terms with highest similarity score. After a merge operation is applied, the similarity matrix is updated and this process continues until there is not a single pair of terms/clusters with a similarity score above a certain threshold. The overall similarity between two terms is calculated based on the

---

[2] `http://en.wikipedia.org/wiki/2012.`

content similarity $C\_Sim(k_1, k_2)$ and the appearance similarity $A\_Sim(k_1, k_2)$:

$$Sim(k_1, k_2) = \alpha \cdot C\_Sim(k_1, k_2) + \beta \cdot A\_Sim(k_1, k_2) \tag{2.3}$$

where, the content similarity $C\_Sim(k_1, k_2)$ between the terms k_1 and k_2 is calculated using the frequently co-occurring bigrams (FCB):

$$C\_Sim(k_1, k_2) = \frac{FCB(k_1) \cap FCB(k_2)}{FCB(k_1) \cup FCB(k_2)} \tag{2.4}$$

and, the appearance similarity is calculated based on the frequent time intervals (FI) of k_1 and k_2:

$$A\_Sim(k_1, k_2) = \frac{FI(k_1) \cap FI(k_2)}{FI(k_1) \cup FI(k_2)} \tag{2.5}$$

Bursty terms are detected in EDCoW [27] by building a signal for each word using wavelet theory [34]. Modularity-based graph partitioning has been utilized to detect events by calculating the cross correlation among the signals generated from each word. However, the process of wavelet generation for each word and the calculation of correlation is computationally expensive.

The event detection system proposed by Zhang et al. [28] detects bursty terms, where a term burst is identified based on the increase in the term weight within a given time window. Each term is weighted based on a combination of augmented normalized term frequency [29] and user authority. The user authority is incorporated in term weighting with the assumption that the use of a term in a microblog post by an influential user with many followers will boost the use of the term through the followers. The approach taken to estimate the user authority is similar to the PageRank algorithm [33].

Zhang et al. [28] have used a two-state automata based on the Hidden Markov Model to detect bursty terms by proposing an incremental strategy with a complexity of $O(1)$. From the set of bursty terms in a given time window, a term relation directed graph is constructed. Each vertex in this graph is a bursty term and an edge between the terms represents the co-occurrence relation. Subsequently, a graph-based clustering technique is applied to extract the strongly connected components, each of which represents an event. The system proposed by Zhang et al. [28] also predicts the popularity of an event based on the spread model, which relies on determining the user's influence, the user's interest in the event and the tweet volume of the event.

2.3 Topic-modelling-based Approaches

In this section, we discuss the event detection methods which are dependent on the probabilistic topic models to detect real-world events, by identifying the latent topics from the Twitter data stream. The topic-modelling-based approaches for event detection associate each tweet with a probability distribution over various latent topics to find the hidden semantic structures from a collection of tweets to guide the event detection task. These methods rely on sophisticated models to infer latent topics. In the following, we describe the motivations for using more complex modelling to account for the different aspects of topic detection with respect to event detection.

The topic-modelling-based approaches discussed in this section revolve around the complex mathematical models to estimate the probability distribution of the latent topic variables. We have focused our discussion on the different tweet-related attributes that were used to estimate the joint probability distribution along with the other unique aspects of these approaches which distinguish them.

TwiCal [35] populates an open-domain calendar for important events, to provide a structured representation of the significant events extracted from the Twitter data stream. TwiCal extracts named entities [36], along with associated event phrases and dates [37], from each of the streaming tweets. A supervised approach to extract event phrases by using Conditional Random Fields [38] was adopted. A multitude of features, including but not limited to, contextual, dictionary and orthographic features were used to guide the learning and inference of event phrases. Due to the highly volatile nature of topics in the Twitter data, a latent variable model is used to discover the important event types from a large tweet corpus. From the discovered event types, only the coherent types found during inspection were retained and manually annotated with informative labels.

These event types can be used to categorize event phrases extracted from subsequent new data. Events are ranked by measuring the association strength between an entity and a specific date, based on G^2 log likelihood ratio statistic. This approach - identifying events based on the association between an entity and a specific date/time - does not work well for the events that are unexpected or smaller in terms of significance.

Latent Event and Category Model (LECM) [14] uses a latent variable model similar to TwiCal [35], that detects events in a structured manner. In LECM, each tweet is modelled as a joint distribution over the named entities, date/time and location of the event occurrence, and the event related terms. This distribution is expected to ensure that, two events occurring at the same place and time are distinguishable. To categorize events of different types, each named entity is mapped to its related semantic concepts using Freebase API[3]. Afterwards, each tweet is modelled as a joint distribution over the event related terms and the semantic concepts of the named entities. LECM uses the Collapsed Gibbs sampling algorithm to infer the parameters used in the model, and the event types and their semantic class. Unlike TwiCal [35], LECM uses a Bayesian modelling approach that can extract event-related keywords directly from the tweets without requiring supervised learning. Moreover, the LECM model can produce a structured representation of events directly, whereas TwiCal depends on G^2 log likelihood ratio statistic to measure the association strength between an entity and a specific date.

General and Event related Aspects Model (GEAM) [39] is a hierarchical Bayesian model based on Latent Dirichlet Allocation (LDA), similar to LECM [14]. However, each tweet in GEAM is modelled not only over event-related aspects (i.e., time, locations, entities), but also on general topics. Each named entity or hashtag in a tweet is assigned an event-related aspect. Other terms in a tweet are assigned a general topic or an event related aspect based on a switching variable drawn over a binomial distribution. Collapsed Gibbs sampling algorithm is used to estimate the multinomial (i.e., events, general topics, event's aspect) and binomial distributions.

TopicSketch [13, 40] detects bursty events by maintaining a novel data sketch to detect acceleration on three quantities: the whole Twitter stream, every word, and every pair of

---

[3] `http://www.freebase.com/`

words. The system provides a low cost solution to maintain and update this information. The sketch-based topic modelling approach triggers topic inference, when an acceleration on these stream quantities is detected. As this strategy will result in data with dimensions in the order of millions, a hashing-based dimension reduction scheme is utilized to address this issue. As this strategy will result in data with dimensions in the order of millions, a set of most recently encountered active words are kept track of and a hashing-based dimension reduction scheme is utilized to address this issue. A lazy maintenance scheme is employed using H hash functions to limit the number of accelerations to be updated to $0(H.|d|^2)$, where $|d|$ represents the average number of words in a tweet. The hashing scheme can handle the high dimensionality of data in TopicSketch, which originates from the high number of distinct words encountered, by reducing the dimensionality of data to $o(H.B.K)$, where K refers to the number of active latent topics and B refers to the number of buckets where the words are mapped using the H hash functions.

Bursty Event dEtection (BEE+) [41] is an incremental topic model that discovers bursty events by modelling the temporal information of events, and utilizes a distributed execution framework to achieve real-time processing capability. Compared to the documents processed in the standard topic models where a large document has a probability distribution over a mixture of topics, a post in a microblog containing only a few sentences are more likely to be associated with a single topic; therefore, a single event. Unlike most of the other topic models for event detection, in BEE+ a microblog post related to an event is determined by associating only a single hidden variable and an additional background topic, which has a distribution over the common words. Burst detection in BEE+ is similar to the approach used in TwitInfo [19] which was inspired by the TCP congestion control algorithm, and the process to estimate the parameters in BEE+ has a faster convergence time than the traditional topic models. Moreover, the incremental parameter update process in BEE+ is able to keep track of topic drifts over time.

Spatio-Temporal Multimodal TwitterLDA (STM-TwitterLDA) [42] is a topic-model-based framework for event detection that extracts text, image, location, timestamp and hashtag-based Twitter features, from each tweet in the Twitter data stream, as input and jointly models the probability distribution of these features to detect events. Note that all the features except for the text might not always be present in a tweet. STM-TwitterLDA employs a Support-Vector-Machine (SVM) classifier to remove noisy images, a latent filter to remove general images and words, and uses Convolutional Neural Networks (CNN) [43] to extract visual features from images to leverage in event detection. Finally, maximum-weighted bipartite graph matching is applied on the events detected in consecutive periods to track the evolution of the detected events.

Shepard [44] proposed a theoretical framework for retrospective event detection. The proposed system combines the unified model proposed by Diao and Jiang [45] with Nonparametric Pachinko Allocation Method (NPAM) [46], to detect nested hierarchies of topics called super topics and sub-topics. Here, the super topics correspond to long lasting super events and sub-topics are the smaller sub-events contained within the super-topics.

Madani et al. [47] applied a nonparametric Bayesian model called Hierarchical Dirichlet Processes (HDP) [48], to detect trending topics from the Twitter data stream. Initially, a vector of topics is discovered by applying the generative model of the HDP on tweets and for each tweet a distribution of topics is calculated by exploiting the vector of topics. The topic with the highest probability in a distribution of topics is considered as a trending topic. Tweets with similar trending topics are grouped into clusters. The authors have used the Gibbs sampling algorithm to estimate the parameters used in their model. In addition, the system utilizes a thesaurus of terms created using the YAGO (Yet Another Great Ontology[4]), to incorporate semantic information which can aid in the clustering of trending topics.

2.4 Incremental-clustering-based Approaches

As traditional clustering algorithms usually require that the total number of clusters be fixed, it is difficult to predict the total number of expected event clusters in advance for high volume, real-time Twitter data where a wide variety of topics are discussed. The event detection methods discussed in this section follow, at their very core, a clustering strategy

---

[4] `http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/`

that is incremental in nature in order to avoid having a fixed number of clusters. These methods are summarized in Table 2.2, focusing on the approaches taken to determine the term weights to generate a tweet vector, methods applied along with the incremental clustering to group event-related tweets, and the techniques employed to rank the events that were detected by an event detection system.

Petrovic et al. [8] have implemented a First Story Detection (FSD) system based on an adapted variant of the locality-sensitive hashing (LSH) technique. The LSH-based approach provides a fast way to compare the similarity of the input tweet with the previously encountered tweets to determine its novelty while complying with a constant time and space requirement for a streaming environment. A novel tweet represents a new story, which will be assigned to a newly created cluster. On the other hand, a tweet determined as 'not novel' will be assigned to an existing cluster containing the nearest neighbor. The proposed system uses the number of unique user posts and the entropy information in a cluster to rank the event clusters. Besides relying on those aforementioned redundancies, the system lacks the capability to distinguish between significant and trivial events. However, in their later work [49], the authors have used the Wikipedia as an external source to identify the significant events. The approach taken to track unusual spikes in the Wikipedia page views suffers from a latency. That means, real-time event detection is delayed as the Wikipedia stream has a substantial time lag with respect to the Twitter stream.

ReDites [50] is built on the LSH-based FSD system [8], tailored for the security domain to detect events related to security (e.g., violent events, natural disasters, and emergency situations). In order to improve the precision of the LSH-based event detection system [8], a content classifier was built on a passive-aggressive algorithm, and trained on manually labeled events that were automatically discovered by the FSD system. Out of the entire set of detected events, security related events are extracted, using a weakly supervised Bayesian modelling approach based on the Violence Detection Model (VDM) proposed by Cano et al. [51]. The VDM starts with a security-related word lexicon created from DBpedia[5], and subsequently discovers new words related to security events for better

---

[5] `http://wiki.dbpedia.org/`

Table 2.2: A summary of the incremental-clustering-based approaches

Article	Term Weight	Clustering	Ranking Event
Petrovic et al. [8]	$tf-idf$	Incremental clustering based on similarity threshold, exploiting locality sensitive hashing and cluster expiration after a specific time to achieve scalability.	Events are ranked on the number of unique user posts and entropy information in a cluster.
Osborne et al. [50] (ReDites)	$tf-idf$	Incremental clustering utilizing the LSH-based scheme proposed by Petrovic et al. [8].	Events are reported based on a content classifier, then an adapted VDM model [51] is used to classify security-related events.
Becker et al. [5]	$tf-idf$, boosted weight on hashtags.	Incremental clustering based on similarity threshold, with a periodic second pass to handle cluster fragmentation.	Events are ranked using the confidence score assigned by the SVM classifier.
De Boom et al. [16]	$tf-idf$	Incremental clustering approach in [5], augmented with hashtag-level semantics.	A logistic regression classifier [52] is used to filter out trivial events.
Phuvipadawat and Murata [53]	$tf-idf$, boosted weight on proper nouns.	Incremental clustering based on similarity threshold.	Popularity, reliability, and message-freshness factors are exploited to rank events.
McMinn and Jose [15]	$tf-idf$	Incremental clustering based on similarity threshold.	N/A

Table 2.2. continued.

Article	Term Weight	Clustering	Ranking Event
Unankard et al. [54] (LSED)	Augmented normalized term frequency [29] with slang conversion and synonym expansion.	Leader-follower clustering algorithm [55], exploiting content and concept similarity.	Events are ranked on location correlation score, and tweet burst in a cluster.
Kaleel and Abhari [56]	$tf-idf$	Incremental clustering utilizing LSH-based scheme with prefix tree data structure.	N/A
Lee and Chien [57]	Dynamic term weighting scheme based on the arrival rate and the occurrence probability of a term within a sliding window [58].	Modified IncrementalDBSCAN clustering algorithm [59, 60].	Events are ranked by applying an energy function over the topic clusters [61].

classification.

Becker et al. [5] have used an incremental clustering algorithm to detect events from the Twitter stream. For each tweet, its similarity is computed against each of the existing clusters. If the similarity of a tweet is not higher than a specific threshold in any of the existing clusters, a new cluster is created. Otherwise the tweet is assigned to a cluster with the highest similarity. Once the clusters are formulated, a SVM-based classifier is used to distinguish between real world events and non-events. The classifier is trained on temporal, social, topical, and Twitter-centric features. Events are ranked based on the confidence score assigned by the classifier.

De Boom et al. [16] augmented semantic information with individual tweets on the incremental clustering approach adopted by Becker et al. [5]. The authors have used TwitterLDA [62] to assign a semantic topic to each tweet and reported that, instead of using semantic topics for individual tweets, assigning semantic labels based on a coarser hashtag level provides a significant gain in precision and recall over the Becker et al. [5] baseline. However, the hashtag level semantics will work only when event related tweets contain hashtags. A similar observation was pointed out by Mehrotra et al. [63], mentioning the problem with hashtag-based pooling when a lot of tweets do not contain any hashtags, although automatic hashtag labeling can improve this situation to some extent.

An incremental-clustering-based approach similar to Becker et al. [5] is taken by Phuvipadawat and Murata [53]. The authors have used pre-defined search queries, such as, #breakingnews, to fetch a more focused set of tweets from Twitter for the event detection task, and boosted the $tf - idf$ scores on proper nouns to obtain a better grouping of event related tweets. Event groups are ranked based on the popularity, reliability and freshness of the tweets in each group. The popularity is measured by using the total number of retweets in a group, and the reliability is measured based on the total number of followers of all the users within the group. Finally, the group score is adjusted by assigning more weight to recent tweets.

McMinn and Jose [15] used an aggressive filtering that retain a very small percentage of tweets (around 5%) from the Twitter data stream that contain named entities. The filtering is done based on the hypothesis that named entities are the building blocks of events and as a proof of concept, the authors [15] implemented an *entity-based* system that identifies bursty named entities for detection and tracking of events. However, this approach is completely dependent on the accuracy of the underlying Named Entity Recognizer [64] it uses. For each named entity found in the tweets being processed by the system, two lists are maintained: 1) the first one is a list of all the tweets that contain the named entity, and 2) the second one is a list of clusters, each of which consists of tweets having the named entity. Each cluster in the second list contains tweets that are near neighbors and possibly discussing different events or sub-events.

At the initial stage of the *entity-based* system's [15] operation a new tweet is clustered based on the named entities it contains. For each named entity contained in the new tweet: the *entity-based* system retrieves a fixed number of tweets from the first list it maintains for the entity; if the maximum similarity between the new tweet and one of the retrieved tweets is above a certain threshold, then the maximum matching tweet is considered as a near neighbor of the new tweet; the *entity-based* system then searches all the clusters in the second list it maintains for the entity to find the cluster that contains the near neighbor and assigns the new tweet to it; if there is no such cluster that contains the near neighbor, a new one is created, where both the tweet and its near neighbor are assigned; the newly created cluster is then added to the second list and subsequently the new tweet is added to the first list.

The *entity-based* system [15] also maintains information on seven time windows of varying sizes associated with a named entity in order to detect a burst on it. For each time window, moving mean and standard deviation values of the entity's frequency are maintained. Once a tweet has been clustered, the seven time windows associated with each of the named entities contained in the tweet are checked. If the entity's frequency, calculated based on the first list associated with it, exceeds three standard deviations of the mean of any time window, then the named entity is considered to be bursty. The burst detection is based on the Three Sigma rule [65], which states that, with an empirical near-certainty of 99.7%, all values in a normal distribution lie within three standard deviations of the mean. Once a burst is detected on a named entity, an event associated with the bursty entity is created. A cluster of tweets, which is selected from the second list of clusters maintained by the system for the bursty entity, is assigned to an event if the average timestamp of the tweets in the cluster is after the initial burst. An event is considered to be expired when all the entities associated with the event are no longer bursty.

Location Sensitive Emerging Event Detection (LSED) [54] detects emerging hotspot events within a sliding time window, by identifying strong correlations between the user locations and the event locations. LSED clusters similar tweets based on a combination of the content similarity and the concept similarity, using an incremental clustering approach called leader-follower clustering [55]. Tweet clustering is triggered when the timestamp of the incoming tweet is greater than the current sliding temporal window. Terms are weighted based on the augmented normalized term frequency [29], and the WordNet [66] is used for the synonyms expansion of nouns and verbs from the top 20% weighted terms in a tweet to ensure efficient clustering.

The content similarity is calculated based on the cosine similarity measure. In order to reduce duplicate clusters, the top terms are extended with the hypernyms derived from the WordNet to calculate the concept similarity based on the Jaccard similarity measure [67]. Once the clusters are formed by the LSED system, each candidate event cluster e is considered to detect the emerging hotspot events based on a combination of the location correlation score and the tweets burst in a cluster:

$$EmergScore_e = (1 + LocScore_e) \times \frac{N_e}{Mean_{prev} + 2SD_{prev}} \tag{2.6}$$

where, $LocScore_e$ is the score for the correlations between the user locations and the event locations in e, N_e is the number of tweets in e in the current time slot, $Mean_{prev}$ and SD_{prev} are the mean and the standard deviation of the number of tweets in the previous time slots in e, respectively.

Kaleel and Abhari [56] used a locality-sensitive-hashing-based scheme with a prefix tree data structure to discover the event related clusters. Each cluster is labeled with the most frequent terms which represent the cluster centroid. A new cluster is created for a tweet that does not belong to any cluster. Lee and Chien [57] proposed an event detection system that incorporates a dynamic term weighting scheme [58] to handle the concept drifts in a topic and adapts the IncrementalDBSCAN algorithm [59, 60] which utilizes the neighborhood relations among the tweets to detect events.

2.5 Miscellaneous Approaches

This section discusses the event detection methods that adopt hybrid techniques, which do not directly fall under the three categories discussed above.

Chierichetti et al. [68] proposed a linear classifier to detect events, which relies on mathematically formalizing the Twitter users' communication behavior, instead of textual information. The proposed system follows the users' activity around a key event by monitoring the shift in the amount of tweets/retweets produced, and the level of communication between individual users.

Guille and Favre [17] proposed an event detection system called MABED (Mention-Anomaly-Based Event Detection), which detects events by exploiting the statistical values calculated from the textual contents of tweets and the frequency of user interactions through user name mentions. Each event produced by MABED is characterized thorough the time duration of the event, a main event word and its related weighted words, and the magnitude of impact of the event over the Twitter users.

Huang et al. [69] proposed a pattern-mining-based approach to detect events by clustering event representative High Utility Patterns (HUP) from microblog texts. A high utility pattern is determined based on the importance of the terms contained in a pattern. Patterns are generated using the FP-Growth algorithm [70] which provides an efficient and scalable method for mining the complete set of frequent patterns by exploiting an adapted prefix-tree structure. To reduce the computational and the memory cost incurred with a high number of generated patterns, a top-K HUP mining algorithm is proposed that performs HUP selection and pattern generation simultaneously. HUP mining is performed from a set of microblog texts within a fixed time window by maximizing the detected pattern utilities where the overlap-degree between each pair of patterns is always below a certain threshold.

Once the HUPs are detected, an incremental pattern clustering is applied to group related patterns in clusters. As the HUPs contain a little amount of text, additional texts associated with high utility patterns are used while measuring pattern similarity. The clustering process is a combination of kNN classification and modularity-based clustering [71] which allows simultaneous identification of emergent and coherent topics.

Adedoyin-Olowe et al. [72] focused on detecting events from the sports and the politics domain by extracting newsworthy hashtag keywords, using a Transaction-based Rule Change Mining framework built on the Apriori method for association rule mining [73]. Dang et al. [74] proposed a Dynamic-Bayesian-Network-based model [75] that utilizes the networks formed by the retweet chains on a topic and the follower-followee relations among the users, to infer the emergent keywords from the Twitter data stream within a specific time interval. Finally, the emergent topics are detected by applying the DBSCAN algorithm [76] to cluster the emergent keywords based on their co-occurrence relations. Fang et al. [77] proposed a MultiView Topic Detection (MVTD) framework that fuses semantic, hashtag, and temporal relations among tweets to perform a Co-training-based Multiview Clustering (CMC) [78–80] for topic detection. The MVTD framework uses a suffix-tree-based vector space model as a document similarity measure.

Thapen et al. [81] proposed an event detection system to detect disease outbreaks by adapting the existing bio-surveillance EARS algorithm [82] to detect location sensitive spikes in Twitter data. A linear SVM classifier was used to remove tweets that do not discuss disease outbreaks. Robinson et al. [83] applied statistical process control methods, such as dynamic bi-plots [84], Adaptive Exponentially Weighted Moving Averages (AEWMA) and Adaptive Cumulative Sums (ACUSUM) [85], on the tweets posted by the users of Twitter in Australia to identify unusual tweets that correspond to potential disease outbreaks.

Yin et al. [86] proposed an event detection framework for emergency situation awareness that utilizes a probabilistic burst-detection module [87] to filter out the trivial tweets and perform an incremental clustering on the tweets that contain bursty features. A similar specialized event detection technique is proposed by Sakaki et al. [88] that predicts earthquakes and provides early warnings. In their work, each Twitter user is considered as a sensor that propagates tweets as sensory information (i.e., text, time, location). A probabilistic approximation algorithm called particle filter [89] is used to estimate the centre and trajectory of the event location.

2.6 Evaluation Methods

In this section, we discuss the widely employed evaluation methods from a selected number of articles (summarized in Table 2.3). This discussion focuses on the various performance metrics and the information regarding the tweet data sets used in different articles. As there is no standard evaluation method, the articles discussed in this section were selected to cover the measures most commonly employed (with small variations) by researchers, along with a few exceptional ones.

Table 2.3: A summary of the evaluation approaches

Article	Ground Truth	Public Corpus	Evaluation Result			
			Method	Precision	Recall	DERate
Li et al. [10] (Twevent)	No	No	Baseline [27]	0.762	13/21	0.231
			Twevent	0.861	75/101	0.16
Marcus et al. [19] (TwitInfo)	Yes	No	Baseline	x	x	x
			TwitInfo	0.80-0.95	0.80-0.95	x
Gaglio et al. [11] (TLDF)	No	No	Baseline1 [21]	0.604-0.687	x	x
			Baseline2 [20]	0.546-0.753	x	x
			TLDF	0.923	x	x
Stilo and Veraldi [12] (SAX*)	No	No	Baseline	x	x	x
			SAX*	0.79-0.91	x	x
Unankard et al. [54] (LSED)	No	No	Baseline1 [90]	0.870	12/54	x
			Baseline2 [91]	0.913	13/949	x
			Baseline3 [21]	0.864	82/1024	x
			Baseline4 [92]	0.967	90/151	x
			LSED	0.973	90/136	x
De Boom et al. [16]	Yes	No	Baseline [5]	0.649	0.363	x
			Proposed [16]	0.64	0.377	x
McMinn and Jose [15]	Yes	Yes	Baseline1 [93]	0.048	32/506	x
			Baseline2 [8]	0.285	156/506	x
			Proposed [15]	0.636	194/506	x
Zhou et al. [14] (LECM)	Yes	No	Baseline [35]	0.642	x	x
			LECM	0.704	x	x
Li et al. [41] (BEE+)	No	No	Baseline1 [20]	0.50	0.800	x
			Baseline2 [94]	0.50	0.733	x
			Baseline3 [95]	0.60	0.733	x
			BEE+	0.70	0.733	x
Parikh and Karlapalem [26] (ET)	No	No	Baseline	x	x	x
			ET	0.91	21/23	x
Guille and Favre [17] (MABED)	No	No	Baseline1 [96]	0.600	0.456	0.250
			Baseline2 [26]	0.575	0.575	0
			Baseline3 [17]	0.625	0.525	0.160
			MABED	0.775	0.608	0.167

The different data sets collected by the researchers are usually from a small sample (around 1%) of the public data made available by the Twitter Streaming API, which can be filtered by location, keywords, language and so on.

Twevent [10] is evaluated on a collection of 4.3 million tweets published by the

Singapore-based users, collected over a period of one month. To evaluate Twevent, Li et al. [10] defined precision as the fraction of detected events that are related to realistic events, recall as the number of distinct realistic events detected from the data set on a daily basis, Duplicate Event Rate (DERate) as the percentage of events among all the detected realistic events, that are duplicates. The system proposed by Weng and Lee (EDCoW) [27] is used as the baseline for Twevent. Marcus et al. [19] used tweets related to three soccer games and one month of earthquakes to evaluate their system. A human annotator was used to produce the ground truth for evaluation, using game videos from online, web-based game summaries, and US Geological Survey on major earthquakes, to identify major soccer events and earthquakes within a specific time period.

TLDF [11] was evaluated on the data collected from Twitter, during the 64 matches of FIFA World Cup 2014. The precision is defined as the number of distinct events the system is able to detect compared to the actual number of distinct events observed during a session. The evaluation is also performed to determine redundancy, which is defined as the complimentary of the number of distinct events the system is able to detect, compared to the total number of detected events. TwitterMonitor [20] and enBlogue [21] were used as the baselines. Stilo and Veraldi [12] propose an evaluation based on forming a web query using the detected event clusters and their associated dates. A candidate event cluster is considered as an actual event, if a direct match is found using the web query formulated from the candidate event cluster. The evaluation was performed on a 1-year Twitter stream, with system parameters tuned to reduce the total numbers of events to be manually labelled for evaluation.

Unankard et al. [54] evaluated their system based on one week of around 200,000 tweets. No ground truth was used for the evaluation. The recall and precision were reported by manually inspecting the events detected by the system. Location-based Emerging Event Detection (LEED) [92], enBlogue [21], and the systems proposed by Sayyadi et al. [90] and Ozdikis et al. [91] were used as the baselines. De Boom et al. [16] gathered two weeks worth of tweets collected from the Flanders, Belgium-based users. A pool of hashtags gathered from the dataset was used to query Twitter REST API to collect additional tweets. Finally, all the tweets in the dataset were clustered to generate the ground truth, and only the events containing hashtags were considered for the evaluation. The system proposed by Becker et al. [5] was used as the baseline.

McMinn and Jose [15] used the Events2012 corpus for evaluation. The corpus contains 120 million tweets and relevance judgments for 506 events. Crowdsourcing was utilized to evaluate the performance of their system, and the LSH approach of Petrovic et al. [8], and the Cluster Summarization approach of Aggarwal and Subbian [93] have been used as the baselines. Zhou et al. [14] evaluated their work on a data set of 64 million tweets collected in December, 2010, with TwiCal [35] as the baseline. Only the precision of the system was reported. However, the authors used a small manually labeled data set to evaluate the system recall and reported a recall of 25.73%. BEE+ [41] was evaluated on a Weibo (Chinese Microblog) data set along with PLSA [94] and TwitterMonitor [20], and BEE [95] as baselines, and the authors reported the best result achieved by their system with the maximum number of topics (clusters) set to 10. Parikh and Karlapalem [26] used a small data set of 33579 tweets. No ground truth was used for the evaluation. The recall and precision were reported based on a manual inspection of the events detected by the system.

Guille and Favre [17] evaluated their system on an English language corpus containing around 1.4 million tweets, published in November, 2009. TS [96], ET [26], α–MABED (variant of MABED [17]) were used as the baselines for their system. The top 40 events reported by each system were manually inspected to report the precision of the detected events. Recall is reported as the fraction of distinct significant events among all the detected events and DERate [10] is reported as the percentage of events that are duplicates among all the significant events. Xie et al. [13, 40] performed a comparison with Twevent [10] as part of their evaluation. The comparison was performed based on a manual analysis of the events detected by both the systems, in order to identify the differences between their results.

Apart from the common evaluation measures, Weiler et al. [97] have suggested the inclusion of an additional evaluation measure to determine the run-time performance of an event detection system by its number of tweets per second processing capability.

2.7 Pre-processing

The common pre-processing techniques applied on the Twitter data stream are as follows: Part-of-Speech (POS) tagging, Named Entity Recognition (NER), resolving temporal expressions, slang word conversion, tweet filtering based on specific criteria (i.e., discarding retweets and/or non-English language tweets), and removing stop words, URL, and user-name mentions from tweets.

The named entity tagger called T-NER [36] was developed to deal with the noisy data present in a tweet. The authors have reported that, T-NER outperforms the Stanford tagger by a 52% increase in $F1$ score. T-NER was used for named entity tagging in TwiCal [35], also by the GEAM event detection system [39]. McMinn and Jose [15] used the GATE Twitter POS model [64] for POS tagging and NER task. Zhou et al. [14] used a Twitter-trained POS tagger by Gimpel et al. [98] to perform POS tagging and named entity tagging was performed using T-NER [36].

A temporal expression extracted from a tweet in TwiCal [35] is resolved into an unambiguous calendar reference by using a tool, called TempEx [37]. LECM [14] has used SUTime [99] to resolve temporal expressions. You et al. [39] have used the natural language date parser - Natty[6] - to extract time information.

Unankard et al. [54] performed slang conversion based on the Internet slang dictionary[7]. Each term is searched in the dictionary to convert a matching slang into a proper English word. Madani et al. [47] used the Porter algorithm [100] to transform variants of words into a single stem by removing suffixes or prefixes. Different word forms are mapped to their base forms using WordNet [101] for lemmatization.

McMinn and Jose [15] use different filtering approaches as part of their pre-processing. They discard retweets, and all the tweets that do not contain any named entities. Term level filters are also employed to remove the tweets containing terms that are usually associated with spam and noise (e.g., “follow", “watch" etc.). Collectively these filters remove more than 90% of tweets, significantly reducing the amount of data to be processed by the system. You et al. [39] discard tweets that do not contain any named entities or

---

[6] `http://natty.joestelmach.com/`

[7] `http://www.noslang.com`

hashtags. Zhou et al. [14] have experimented with two different tweet filtering techniques. The first one involved creating a word lexicon based on the terms extracted from the newspaper articles that were published around the same time as the collected corpus. The second approach was to build an SVM-based classifier using a multitude of word features and event related features. They have reported that the word-lexicon-based approach achieves a higher precision than the SVM-based approach.

2.8 Discussion

Not all the approaches discussed in this survey may be applicable in the real-world. This is mostly due to the scalability issue faced in a streaming environment, as some event detection methods are not equipped to handle the high volume of Twitter data. Moreover, the use of a clustering approach which requires the total number of clusters to be fixed is often not useful for a streaming data environment, where a wide variety of topics are discussed, making it difficult to predict the total number of expected event clusters in advance.

Term-interestingness-based approaches usually differ on the term selection methods they employ, as well as on the way in which term correlations are computed and changes in the term correlations are tracked. The approaches based on term interestingness can often capture misleading term correlations, while measuring term correlations can be computationally prohibitive in an online setting.

Topic-modelling-based approaches work under the assumption that some latent topics always exist in the tweets that are processed. Tweets are modelled as a mixture of topics, and each topic has a probability distribution over the terms contained in those tweets. Latent Dirichlet Allocation (LDA) [102] has been the most used probabilistic topic model, where the topic distribution is assumed to have a Dirichlet prior. Due to the limit imposed on the length of a tweet, capturing good topics from the limited context is a problem that needs to be addressed. Moreover, the topic-modelling-based approaches usually incur too high of a computational cost to be effective in a streaming setting, and are not quite effective in handling the events that are reported in parallel [103]. Stilo and Veraldi [12] noted that, the LDA-based methods usually can only work in an off-line manner, as the temporal aspect of the events are not often considered.

Incremental-clustering-based approaches are prone to fragmentation, and are usually unable to distinguish between two similar events taking place around the same time. The fragmentation refers to the phenomena where the same event is detected multiple times as a new event. To that end, Becker et al. [5], and McMinn and Jose [15] incorporated a separate component to address the fragmentation issue. Incremental-clustering-based approaches usually employ a similarity threshold value to perform clustering. Caution needs to be taken while empirically setting the threshold for determining the similarity of a tweet with a cluster, to ensure that the threshold value works well with the dynamic nature of the fast changing topics in the Twitter stream.

Lehmann et al. [104] and Yang and Leskovec [105] have shown the existence of a number of different temporal patterns of events besides the event-pattern with a bursty characteristic; therefore, event detection approaches which are solely dependent on identifying the bursty characteristic can fail to detect other event patterns such as minor events which instigate a smaller volume of tweets.

Some event detection methods incorporate specific measures to rank events, in order to address the task of identifying newsworthy events from a set of candidate events. The supervised approaches adopted in De Boom et al. [5, 16, 50] might not be able to account for the situations when there is a concept drift in the Twitter data stream. In addition, training a classifier in an off-line manner, by labeling the Twitter data manually can be very time consuming. There are also unsupervised approaches [8, 24, 27, 28, 53, 54] adopted by the researchers with a varying degree of success, to rank events for the purpose of filtering trivial events out of the system generated candidate event set.

Event ranking depends a good deal on the system's ability to filter out the spam tweets. Determining the credibility of the tweets can also help in the event ranking process. Castillo et al. [106] adopted a supervised learning approach to determine the credibility of a given set of tweets, with a maximum precision and recall achieved close to 80%. The features used in their supervised learning technique were based on the information found on the microblogging social media platform. The most important features to determine tweet credibility were generally: message-based, topic-based, user-based, and propagation-based features. Initially, TwitterMonitor [20] along with a J48 decision tree algorithm implemented in the Weka data mining tool, is used to automatically identify news events from a Twitter data set. Once the news events were identified, supervised classification is used to assess the credibility of the news. Gupta et al. [107] have also used a supervised learning and pseudo-relevance-feedback-based approach to determine the credibility score of tweets. Regression analysis was used to discover the best features for credibility assessment. Tweet credibility detection can be used as part of the pre-processing stage or even at the post-processing stage to detect the credibility of the related tweets of a detected event.

Although most of the proposed approaches are evaluated using tweet data sets collected from varying locations and durations, the unavailability of a publicly available text corpus, along with the ground truth for a system performance evaluation, is a major setback suffered by the event detection techniques discussed in this chapter. To the best of our knowledge, the only publicly available corpus of data with the ground truth for evaluation purposes is the one provided by McMinn et al. [108]. The use of this corpus can help us conduct a fair performance comparison among the different event detection approaches proposed by the researchers.

The study conducted by Atefeh and Khreich [2] has discussed the event detection techniques as unsupervised and supervised event detection approaches. The authors (Atefeh and Khreich [2]) pointed out the benefit of not requiring any labeled data for the unsupervised approaches over the supervised approaches which require a manually labeled data set to train a classifier. However, they suggested that the unsupervised clustering approaches can be further optimized to avoid cluster fragmentation and improve clustering efficiency by incorporating additional indicators such as location proximity besides the temporal aspect of the tweets for clustering. Geotags as a measure of location proximity can be used to determine whether the tweets under consideration belong to the same event. On the other hand, the problem of sparse labeled data used in supervised clustering can be handled by semi-supervised learning [109] where a small set of labeled data can be coupled with a large unlabeled data set to build a classifier and by transfer learning [110] where useful information can be extracted from different but related domains. Atefeh and Khreich [2] also stressed on the need for representative data sets and a common test bed for the performance evaluation of event detection systems.

2.9 Conclusion

We have conducted a survey on the noteworthy and recent event detection techniques that focus on detecting real world events of global and/or local interest from the Twitter data stream, broadly categorizing different approaches based on term interestingness, topic modelling, incremental clustering, and hybrid methods. We have discussed the specific detection methodologies employed by researchers and the measures taken to distinguish between real world events and trivial events of no consequence. This chapter also contains a discussion on the evaluation methods adopted by different researchers and the common Twitter data pre-processing techniques involved in the proposed systems.

Based on our findings from the survey, we observe that an incremental-clustering-based approach can be utilized in our proposed event detection system because of its inherently low computational complexity compared to the most of the state-of-the art approaches. Incremental clustering is also suitable as the use of a clustering approach that requires the total number of clusters to be fixed is often not useful for a streaming data environment where a wide variety of topics are discussed and thus, make it difficult to predict the total number of expected event clusters in advance.

Most of the state-of-the art approaches detect major events as they focus on detecting a substantial burst in the volume of tweets that discuss a particular topic. Although minor events instigate a low volume of tweets being posted to Twitter, they often contain important news. Our proposed system should focus on detecting both major and minor events from the Twitter data stream in order to maximize its usefulness.

It is imperative that our proposed event detection system is able to detect events in real-time and the system is evaluated on a publicly available corpus with a rich collection of a variety of events to facilitate a fair comparison with other systems.

3

TwitterNews

3.1 Introduction

The problem of event detection from the Twitter data stream in an incremental clustering context can be divided into two major stages. The first stage involves detecting a burst in the number of tweets discussing a topic related to an event and the second stage involves clustering the tweets that discuss the same event. The operation of our initial proposed system, *TwitterNews*, is therefore divided into two major stages. After the pre-processing of a tweet, the first stage is responsible for determining the fact whether the current input tweet to the system is discussing a previously encountered topic. If the input tweet is discussing a previously seen topic, then it means a soft tweet burst related to an event has occurred and the output of the first stage declares the input tweet to be "*not unique*".

The operation in the first stage of *TwitterNews* is implemented by combining a random-indexing-based (RI) term vector model [9] with the Locality Sensitive Hashing (LSH) scheme proposed by Petrovic et al. [8], to determine whether a tweet is "*not unique*". This is a novel approach that combines RI with LSH to reduce the time needed to determine the novelty of a tweet and performs a fast text similarity comparison between the current input tweet and the most recent tweets while maintaining a constant time and space.

Subsequently, the second stage, which is implemented using a novel approach by adapting the generic incremental clustering algorithm, deals with generating the candidate event clusters by incrementally clustering the tweets that were determined as bursty ("*not unique*") during the first stage. The second stage also incorporates a defragmentation strategy to deal with the fragmented events that were generated when a particular event is detected multiple times as a new event. To achieve scalability in a true streaming setting, each cluster generated in the second stage has a dynamic expiry time, dependent on the subsequent tweet arrival time in a cluster. Finally a set of filters are applied after the second stage to report the newsworthy events from the candidate event clusters.

The remainder of this chapter is organized as follows: we introduce *TwitterNews* in Section 3.2 and discuss its two major components in Sections 3.3 and 3.4. We discuss the results of our experiments and evaluation of *TwitterNews* in Section 3.5. Finally, we conclude in Section 3.6.

3.2 Architecture of the TwitterNews System

As a continuation of our discussion regarding the two major stages of operation in *TwitterNews*, in this section we will discuss the two major components in our system, each of which deals with the operation of a specific stage (Figure. 3.1). The decision making process on the novelty of an input tweet during the first stage is handled by the Search Module, and generating the candidate event clusters during the second stage is handled by the EventCluster Module.

To facilitate the decision on novelty, the Search Module allows a fast retrieval of the neighboring tweets of the input tweet for text similarity comparison. This is achieved by

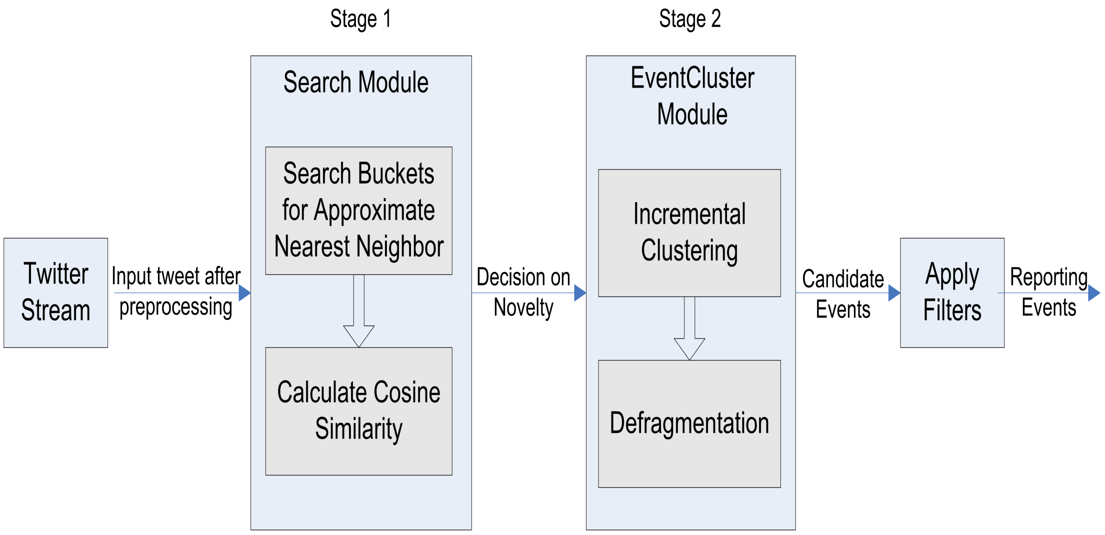


Figure 3.1: Architecture of TwitterNews

using an adapted variant of the Locality Sensitive Hashing (LSH) approach employed by Petrovic et al. [8], where a set of most recent tweets are stored in a fixed number of hash tables. However, Petrovic et al. [8] have used a variable length $tf - idf$ weighted term vector model to calculate the hash keys for each input tweet, which is computationally expensive. The hash keys are used to retrieve the neighboring tweets of the input tweets which are stored in the hash tables.

In order to reduce the computational cost of calculating the hash keys, *TwitterNews* uses a random-indexing-based term vector model [9]. The advantage of using random indexing is the capability to have a fixed length vector generated for each input tweet to the system regardless of the number of tweets encountered. This allows a fast calculation of the hash keys for an input tweet, and thus a fast retrieval of the neighboring tweets from the hash tables.

If the Search Module finds a neighboring tweet of the input tweet with a cosine similarity which is above a specific threshold value, then the input tweet is decided to be "*not unique*", and sent to the EventCluster Module. For each tweet sent to this module, a candidate event cluster to which the tweet can be assigned is searched. If the cosine similarity between a tweet and the centroid of an event cluster is above a certain threshold, then the tweet is assigned to that cluster. When no such cluster is found, a new event cluster is created and the tweet is assigned to the new cluster. The EventCluster Module contains a defragmentation sub-module that helps to merge fragmented event clusters that are sub-events of an event discussing different aspects of the same event. Finally, *TwitterNews* uses a novel scheme based on a word-level Longest Common Subsequence (LCS), along with a set of different filters to discard the trivial events from the candidate event clusters and reports the remaining event clusters as newsworthy events.

3.3 The Search Module

Using the Search Module we aim to detect a soft burst, that is, we intend to find at least one previous tweet that discusses the same topic as the input tweet. To do that we need to store the previously encountered tweets and access them quickly to perform a text similarity comparison with the input tweet.

TwitterNews maintains a fixed number of most recent tweets that are stored and continuously updated in a set of hash tables. Each hash table can be thought of as a collection of buckets, where a hash key maps to a bucket in a hash table and each bucket contains a fixed number of similar tweets. The contents of a bucket are updated regularly by replacing the oldest tweet in the bucket with a new tweet. A fixed number of hash tables are maintained to increase the chance of finding the nearest neighbor of the input tweet. The number of hash keys needed to be calculated for an input tweet is equivalent to the number of hash tables maintained. In our approach, the calculation of the hash keys for the input tweet involves combining a random-indexing-based (RI) term vector model [9] and the Locality Sensitive Hashing (LSH) scheme [8].

Before discussing how and why RI is combined with LSH, we will go through each of these concepts in subsections 3.3.1 and 3.3.2. Then, in subsection 3.3.3, we discuss combining RI with LSH to find previously encountered tweets similar to the input tweet.

3.3.1 Random Indexing (RI)

Random indexing [9] is a term vector based model for semantic similarity. It deals with the high dimensionality issue faced by other term co-occurrence based models by performing random projection for dimensionality reduction. RI works by associating each term with two vectors: an index vector and a context vector. An index vector is a sparse, high dimensional, and unique random vector. In addition, an index vector associated with a term has a fixed dimension. The context vector is initialized with zeroes and its size is equivalent to that of the index vector. The context vector of a term is updated by adding the index vector of another term when it co-occurs with the other term in a sliding context window.

For each tweet in the system, a random-indexing-based incremental term vector can be created, by processing each term of the tweet. Afterwards, a random-indexing-based vector representation for a tweet can be produced from an average of the vectors associated with each term contained in the tweet. Having a random-indexing-based term vector model ensures that, every tweet vector will have the same dimension, regardless of the amount of new terms encountered in any subsequent tweets.

3.3.2 Locality Sensitive Hashing (LSH)

The basic idea behind the LSH-based scheme [8] is that if two high dimensional points are close in some metric space X, a random projection operation on those two points on a randomly drawn hyperplane will allow them to remain close on a low dimensional subspace. The probability of two points i and j colliding in a single hyperplane in this hashing scheme with random projection is:

$$Pr[h(i) = h(j)] = 1 - \frac{\theta(i,j)}{\pi} \tag{3.1}$$

where, $\theta(i,j)$ is the angle between i and j. To simplify, the probability of two points colliding is proportional to the cosine of the angle between them. Increasing the number of hyperplanes k decreases the probability of collision with points that are not close together. Using this concept, similar tweets are hashed into the same bucket of a hash table. Each bucket in a hash table represents the subspace formed by intersecting X with k independently drawn random hyperplanes. The hash function based on which the tweets are placed in a bucket can be defined as:

$$h_u(q) = sgn(u.q) \tag{3.2}$$

where, u is a random vector drawn from a Gaussian distribution $N(0,1)$ and q is a query point, i.e., tweet vector. The output of the previously defined hash function is,

$$h_u(q) = \begin{cases} 1 & \text{if } u.q \geq 0 \\ 0 & \text{if } u.q < 0 \end{cases} \tag{3.3}$$

Each hyperplane $m \in [1...k]$ represents a single bit in m^{th} position of the hash key for a bucket and each bit position of the hash key is calculated by equation (3.3). So, k bits of the hash key for a bucket are formed by gluing all the values of the corresponding bit position together. Although, increasing the number of hyperplanes k decreases the probability of collision with non-similar points, it also decreases the probability of collision with the nearest neighbors. Thus, H number of hash tables are created, each having k independently chosen random hyperplanes, in order to obtain a high probability of collision with the nearest neighbors.

3.3.3 Combining RI with LSH

The calculation of the H hash keys for each input tweet, to retrieve previously encountered similar tweets from the hash tables, is computation intensive even with parallel processing. Generating a single hash key from a $tf-idf$-weighted tweet vector requires k number of independent random vectors whose dimensions are equivalent to the number of unique terms that are encountered by the system. With a $tf-idf$-based scheme, the dimensions of the random vectors are needed to be continually updated as more unique terms are encountered. To alleviate this problem, we use a random-indexing-based term vector model so that each tweet's vector and the k random vectors have a fixed length regardless of the number of unique terms.

Algorithm 1 shows the pseudocode for the Search Module, which combines a random-indexing-based term vector model [9] with the LSH-based scheme [8] to provide a scalable approach to determine the novelty of a tweet. We have used the recommended parameter settings for the LSH-based-scheme [8] and utilized the S-Space Package [111] to create a random-indexing-based vector representation for the tweets. A threshold value t_{sr} is empirically set for the Search Module to determine the novelty of the input tweet using the cosine simialarity measure. If the cosine similarity of the approximate nearest neighbor of the input tweet is below the t_{sr} value, the input tweet is considered as "*unique*", otherwise "*not unique*".

Text similarity between two tweet vectors $t\vec{w}1$ and $t\vec{w}2$ in Algorithm 1 with n dimensions is computed using the cosine similarity measure:

$$\cos(\theta) = \frac{\sum_{i=1}^{n} tw1_i \times tw2_i}{\sqrt{\sum_{i=1}^{n}(tw1_i)^2} \times \sqrt{\sum_{i=1}^{n}(tw2_i)^2}} \tag{3.4}$$

In order to calculate the cosine similarity between the tweets, a $tf-idf$-based vector for each tweet tw is generated by using the following formula:

$$tf-idf(t,tw,D) = tf(t,tw) \times idf(t,D) \tag{3.5}$$

where, t is a term in the input tweet tw, D is the corpus representing the tweets processed so far, $tf(t,tw)$ is the number of times t is found in tw, and

$$idf(t,D) = \log \frac{N}{|\{tw \in D : t \in D\}|} \tag{3.6}$$

where, N is the number of tweets processed so far, and $|\{tw \in D : t \in D\}|$ is the total number of tweets in D in which the term t appears.

Algorithm 1 . TwitterNews: Search Module

```
Require: threshold value $t_{sr}$ for the cosine similarity measure
for each tweet $tw$ in the Twitter data stream do
    generate vector for $tw$ with $RI$
    generate vector for $tw$ with $tf-idf$
    $S \leftarrow$ set of tweets that collide with $tw$ in $H$ hash tables    ▷ hash calculated using a random-indexing-based tweet vector
    for each tweet $tw^{'}$ in $S$ do
        calculate the cosine similarity $cosSim$ between $tw^{'}$ and $tw$ using an available thread from a pool of threads in a multithreaded execution environment
        assign $cosSim$ to the set $S^{'}$
    end for
    $novelInputTweet \leftarrow$ true
    for each $cosSim$ in $S^{'}$ do
        if $cosSim > t_{sr}$ then
            $novelInputTweet \leftarrow$ false
        end if
    end for
    if $novelInputTweet$ then
        $tw$ is "unique"
    else
        $tw$ is "not unique"
    end if
    add $tw$ in each colliding buckets of $H$ hash tables
end for
```

3.4 The EventCluster Module

The EventCluster Module incrementally clusters the tweets discussing the same event and produces a set of candidate events. Algorithm 2 shows the pseudocode for the EventCluster Module, where the event threshold value t_{ev} for a tweet to be assigned to a cluster and the defragmentation granularity value g_{ev} to merge fragmented events are empirically determined.

In order to reduce noise, only the tweets that are decided as "*not unique*" are sent to the EventCluster Module. If a tweet is decided as "*unique*" and sent to the EventCluster Module, there is a chance that it might not have any more similar tweets in the future. Therefore, unnecessarily creating a new event cluster where no new tweets might be assigned and increasing the overall number of active clusters to be searched. Although

Algorithm 2 . TwitterNews: EventCluster Module

```
Require: threshold value $t_{ev}$ for the cosine similarity between a tweet vector and an event
    centroid, and defragmentaion granularity threshold $g_{ev}$ to merge fragmented event
    clusters
for each active event cluster $c$ in $C$ do
    calculate the cosine similarity $cosSim$ between $c$ and $tw$ using an available
        thread from a pool of threads in a multithreaded execution environment
    assign $cosSim$ to the set $C^{'}$
    if $cosSim \geq (t_{ev} + g_{ev})$ then
        assign $c$ to the set of fragmented event clusters $S_c$
    end if
end for
$cosSim_{max} \leftarrow 0$
for each $cosSim$ in $C^{'}$ do
    if $cosSim > cosSim_{max}$ then
        $cosSim_{max} \leftarrow cosSim$
    end if
end for
if $cosSim_{max} > t_{ev}$ then
    assign $tw$ to the cluster $c_{target}$ with the maximum similarity $cosSim_{max}$
    merge the clusters from the set $S_c$ with $c_{target}$
    update $c_{target}$ centroid by averaging with the event centroids in $S_c$ and the
        tweet vector
    update $c_{target}$ expiry time
else
    create a new cluster $c_{new}$ and assign $tw$ to it
    assign the vector of $tw$ as the centroid of $c_{new}$
    assign an initial expiry time to $c_{new}$
end if
```

the novelty testing strategy in the Search Module reduces the total number of clusters being created, there are still a lot of clusters to search in order to decide which cluster a tweet belongs. To mitigate this problem, each cluster has an expiry time associated with it. When a cluster is created an initial expiry time ts_i for the cluster is set. Each time a new tweet is added to the cluster c, the expiry time is updated based on the average time stamp difference between the arrival of successive tweets in c. Once an event cluster is expired, it is marked as inactive to avoid further similarity comparison with any tweet that arrives in the EventCluster Module.

Any incremental algorithm such as ours for the EventCluster Module, suffers from fragmentation. We have employed a defragmentation strategy to avoid cluster fragmentation as much as possible. While searching for a cluster that is closest in similarity to the input tweet, we also keep track of the clusters in a set S_c whose cosine similarity with input tweet is above $t_{ev} + g_{ev}$, as shown in Algorithm 2. After we assign the tweet to the closest matching cluster (given that, similarity is above t_{ev}), all the clusters in S_c are merged to achieve defragmentation.

The time complexity of Algorithm 2 is $O(m)$, where m is the number of clusters. However, from the point in time of system execution where the clusters start getting inactive, the total number of active clusters remain fairly constant.

3.5 Experiment Results and Evaluation

Corpus. In our experiments, we have used the Events2012 corpus provided by McMinn et al. [108]. The corpus contains 120 million tweets from the 9^{th} of October to the 7^{th} of November, 2012. Along with the corpus, 506 ground truth events were provided in a separate file, which we have intended to use to evaluate the results of our experiments. Initially, McMinn et al. [108] have generated a set of candidate events using three different methods. The first two methods are the LSH approach of Petrovic et al. [8], and the Cluster Summarization approach of Aggarwal and Subbian [93]. The third method extracts the events found from the Wikipedia current event portal[1], which are within the dates covered by the corpus. Each event found on the Wikipedia, relevant tweets of that event are retrieved from a Lucene[2] indexed version of the corpus. The candidate events produced by each of these three methods are combined using clustering. McMinn et al. [108] then used crowdsourcing to generate the final set of 506 ground truth events for the time duration covered by the corpus.

Pre-processing. We have performed pre-processing on each tweet before it is sent to the Search Module. Each tweet is tokenized using Twokenize [112]. Tokens containing Username/mentions, stop words, and URLs are removed in the pre-processing phase as

---

[1] `http://en.wikipedia.org/wiki/Portal:Current_events`
[2] `https://lucene.apache.org/`

they do not contribute in clustering the event related tweets. Tokens containing hashtags are retained, as hashtags often contain important information.

Reporting newsworthy events. We have conducted experiment on the first 3 days of approximately 17 million tweets from the Events2012 corpus. The candidate events generated by *TwitterNews* are then filtered by applying a combination of different filters to retain only the newsworthy events which are reported by our system. In *TwitterNews*, a filter based on an entropy threshold is used which ensures that a minimum amount of information is contained in an event cluster and a filter based on a positive user diversity threshold is used which ensures that the event cluster contains tweets from more than one user. Entropy [8] and User Diversity [113] value of an event cluster c, are computed as:

$$Entropy = -\sum_i \frac{n_i}{N} \log \frac{n_i}{N} \tag{3.7}$$

where, n_i is the number of times word i appears in a cluster and N is the total number of words in that cluster.

$$UserDiversity = -\sum_i \frac{u_i}{T} \log \frac{u_i}{T} \tag{3.8}$$

where, u_i is the number of tweets published by user i in a cluster, and T is the total number of tweets in that cluster.

We also employ a word-level Longest Common Subsequence (LCS) based filtering. The idea here is based on the empirical evidence found from inspecting the candidate event set. We have noticed that any news propagated by the general users or the news agencies usually follow a similar sentence structure. We have applied the traditional word-level LCS algorithm on the relevant tweets of an event cluster and identified the tweet with the longest common subsequence of words. Then we use the length of the LCS to determine whether the event cluster is about a newsworthy event. If the longest common subsequence in an event cluster c is below a certain threshold, it means the tweets in c do not have an appropriate level of similarity in their sentence structure and c is not likely to be a newsworthy event.

The LCS-based scheme also selects a representative tweet from the event cluster, by emitting the tweet having the maximum LCS in an event. Before applying the LCS-based scheme on the set of candidate events, all the tweets of each event cluster are discarded that do not contain at least one proper noun or possessive noun. Doing so reduces the total number of tweets in a cluster by discarding the tweets that do not contain any useful information.

Evaluation. Due to the restriction imposed by Twitter, the Events2012 corpus only contains unique tweet IDs using which the tweets belonging to the corpus need to be downloaded. After downloading the tweets, we have inspected the corpus and discovered that a large number of tweets (around 30%) belonging to the corpus were not downloaded as they are not available any more. The effect of a partially incomplete corpus, due to the unavailability of the tweets, is going to negatively impact the results produced by our system. To remedy this problem we have decided to manually reconfirm the ground truth events provided by McMinn et al. [108]. However, there are a total of 506 ground truth events spanning from the 9^{th} of October to the 7^{th} of November, 2012. As this can take a substantially long time, we have only reconfirmed the first three days of the ground truth events and manually selected a total of 41 events that belong to our selected time window. Further inspection of these 41 events were required to identify and remove the events which contained a large number of unavailable tweets. Doing so led us to a final set of 31 events to be used as the ground truth, as shown in Table 3.1.

We have used the First Story Detection (FSD) system [8] as a baseline, which utilizes locality sensitive hashing to facilitate real-time event detection, to compare against *TwitterNews*, which combines a random-indexing-based term vector model and locality sensitive hashing to facilitate real-time event detection. The FSD system achieved a recall of 0.52 by identifying 16 events out of the 31 ground truth events. McMinn and Jose [15] have used the same baseline over the Events2012 corpus and reported the FSD system to achieve a very low precision. As calculating the precision is a time consuming task we have skipped this for the FSD system which has been reported to have a low precision on the same corpus.

Table 3.1: The ground truth events for the Events2012 corpus, between the 9^{th} and the 11^{th} of Oct, 2012

Event Topic	Event Description
BET award	They are discussing a televised award show for the BET network.
Keyshia Cole	It is about a TV show by Keyshia Cole and her husband.
Meek Millz	They all like to watch Meek Millz show.
Fat Joe	They all discuss about fat Joe.
Kendrick Lamar	Best lyricist of the year awarded to Kendrick Lamar.
Omarion	About Omarion dancing on the stage.
Pope Benedict	Pope Benedict XVI adds Arabic to weekly Vatican address in front of the pilgrims.
Jerry Sandusky	Penn State scandal involving imprisoned former football coach Jerry Sandusky.
HP and Lenovo	HP and Lenovo battle for top spot in the PC market of Computerworld.
Nobel prize in physics	Serge Haroche and David Wineland win the 2012 Nobel Prize in Physics.
Moscow court	A court in Moscow, Russia, frees one of the three Pussy Riot members at an appeal hearing.
Los Zetas	Heriberto Lazcano Lazcano, the top leader of the criminal organization Los Zetas, was killed.
BAE and EADS	BAE and EADS announce their merger talks are cancelled over political disagreements.
Nobel prize in chemistry	Two American scientists, Robert Lefkowitz and Brian Kobilka, win the 2012 Nobel Prize in Chemistry.
Lance Armstrong	The USADA details witness-based doping claims against Lance Armstrong in its long-due report to the UCI.
Malala Yousafzai	Malala Yousafzai, a 14 year old activist for women education rights is shot by Taliban gunmen in the Swat Valley.
Nobel prize in literature	Chinese author Mo Yan, famous for working in the style of writing known as hallucinatory realism, wins the Nobel Prize in Literature.
VP debate	Vice presidential debate between Joe Biden and Paul Ryan.
Jayson Werth	Jayson Werth hitting a walkoff home run for the Nationals during the playoff game against the Cardinals.
Cleveland bus	A Cleveland bus driver punched a female passenger in the face.

Table 3.1. continued.

Event Topic	Event Description
Buster Posey	Buster Posey grand slam leads SF Giants to historic Division Series.
Ryan Bertrand	Ryan Bertrand has had to pull out of the England Squad with a sore throat.
Syrian plane	A Syrian passenger plane is forced by Turkish fighter jets to land in Ankara due to the allegations of carrying weapons.
Space shuttle	Space shuttle Endeavour makes a final trip to a Los Angeles museum.
Shell	Oil giant Shell is sued by Niger Delta farmers in a civil court in the Hague.
Clovelly	Heavy rain in the United Kingdom causes flash flooding in the coastal village of Clovelly.
Marie Stopes	The Marie Stopes organization is to open the first private clinic to offer abortions to women in Northern Ireland from 18 October.
Google Android	A U.S. appeals court has overturned a district court order that had banned the sale of Samsung's Galaxy Nexus in the US, delivering a winning round for Google's Android against Apple Inc.
Pep rally	The topic is about a Pep rally.
Qassem M. Aqlan	A gunman kills Qassem M. Aqlan, the Yemeni chief of security employed at the U.S. embassy in the capital, Sana'a.
Syrian conflict	Syrian rebels claiming control of a strategic town.

TwitterNews achieved a recall of 0.87 by identifying 27 events out of the 31 ground truth events. A total of 1619 events were reported by *TwitterNews* within the time window of three days. McMinn and Jose [15] noted that a lot of events can be detected from the Events2012 corpus in addition to the set of 506 events provided as the ground truth. Hence, instead of calculating the precision with respect to the ground truth, we have asked two human annotators to determine the precision of 100 randomly chosen events out of the reported 1619 events. The precision is calculated as a fraction of the 100 randomly chosen events that are related to realistic events. A total of 72 events out of the 100 randomly chosen events were agreed as newsworthy events by both annotators. The results of our evaluation is shown in Table 3.2. Further details on our experimental setup

Table 3.2: A summary of the evaluation results

Method	Recall	Precision
FSD [8]	0.52	-
TwitterNews	0.87	0.72

and evaluation process is discussed in Section 4.4.

3.6 Conclusion

TwitterNews incorporates a random-indexing-based term vector model with the locality sensitive hashing scheme to make a fast decision on the novelty of an input tweet. The incremental-clustering-based approach adopted in our system, along with the defragmentation sub-module, provides an efficient way to cluster the event-tweets. *TwitterNews* performs reasonably well in detecting newsworthy events. However, our experiments revealed that *TwitterNews* does not scale well in a streaming setting. Based on the experience acquired from developing *TwitterNews*, we have designed a low computational cost event detection system, *TwitterNews+*, which is discussed in Chapter 4. *TwitterNews+* addresses the scalability issue faced in *TwitterNews* by implementing more efficient algorithms for the Search Module and the Event Cluster Module, and also improves its performance in detecting newsworthy events.

4

TwitterNews+

4.1 Introduction

In this chapter, we discuss our proposed system, *TwitterNews+* [114]. The architecture of *TwitterNews+*, described in Section 4.2, incorporates a variant of the incremental clustering approach to provide a low computational cost and a scalable solution to the problem of event detection in real-time from the Twitter data stream and operates in two major stages. The first stage, handled by the Search Module, discussed in Subsection 4.2.1, allows the detection of both major and minor events by detecting a soft burst in the number of tweets that discuss an event and the second stage, handled by the EventCluster Module, discussed in Subsection 4.2.2, involves clustering the tweets that discuss the same events and tracking these events. The pre-processing and the post-processing techniques used in *TwitterNews+* are discussed in Section 4.3.

In this chapter, we also perform a parameter sensitivity analysis on the different parameters to determine the optimal parameter settings for our system, discussed in Section 4.5, for which we use a one-at-a-time sensitivity measure [115–117], also known as 'local' sensitivity analysis, as one parameter at a time is repeatedly varied while keeping the others fixed. Note that this approach only deals with sensitivity relative to the chosen parameter and does not look at the entire parameter distribution. A sensitivity ranking for each parameter is obtained by varying its value while leaving all the other parameters constant, and quantifying the change in terms of recall and precision, calculated from the newsworthy events reported by *TwitterNews+*. The parameter sensitivity analysis gave us the optimal parameter settings for our system to improve its performance in terms of recall and precision.

Subsequently, this chapter investigates the performance of *TwitterNews+* and five state-of-the-art baselines that cover a wide range of event detection techniques in Section 4.6 based on the performance evaluation process outlined in Section 4.4. Our experiments revealed that *TwitterNews+* outperforms the baselines and at the same time achieves real-time processing capability. Most of the selected baselines are publicly available. We perform our evaluation using a publicly available corpus, Events2012 [108], which makes the results of our experiments reasonably reproducible.

4.2 Architecture of the TwitterNews+ System

The two main components of *TwitterNews+* are the Search Module and the EventCluster Module (Figure 4.1). The Search Module handles the operation of the first stage in our system and facilitates a fast retrieval of similar tweets from the set of the most recent tweets maintained by *TwitterNews+* to provide a binary decision on the novelty of an input tweet. An input tweet decided as "*not unique*" by the Search Module assures that similar tweets have been encountered before. Our system uses this information to confirm the fact that either an event related tweet burst has occurred (soft burst) or the input tweet is part of an ongoing burst and needs to be tracked.

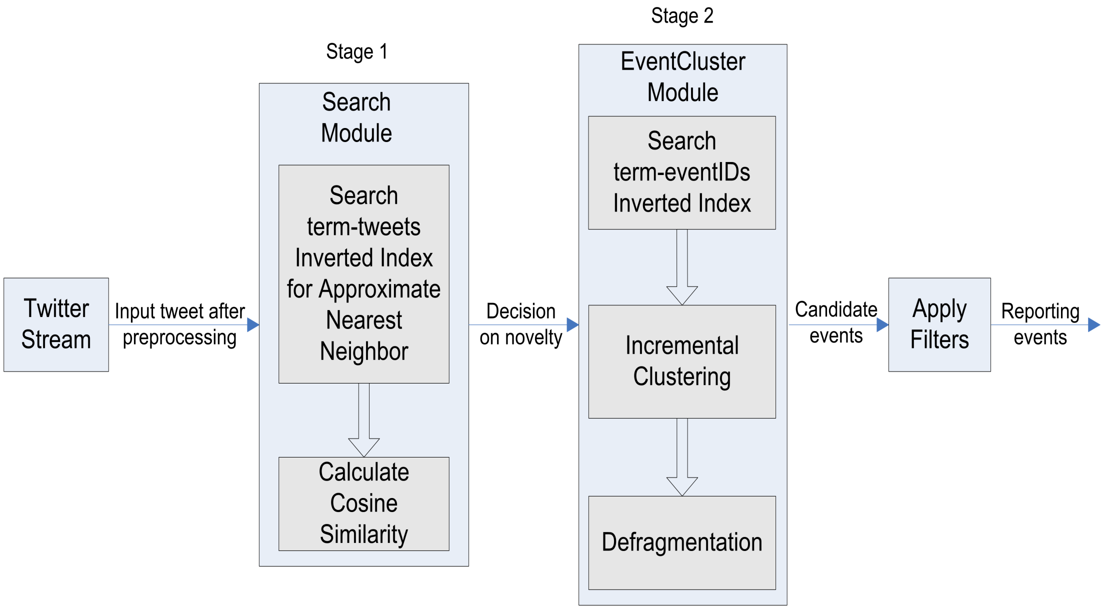


Figure 4.1: Architecture of TwitterNews+

A tweet decided as "*not unique*" by the Search Module is sent to the EventCluster Module which handles the operation of the second stage in our system. For every tweet sent to this module, a candidate event cluster to which the tweet can be assigned is searched. A tweet is assigned to an event cluster if the cosine similarity between the $tf-idf$ weighted tweet vector and the centroid of the event cluster is above a certain threshold. When no such cluster is found, a new event cluster is created and the tweet is assigned to the new cluster. The EventCluster Module contains a defragmentation sub-module that merges together fragmented event clusters. The defragmentation sub-module is also helpful to merge clusters that are sub-events of an event. Finally, *TwitterNews+* uses a novel scheme based on a word-level Longest Common Subsequence (LCS) approach, along with a set of different filters to retain newsworthy events from the candidate event clusters and identifies a representative tweet for each event.

4.2.1 The Search Module

Unlike most state-of-the art approaches for event detection which only focuses on detecting an event with a bursty characteristic (i.e., major events), *TwitterNews+*, also aims to detect events that are non-bursty in nature (i.e., minor events). Our system uses a soft burst

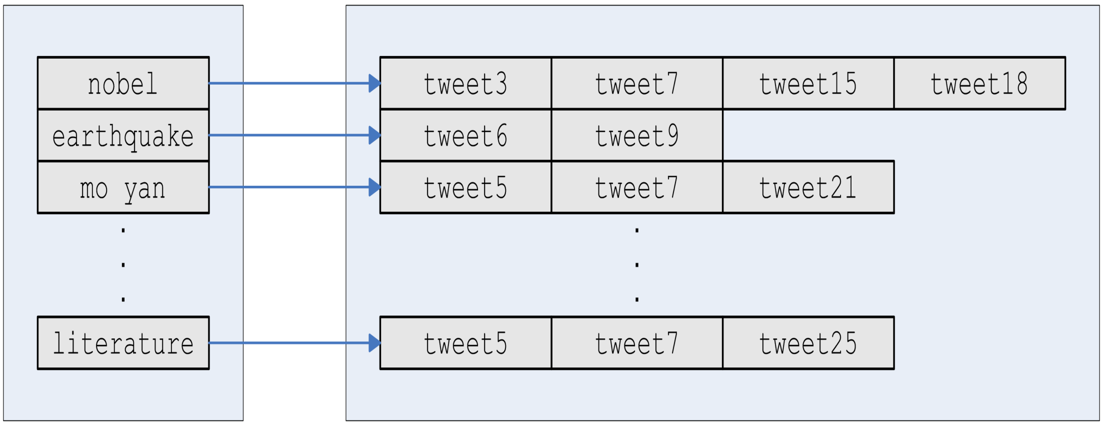


Figure 4.2: The "term-tweets" inverted index

detection approach that enables it to detect both minor and major events. The detection of a soft burst involves simply determining the novelty of the input tweet, which is achieved by storing a continuously updated but fixed number of the most recent tweets in an inverted index and performing a text similarity calculation on them with the input tweet. If for an input tweet a textually similar tweet can be found, then it means the input tweet is "*not unique*" and a soft tweet burst for a particular event has occurred.

To reduce the time needed to search for previously encountered tweets similar to the input tweet tw, while maintaining a constant time and space requirement, we utilize a *term-tweets* inverted index (Figure 4.2) maintained by *TwitterNews+* on a finite set, M, of the most recent tweets. The set M is continuously updated by replacing the oldest tweet with the latest input tweet to keep the memory requirement constant for the *term-tweets* inverted index as the number of unique terms can grow very large due to the unconstrained use of vocabulary in the streaming tweets. Each entry of the *term-tweets* inverted index contains a term and a finite set, Q, of the most recent tweets in which the term appeared. The oldest tweet is replaced with the latest tweet containing the term when the number of tweets exceeds the limit of Q. To find an approximate nearest neighbor of tw, the tweet is first tokenized and part-of-speech tagged [112] as part of the pre-processing stage and an incremental $tf - idf$-based term vector is generated.

Subsequently, the top-k $tf - idf$ weighted terms are selected from tw, and for each of the K terms the *term-tweets* inverted index is searched to retrieve a maximum of $K \times Q$

tweets in which at least one of the K terms appeared. To elaborate this idea with an example, let us consider the input tweet "*Mo Yan wins Nobel in Literature*", where the top three $tf-idf$ weighted terms are "*mo yan*", "*nobel*", and "*literature*". Note that the term "*mo yan*" is shown in this example as a compound noun for simplicity and a similar result can be achieved when the individual parts of the compound noun are used in the index. Each of the terms is searched in the *term-tweets* inverted index (Figure 4.2) and the tweets with IDs 3, 5, 7, 15, 18, 21, and 25 are retrieved. Finally, the approximate nearest neighbor of the input tweet among the retrieved tweets is calculated using the cosine similarity measure. The cosine similarity between two tweet vectors is computed using the Euclidean dot product formula. A threshold value t_{sr} for the cosine similarity is empirically set for the Search Module to determine the novelty of the input tweet. A detailed analysis on the optimal values for the various parameters used in the system is provided in Section 4.5. If the cosine similarity of the approximate nearest neighbor of the input tweet is above the t_{sr} value, then the input tweet is considered to be "*not unique*", thus confirming the occurrence of a soft burst.

The most expensive operation in the Search Module (see Algorithm 3) is determining the approximate nearest neighbor based on the cosine similarity measure. However, using the *term-tweets* inverted index restricts the total number of tweets to compare with the input tweet within $K \times Q$. As the $K \times Q$ number of comparisons are not dependent on each other, we have utilized multi-threading for parallel processing of these similarity comparisons which effectively renders the computational cost to $O(1)$ depending on the number of cores available on a machine.

4.2.2 The EventCluster Module

The Search Module sends the tweets that are decided as "*not unique*" to the EventCluster module. Upon receiving a tweet tw, the EventCluster module utilizes a *term-eventIDs* inverted index, in a manner similar to the Search module, to provide a low computational cost solution to find an event cluster in which tw can be assigned (Figure 4.3). Each entry

Algorithm 3 . TwitterNews+: Search Module

```
Require: threshold value $t_{sr}$ for the cosine similarity measure
for each tweet $tw$ in the Twitter data stream do
    select top-k $tf-idf$ weighted terms of $tw$
    $S \leftarrow$ set of tweets retrieved from the term-tweets inverted index containing
    any top-k terms of $tw$                                  ▷ $|S| \leq K \times Q$
    for each tweet $tw^{'}$ in $S$ do
        calculate the cosine similarity $cosSim$ between $tw^{'}$ and $tw$ using an available
        thread from a pool of threads in a multithreaded execution environment
        assign $cosSim$ to the set $S^{'}$
    end for
    $novelInputTweet \leftarrow$ true
    for each $cosSim$ in $S^{'}$ do
        if $cosSim > t_{sr}$ then
            $novelInputTweet \leftarrow$ false
        end if
    end for
    if $novelInputTweet$ then
        $tw$ is "unique"
    else
        $tw$ is "not unique"
    end if
    update the term-tweets inverted index
end for
```

in the *term-eventIDs* inverted index contains a term and a finite set, Q, of IDs of the most recent event clusters in which the term appeared. The oldest event ID is replaced with the latest event ID containing the term when the number of stored event IDs exceeds the limit of Q. For each of the top-k terms in tw the *term-eventIDs* inverted index is searched to retrieve the IDs of the event clusters in which any of the K terms appeared. The total number of retrieved event cluster IDs for K terms does not exceed $K \times Q$ as the maximum capacity of each entry in the *term-eventIDs* inverted index is Q.

To elaborate this idea with the same example used in the Search Module, let us consider that the input tweet "*Mo Yan wins Nobel in Literature*" is decided as "*not unique*" and sent to the EventCluster Module. The top three $tf-idf$ weighted terms of the tweet are "*mo yan*", "*nobel*", and "*literature*". Each of the terms is searched in the *term-eventIDs* inverted index and the event clusters with IDs 1, 2, 4, 7, 8, and 15 are retrieved (Figure 4.3). Note that the total number of event clusters with which the input tweet is compared will always

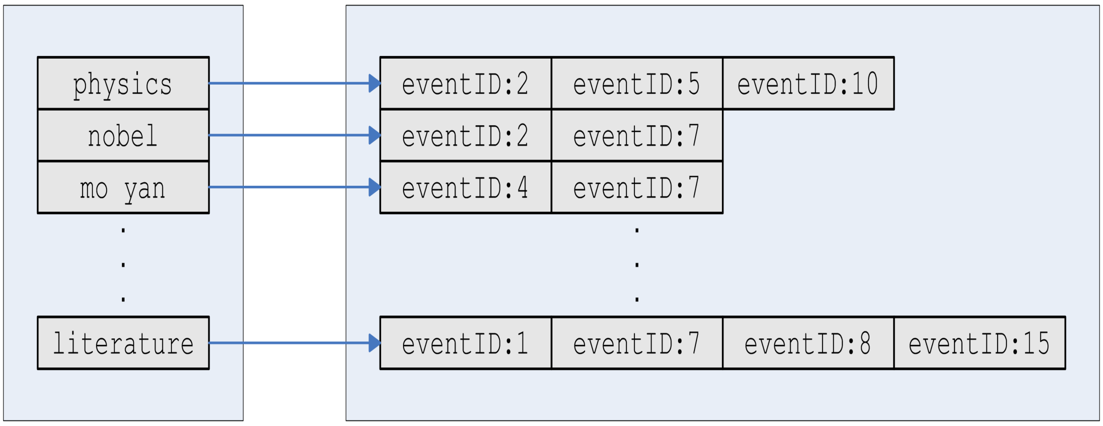


Figure 4.3: The "term-eventIDs" inverted index

be within $K \times Q$. Finally, the input tweet vector is compared with the centroid of each of the retrieved event clusters and assigned to the cluster with the highest cosine similarity. If the cosine similarity is below a certain threshold, t_{ev}, a new cluster is created and the tweet is added to the newly created cluster.

Each event cluster created by the EventCluster Module has an expiry time associated with it. When a cluster c is created, an initial expiry time ts_i for the cluster is set. Each time when a new tweet is added to c, the expiry time is updated based on the average timestamp difference between the arrival of successive tweets in c. Once an event cluster has expired, it is marked as inactive to avoid a similarity comparison with any subsequent tweet that arrives in the EventCluster Module. The *term-eventIDs* inverted index is updated after a fixed interval to remove inactive events in order to maintain a fixed space requirement.

Similar to the Search Module, the most expensive operation in the EventCluster Module (see Algorithm 4) is finding an event cluster in which a tweet can be placed. As the *term-eventIDs* inverted index restricts the total number of event clusters to search for within $K \times Q$, the time complexity of the aforementioned operation becomes $O(1)$ with parallel processing. Any incremental algorithm, such as ours for the EventCluster Module, suffers from fragmentation when a particular event is detected multiple times as a new event, creating multiple event clusters for the same event. We have incorporated a defragmentation strategy to avoid cluster fragmentation as much as possible. The defragmentation strategy is also helpful to merge clusters that contain sub-events of

Algorithm 4 . TwitterNews+: EventCluster Module

```
Require: threshold value $t_{ev}$ for the cosine similarity between a tweet vector and an event
    centroid, and defragmentaion granularity threshold $g_{ev}$ to merge fragmented event
    clusters
$C \leftarrow$ set of events retrieved from the term-eventIDs inverted index containing any of
    the $top-k$ terms of the input tweet $tw$        ▷ $|C| \leq K \times Q$
for each active event cluster $c$ in $C$ do
    calculate the cosine similarity $cosSim$ between $c$ and $tw$ using an available
        thread from a pool of threads in a multithreaded execution environment
    assign $cosSim$ to the set $C^{'}$
    if $cosSim \geq (t_{ev} + g_{ev})$ then
        assign $c$ to the set of fragmented event clusters $S_c$
    end if
end for
$cosSim_{max} \leftarrow 0$
for each $cosSim$ in $C^{'}$ do
    if $cosSim > cosSim_{max}$ then
        $cosSim_{max} \leftarrow cosSim$
    end if
end for
if $cosSim_{max} > t_{ev}$ then
    assign $tw$ to the cluster $c_{target}$ with the maximum similarity $cosSim_{max}$
    merge the clusters from the set $S_c$ with $c_{target}$
    update $c_{target}$ centroid by averaging with the event centroids in $S_c$ and the
        tweet vector
    update $c_{target}$ expiry time
else
    create a new cluster $c_{new}$ and assign $tw$ to it
    assign the vector of $tw$ as the centroid of $c_{new}$
    assign an initial expiry time to $c_{new}$
end if
update the term-eventIDs inverted index
```

an event resulting from the topic drifts. While searching for a cluster that is closest in similarity to the input tweet, we also keep track of the clusters in a set S_c whose cosine similarity with the input tweet is greater than $t_{ev} + g_{ev}$. The defragmentation granularity (g_{ev}) is a threshold value used in the EventCluster Module to merge fragmented events. After we assign the tweet to the closest matching cluster (given that the similarity is greater than t_{ev}), all the clusters in S_c are merged to achieve defragmentation.

From the set of candidate events formed by the EventCluster Module, newsworthy events are reported if they satisfy a few criteria in the post-processing stage as described in Section 4.3.

4.3 Pre-processing and Post-processing Operations in TwitterNews+

In this section, we discuss the main pre-processing and post-processing operations involved in *TwitterNews+*.

Pre-processing. Twitter streaming data often contain information irrelevant for the event detection task. A good amount of tweets contain spams, which unnecessarily slow down the processing time of an event detection system and have a detrimental effect on the precision. To improve on the precision and to reduce the number of tweets to be processed by *TwitterNews+*, a term/phrase level filter has been applied using a manually curated list of around 350 spam phrases (e.g., “click here”, “free access”). Tweets containing these spam phrases are discarded in the pre-processing stage of the system. The spam phrase filter contributes to around 70% of tweets being discarded by *TwitterNews+*.

On the remaining tweets that got through the spam phrase filter, an incremental $tf-idf$-based term vector is generated for each tweet before passing it to the Search Module. Each tweet is tokenized using Twokenize [112]. Tokens containing username/mentions, stop words and URLs are not considered while generating a term vector for a tweet, as they do not contribute in clustering the event related tweets. However, the URLs contained in the tweets play an important role in the post-processing stage which is discussed below.

Post-processing. A combination of three different level of filters as shown in Table 4.1 is applied to discard the trivial events while retaining the newsworthy events from the candidate event clusters generated by *TwitterNews+* (Figure 4.4).

The *first-level* filters use the entropy [8] and the user diversity [113] information in a candidate event cluster and retain the clusters with an entropy and user diversity value above certain threshold values. The entropy threshold (t_{ent}) ensures that a minimum amount of information is contained in a cluster and acts as a filter to remove the clusters

Table 4.1: The filters used in the post-processing stage of TwitterNews+

Filters	Description
first-level	A combination of entropy and user diversity information is used.
second-level	A combination of features such as, number of tweets in a cluster, presence of authentic news portal URLs in a cluster with low number of tweets, and time span between the first and the last tweet in a cluster, are utilized.
third-level	A word-level LCS-based scheme is adopted.

containing insufficient information. On the other hand, a positive user diversity value ensures that a cluster contains tweets from more than one user and acts as a filter to remove the clusters containing tweets from a single spammer.

As *TwitterNews+* is designed to detect a soft burst on a topic in order to detect both minor and major events, it will generate event clusters ranging in size from very small to large. Experimental analysis revealed that there are a significantly high number of clusters that contain a very small number of tweets and/or span a short period of time. These clusters are mostly about mundane topics and will reduce the precision of the detected events if not filtered out. However, some of these clusters contain important newsworthy events and only for those small clusters we have decided that having a URL from a news portal will definitively indicate their newsworthiness. The *second-level* filters in *TwitterNews+* are employed to discard the candidate event clusters that have less than ten tweets and do not contain a URL of a news portal from a collection of top online

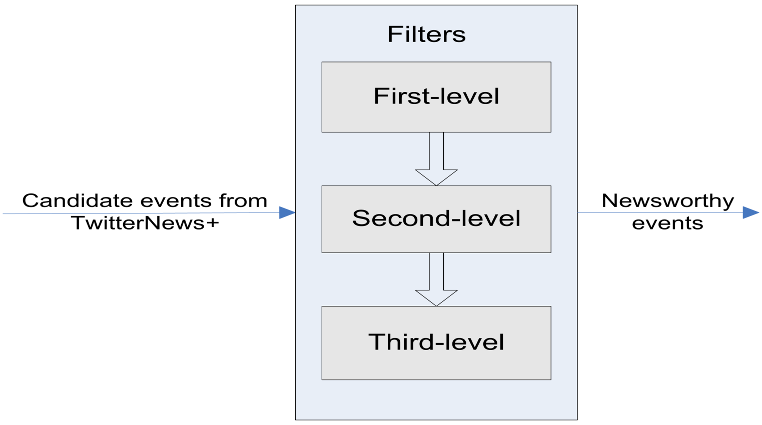


Figure 4.4: The post-processing pipeline in TwitterNews+

news entities. In addition, the event clusters with tweets covering a time span of less than a minute and without a reliable URL of a news portal are filtered out as well. The *second-level* filters help in removing a significant amount of trivial events with very small clusters and contribute highly to the improved precision of our system compared to our previous work, *TwitterNews* [118] (see Section 4.6).

Finally, as a *third-level* filter in *TwitterNews+*, we employ a filtering method based on the Longest Common Subsequence (LCS) approach that works on the word-level (Algorithm 5). The idea here is based on the empirical evidence found from inspecting the candidate event set. We have noticed that news propagated by the users or the news agencies usually follow a similar sentence structure. We have applied the traditional word-level LCS algorithm on the relevant tweets of an event cluster and identified the tweet with the longest common subsequence of words. Then we use the length of the LCS to determine whether the event cluster is about a newsworthy event. If the length of the longest common subsequence of words in an event cluster c is below a certain threshold (t_{lcs}), then the tweets in c do not have an appropriate level of similarity in their sentence structure and c is not considered as a newsworthy event by *TwitterNews+* and thus, discarded. Before applying the LCS-based scheme on the set of candidate events, all the tweets of each event cluster are discarded that do not contain at least one proper noun or possessive noun. Doing so reduces the computational cost of the LCS-based scheme as there will be a lesser number of tweets to process in a cluster by discarding the tweets that do not contain any useful information to describe an event [15].

4.4 Performance Evaluation Process

In order to determine the optimal parameter settings to achieve the best performance in event detection by *TwitterNews+*, we have conducted a series of experiments on the first 3 days worth of tweets from the Events2012 corpus [108] using different parameter settings (see Section 4.5). The corpus contains 120 million tweets collected between the 9^{th} of October and the 7^{th} of November, 2012. Along with the corpus, 506 ground truth

Algorithm 5 . LCS-based filtering and representative tweet selection

```
Require: $t_{lcs}$ value
for each candidate event cluster $c$ in $C$ do                ▷ parallel processing
    $lcs_{max} \leftarrow 0$
    $D \leftarrow$ event $c$ related tweets
    for $i \leftarrow 0$ to $sizeof(D) - 2$ do
        for $j \leftarrow i + 1$ to $sizeof(D) - 1$ do
            $tempLcsLength \leftarrow$ calculateWordLevelLCS($D[i]$, $D[j]$)
            if $tempLcsLength > lcs_{max}$ then
                $lcs_{max} \leftarrow tempLcsLength$
            end if
        end for
    end for
    if $lcs_{max} < t_{lcs}$ then
        remove $c$ from candidate event set
    else
        report the tweet with $lcs_{max}$ as a representative
        of the event cluster $c$
    end if
end for
```

events were provided in a separate file, which we have intended to use to evaluate the results of our experiments.

However, due to a restriction imposed by the Twitter, the Events2012 corpus only contains unique tweet IDs which can be used to download the associated tweets using a web-crawler to populate the corpus. After downloading the tweets, we have inspected the corpus and discovered that a large number of tweets (around 30%) belonging to the corpus were not downloaded as they are not available any more. Sequiera and Lin [119] conducted experiments on the long term effect of tweet deletions from Tweets2013 corpus and noted that the deletions will less likely have an effect on the ranking of systems based on the observation that only 5% of the tweets from the relevance judgments were missing in the corpus. However, the same observation is not valid for the Events2012 corpus as around 50% of the relevance judgments are deleted which was also reported by Repp [120].

The effect of a partially incomplete corpus, due to the unavailability of the tweets, is going to negatively impact the results produced by our system and the baselines. To remedy this problem, we have decided to manually reconfirm the ground truth events provided by the authors [108]. The problem to deal with here is the substantial amount of time required to manually reconfirm the 506 ground truth events. After a careful feasibility consideration, we have only reconfirmed the first three days (9^{th} to 11^{th} of October, 2012) of the ground truth events and manually selected a total of 41 events that belong to our selected time window. Further inspection of these 41 events were required to identify and remove the events which contained a large number of unavailable tweets. Doing so led us to a final set of 31 events to be used as the ground truth (see Table 3.1). We have only kept the ground truth events for which at least five relevance judgments are available. After an analysis of different evaluation methods in the survey on different Twitter-centric event detection approaches [1], we observe that the use of a corpus of approximately 17 million tweets with 31 ground truth events that cover a span of three days is sufficient to be used in experimental evaluations. The selected corpus contains sufficient topic diversity generally encountered in the Twitter data stream and the 31 ground truth events discuss diverse topics with a varying volume of tweets. Note that, in order to provide sufficient time for an event detection system to continue tracking the events detected on the third and final day, as some events can span a long period of time, we ran our experiments on approximately 5 days worth of approximately 17 million tweets.

Setup of experiments. It is difficult for the ground truth events to provide a complete coverage of all the events discussed in Twitter during the Events2012 corpus creation time frame. This led to the rationale that the ground truth events should be utilized only for measuring the recall of an event detection system as any system should be able to detect at least the ground truth events among the many others that are missing in the ground truth. On the other hand, the precision needs to be measured by human evaluators by going through all the events produced by an event detection system. The use of human judgment is required to calculate the precision as the ground truth events alone do not account for all the possible events.

To evaluate the performance of *TwitterNews+* and five state-of-the-art baselines, we calculate the recall and the precision of the reported newsworthy events. In our evaluation process, the recall refers to the fraction of the events in the ground truth that were detected by the system and the precision refers to the fraction of the newsworthy events out of all the events detected by the system.

McMinn et al [108] used automated methods at different levels of granularity which captured sub-events that describe only a part of the event. This means that the tweets provided as relevance judgments do not provide a complete coverage of an event. Moreover, around 50% tweets from the relevance judgments are missing in the corpus due to the long term effect of tweet deletions. Therefore, to calculate the recall we have provided the human annotators the descriptions of 31 ground truth events and the associated relevance judgments only to be used as a guideline to identify the event clusters that discuss the topics in the ground truth events. A match with the ground truth is decided by an evaluator based on the fact that the event cluster consists mostly of similar tweets (above 60%) that discuss the same topic as in one of the ground truth events.

The value of the recall proportionately increases by the number of events out of the 31 ground truth events detected by an event detection system. Due to the manageable amount of ground truth events, the calculation for the recall is feasible in terms of time requirement. On the other hand, the time it can take to calculate the precision, which involves manually determining the newsworthiness of all the events reported by an event detection system, depends on the number of events needed to be judged by a human annotator. McMinn and Jose [15] noted that a lot of events can be detected from the Events2012 corpus in addition to the set of 506 events provided as the ground truth. This observation was further confirmed as our experiments on the first three days of tweets from the corpus yielded a substantially higher number of events than the 31 ground truth events occurring within that time span. The calculation of the precision for a substantially higher number of events for each of the experiments that we have conducted is not feasible. Hence, we have asked two human annotators to determine the precision of 100 randomly chosen events out of all the events reported by an event detection system. The events reported by an event detection system is assigned a unique number, starting from one, as an event ID. Initially, a random number generator was used to obtain 100 random numbers with an upper limit on the magnitude of a number which was set equivalent to the total number of events generated by an event detection system. For a fair evaluation of the precision, the same set of 100 random numbers were used after each of the experiments to extract the events having the same numbers as their event IDs. The precision is calculated as a fraction of the events out of the 100 randomly chosen events that are considered as newsworthy by the annotators and the agreement between the two annotators is measured using Cohen's kappa coefficient [121]. The human evaluators were asked to label the event clusters generated by an event detection system as newsworthy which consist mostly of similar tweets (above 60%) and discuss topics such as business and economy, law and politics, science and technology, disasters and accidents, armed conflicts, crime, sports, arts, culture and entertainments, and any topic of importance.

4.5 Parameter Sensitivity Analysis

TwitterNews+ utilizes a number of different parameters to detect events in real-time. The correct parameter settings for *TwitterNews+* will have an effect on its event detection performance in terms of recall and precision. In this section, we discuss the parameter sensitivity analysis conducted on our system, in order to determine the optimal parameter settings which provides the best performance in detecting newsworthy events. We also discuss the degree of association between the different parameters using correlation analyses and investigate the impact of these parameters on *TwitterNews+*'s performance.

Table 4.2 summarizes the various parameters used in *TwitterNews+* and the different experiments we have conducted to perform a parameter sensitivity analysis on a computer equipped with a quad core 3.40 GHz processor (Intel® Core™ i7-4770) and 16 GB RAM. We have used a one-at-a-time sensitivity measure known as 'local' sensitivity analysis [115, 116] where one parameter at a time is repeatedly varied while keeping the others fixed.

Table 4.2: An outline of the parameter sensitivity analysis for TwitterNews+

Parameter Description	Notation	Experiment
The number of most recent tweets to store for similarity comparison	M	$M0$: 50000 $M1$: 100000 $M2$: 200000 $M3$: 400000 $M4$: 800000
The initial expiry time for an event cluster (minutes)	ts_i	ts_i0: 10 ts_i1: 20 ts_i2: 30 ts_i3: 60 ts_i4: 90 ts_i5: 120
The threshold for the cosine similarity measure in the Search Module which is fixed as 0.6, based on the findings from the related works [8, 15] and our prior experiments	t_{sr}	n/a
The threshold for the cosine similarity measure in the EventCluster Module which is also fixed as 0.6, same as the cosine similarity threshold in the Search Module	t_{ev}	n/a
The number of tweets to store in each row of the *term-tweets* and the *term-eventIDs* inverted indices	Q	$Q0$: 5 $Q1$: 25 $Q2$: 50 $Q3$: 100 $Q4$: 200
The threshold for the entropy-based event filter	t_{ent}	$t_{ent}0$: 0.0 $t_{ent}1$: 2.0 $t_{ent}2$: 2.5 $t_{ent}3$: 2.8 $t_{ent}4$: 3.0
The threshold for the LCS-based event filter	t_{lcs}	$t_{lcs}0$: 4 $t_{lcs}1$: 5 $t_{lcs}2$: 6 $t_{lcs}3$: 7 $t_{lcs}4$: 8 $t_{lcs}5$: 9
The threshold to merge the fragmented event clusters	g_{ev}	$g_{ev}0$: 0.05 $g_{ev}1$: 0.06 $g_{ev}2$: 0.07 $g_{ev}3$: 0.08

Determining the Optimal M Value. An optimal value for each parameter is obtained by varying its value while leaving all others constant, and quantifying the change in terms of recall and precision, calculated from the events reported by *TwitterNews+*. We start the parameter sensitivity analysis on the different M values which represent the number of most recent tweets stored by our system in order to perform a text similarity comparison with the input tweet. Based on the empirical evidence gathered from the different experiments that we have run over the course of implementing the system, we start the experiments for parameter sensitivity analysis with a baseline parameter settings (Figure 4.5a) which gave us fairly good results and conduct experiments with different M values ranging from $M0$: 50000 to $M4$: 800000. Note that the notation $M0$ refers to the experiment with a parameter setting as shown in Figure 4.5a and a value of 50000 for M.

Figure 4.5: The baseline parameter settings in different stages of the parameter sensitivity analysis of TwitterNews+

(a) M

Parameter	Value
ts_i	20
t_{sr}	0.6
t_{ev}	0.6
Q	25
t_{ent}	2.5
t_{lcs}	5
g_{ev}	0.07

(b) ts_i

Parameter	Value
M	400000
t_{sr}	0.6
t_{ev}	0.6
Q	25
t_{ent}	2.5
t_{lcs}	5
g_{ev}	0.07

(c) Q

Parameter	Value
M	400000
ts_i	60
t_{sr}	0.6
t_{ev}	0.6
t_{ent}	2.5
t_{lcs}	5
g_{ev}	0.07

(d) t_{ent}

Parameter	Value
M	400000
ts_i	60
Q	100
t_{sr}	0.6
t_{ev}	0.6
t_{lcs}	5
g_{ev}	0.07

(e) t_{lcs}

Parameter	Value
M	400000
ts_i	60
Q	100
t_{sr}	0.6
t_{ev}	0.6
t_{ent}	2.8
g_{ev}	0.07

(f) g_{ev}

Parameter	Value
M	400000
ts_i	60
t_{sr}	0.6
t_{ev}	0.6
Q	100
t_{ent}	2.8
t_{lcs}	6

Figure 4.6a summarizes the effect on *TwitterNews+*'s performance, in terms of recall and precision, for different M values. In addition, Figure 4.7a illustrates the run-time dependency of our system on the different M values. Note that we only calculate the precision for the M values that yield the highest recall. Therefore, we have calculated the precision for the experiments $M2$ to $M4$, which had the highest recall of 0.96, and found that experiment $M3$ has the highest precision of 0.80. It means experiment $M3$ achieves the best performance for *TwitterNews+* with an M value of 400000. We notice that the experiments conducted to tune the parameter, M, achieve the highest recall within the range of $M2$ to $M4$. However, there is a drop in precision from $M3$ to $M4$ which suggests that between a specific range of values of M there are insignificant or no changes in the recall but the precision of *TwitterNews+* is impacted negatively when the value of M is above a certain value. The reason behind the detrimental effect on the precision could be from the fact that comparing a new tweet with 800,000 ($M4$) previously encountered tweets to determine a soft burst on a topic, suffers more from the topic drifts that happen over time than comparing a new tweet with 400,000 ($M3$) tweets.

Determining the Optimal ts_i Value. Once the M value for our system has been determined, we subsequently conduct experiments with different ts_i values, which represent the initial cluster expiry time that will be assigned when an event cluster is created. In the experiments, while the optimal ts_i value is being determined, the other parameter settings are kept fixed as shown in Figure 4.5b. The results for the experiments to determine the optimal ts_i value are summarized in Figure 4.6b and the run-time dependency of our system on the different ts_i values is shown in Figure 4.7b. Similar to the experiments to determine the M value, we only calculate the precision for the ts_i values that yield the highest recall. We have calculated the precision for the experiments ts_i2 to ts_i5, which had the highest recall of 0.96. The highest precision of 0.82 was achieved by *TwitterNews+* with a ts_i value of 60 minutes.

An event discussed on Twitter takes some time to reach its peak discussion point and afterwards the volume of tweets discussing the associated topics reduces over time. The amount of time an event cluster will be active is a combination of the initial cluster

Figure 4.6: TwitterNews+'s performance on the different parameter settings

(a) M

Experiment	Recall	Precision
$M0$: 50000	0.93 (29/31)	n/a
$M1$: 100000	0.93 (29/31)	n/a
$M2$: 200000	0.96 (30/31)	0.78
$M3$: 400000	0.96 (30/31)	0.80
$M4$: 800000	0.96 (30/31)	0.77

(b) ts_i

Experiment	Recall	Precision
ts_i0: 10	0.93 (29/31)	n/a
ts_i1: 20	0.93 (29/31)	n/a
ts_i2: 30	0.96 (30/31)	0.78
ts_i3: 60	0.96 (30/31)	0.82
ts_i4: 90	0.96 (30/31)	0.79
ts_i5: 120	0.96 (30/31)	0.79

(c) Q

Experiment	Recall	Precision
$Q0$: 5	0.90 (28/31)	n/a
$Q1$: 25	0.96 (30/31)	0.80
$Q2$: 50	0.93 (29/31)	n/a
$Q3$: 100	0.96 (30/31)	0.83
$Q4$: 200	0.93 (29/31)	n/a

(d) t_{ent}

Experiment	Recall	Precision
$t_{ent}0$: 0.0	0.96 (30/31)	0.68
$t_{ent}1$: 2.0	0.96 (30/31)	0.79
$t_{ent}2$: 2.5	0.96 (30/31)	0.83
$t_{ent}3$: 2.8	0.96 (30/31)	0.85
$t_{ent}4$: 3.0	0.77 (24/31)	n/a

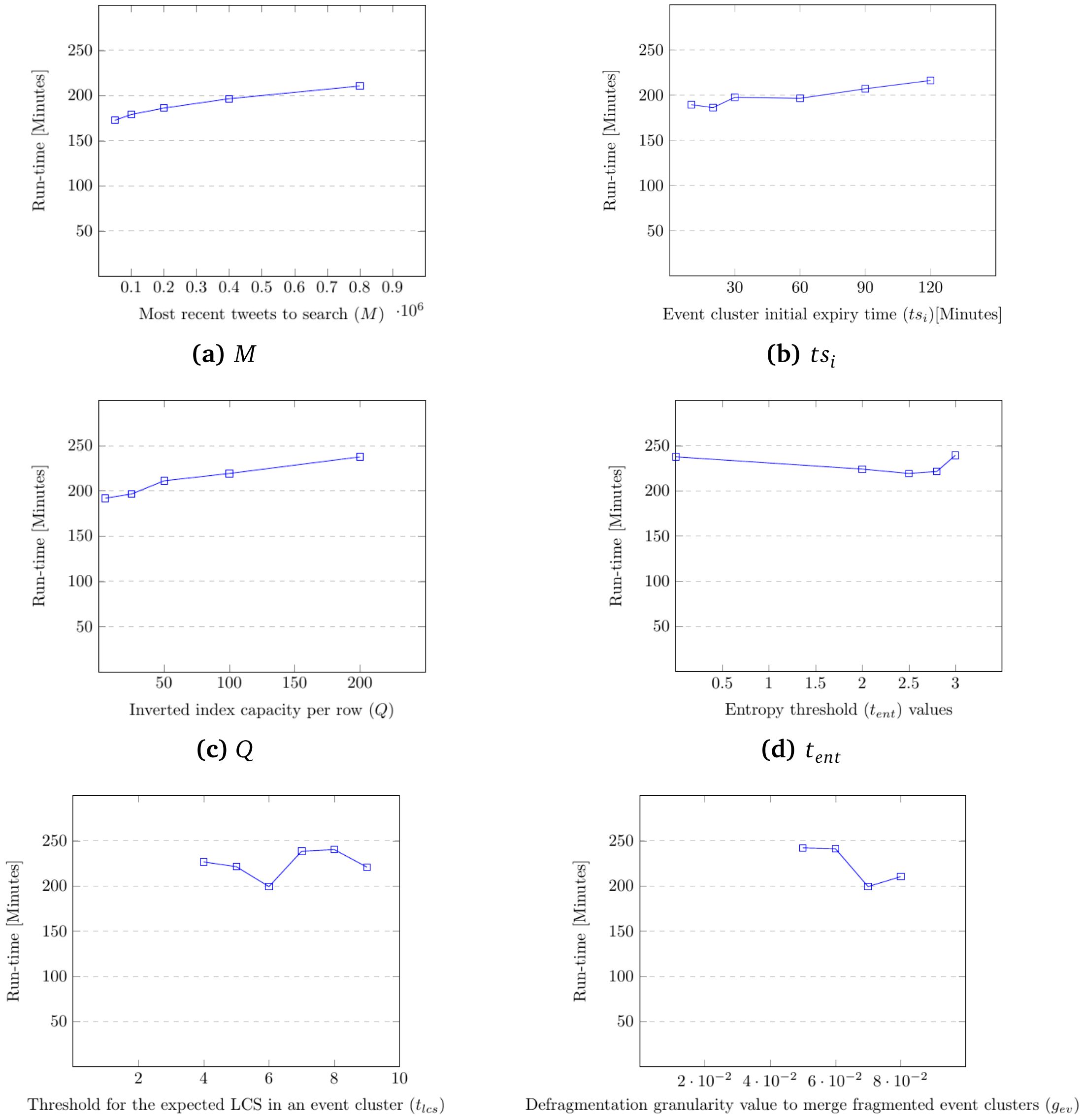
(e) t_{lcs}

Experiment	Recall	Precision
$t_{lcs}0$: 4	0.96 (30/31)	0.78
$t_{lcs}1$: 5	0.96 (30/31)	0.85
$t_{lcs}2$: 6	0.96 (30/31)	0.89
$t_{lcs}3$: 7	0.93 (29/31)	n/a
$t_{lcs}4$: 8	0.90 (28/31)	n/a
$t_{lcs}5$: 9	0.90 (28/31)	n/a

(f) g_{ev}

Experiment	Recall	Precision
$g_{ev}0$: 0.05	0.96 (30/31)	0.79
$g_{ev}1$: 0.06	0.96 (30/31)	0.81
$g_{ev}2$: 0.07	0.96 (30/31)	0.89
$g_{ev}3$: 0.08	0.96 (30/31)	0.83

expiry time ts_i that is assigned when the cluster was created and the average timestamp difference between the arrival of successive tweets in the cluster. The initial expiry time can be thought of as the time it takes for an event to reach its peak discussion point. On the other hand, the use of the average timestamp difference between the arrival of successive tweets in a cluster allows *TwitterNews+* to dynamically decide the end of an event. If the temporal behavior of an event is not rightly captured, it could have a detrimental effect on the recall and the precision of *TwitterNews+*. We notice that the experiments conducted to tune the parameter, ts_i, suffers from a drop in precision from ts_i3 to ts_i4 which suggests that an initial cluster expiry time of 60 minutes is the time that it usually takes for an event to reach its peak discussion point and thus captures the temporal behavior of an



(a) M **(b)** ts_i

(c) Q **(d)** t_{ent}

(e) t_{lcs} **(f)** g_{ev}

Figure 4.7: TwitterNews+'s run-time dependency on the different parameter settings

event as close as possible.

Determining the Optimal Q Value. At this stage of our parameter sensitivity analysis, we have determined the optimal value for two parameters: M and ts_i. Furthermore, based on the findings from the related works [8, 15] and our prior experiments, the value for both of the thresholds (i.e., t_{sr} and t_{ev}) used for the cosine similarity measure in the Search Module and the EventCluster Module is determined to be 0.6. Subsequently, we conduct experiments to determine the optimal Q value, which represents the number of tweets to store in each row of the *term-tweets* and *term-eventIDs* inverted indices. We conduct our experiments while keeping the other parameter settings fixed, as shown in Figure 4.5c. The results for the experiments to determine the optimal Q value are summarized in Figure 4.6c. Furthermore, Figure 4.7c illustrates the run-time dependency of our system on the different Q values. Similar to the previous experiments, we only calculate the precision for the Q values that yield the highest recall. We have calculated the precision for the experiments $Q1$ and $Q3$, which had the highest recall of 0.96. The highest precision of 0.83 was achieved by *TwitterNews+* with a Q value of 100.

Determining the Optimal t_{ent} Value. The next phase of the experiments is conducted to determine the optimal t_{ent} value, while keeping the other parameter settings fixed as shown in Figure 4.5d. The value of t_{ent} represents the amount of information contained in an event cluster, which can be used to decide to some extent whether an event cluster has enough information to be considered as a newsworthy event. In Figure 4.6d, the results of the experiments to determine optimal t_{ent} value are summarized and in Figure 4.7d the run-time dependency of our system on the different t_{ent} values is illustrated. We have calculated the precision for the t_{ent} values that yield the highest recall. Experiments $t_{ent}0$ to $t_{ent}3$ had the highest recall of 0.96 and the highest precision of 0.85 was achieved by *TwitterNews+* with a t_{ent} value of 2.8.

The t_{ent} value acts as a filter to remove the candidate event clusters, that do not contain sufficient information, to improve the precision of our system. This means that the absence of this filter will not have any effect on the recall which was confirmed in our experiments. However, a high enough value set for t_{ent} can filter out non-trivial events which is evident from a sudden drop on recall when t_{ent} was set to 3.0.

Determining the Optimal t_{lcs} Value. Similar to the t_{ent} value, which is used as a filter to discard trivial event clusters, the t_{lcs} value is also used as an additional filter to discard the event clusters in which the tweets do not have a certain level of similarity in their sentence structure. We have conducted experiments to determine the optimal t_{lcs} value, while keeping the other parameter settings fixed, as shown in Figure 4.5e. In Figure 4.6e, the results of the experiments to determine optimal t_{lcs} value are summarized and in Figure 4.7e the run-time dependency of our system on the different t_{lcs} values is illustrated. We have calculated the precision for the t_{lcs} values that yield the highest recall. Experiments $t_{lcs}0$ to $t_{lcs}2$ had the highest recall of 0.96 and the highest precision of 0.89 was achieved by *TwitterNews+* with a t_{lcs} value of 6.

The t_{lcs} value acts as a filter to remove the trivial candidate event clusters to improve the precision of our system. This means that the absence of this filter will not have any effect on the recall, similar to the t_{ent} filter; but a high enough value set for t_{lcs} can filter out non-trivial events, which is evident from the decreasing recall values when t_{ent} was set to above 6.

Determining the Optimal g_{ev} Value. The final part of our experiments determines the optimal value for the defragmentation granularity (g_{ev}), which is used with the t_{ev} value to determine whether two fragmented event clusters should be merged together or not. The parameter settings for other parameters, while conducting the experiments to determine the optimal g_{ev} value, is shown in Figure 4.5f. We refer to Figure 4.7f, where the run-time dependency of our system on the different g_{ev} values is illustrated. Unless set to high, the value of g_{ev} should not affect the recall as it aims to achieve higher precision by merging the sub-events of an event. The results of the experiments in Figure 4.6f to determine the optimal g_{ev} value confirm our hypothesis as the recall value for all the experiments were the same. The highest precision of 0.89 was achieved by *TwitterNews+* with a g_{ev} value of 0.07 which provides the highest improvement on the quality of the clusters.

As a result of our parameter sensitivity analysis, we have determined the optimal parameter settings for *TwitterNews+*, shown in Table 4.3, which achieves a recall of 0.96 and a precision of 0.89.

Correlation analysis. We have performed a correlation analysis using RapidMiner[1] to determine the degree of association between the different parameters and the recall values achieved by *TwitterNews+* in each parameter setting (Figure 4.8). From the Pearson's correlation coefficient matrix in Figure 4.8 we can determine that the parameter Q, which is used to set the row capacity of both of the inverted-indices utilized in *TwitterNews+*,

---

[1] `https://rapidminer.com/`

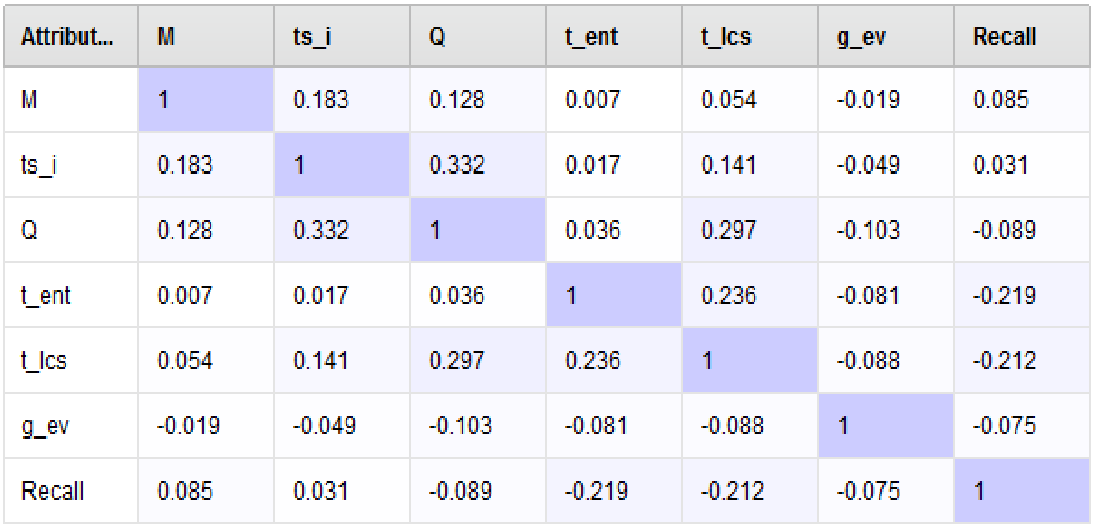
Table 4.3: The recommended parameter settings for TwitterNews+

Parameter	Value
M	400000
ts_i	60
t_{sr}	0.6
t_{ev}	0.6
Q	100
t_{ent}	2.8
t_{lcs}	6
g_{ev}	0.07

is positively correlated to an event cluster's initial expiry time (ts_i) and the LCS-based scheme's threshold (t_{lcs}) parameters to a small degree. Moreover, both the entropy threshold (t_{ent}) and the the LCS-based scheme's threshold (t_{lcs}) parameters have a negative correlation with recall to a small degree.

A Pearson's correlation coefficient of $r = 1$ is only between the same parameters and subsequently the highest Pearson's correlation coefficient of $r = 0.332$ is between the parameters Q and ts_i with a sample size of 31. However, there is no strong evidence for the hypothesis that there exists a linear relationship between the parameters in *TwitterNews+*, as $r = 0.332$, the highest correlation coefficient in Figure 4.8 does not exceed the critical value of 0.367 at $P < 0.05$ for it to be statistically significant. Similarly, there is no strong evidence that there is a linear relationship between the *TwitterNews+* parameters and recall.

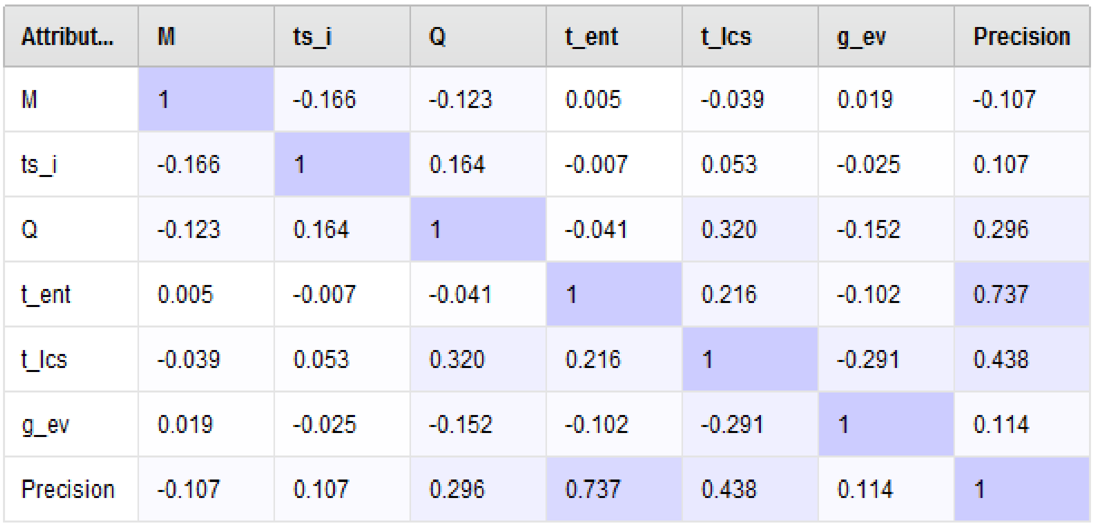
A correlation analysis was also performed between the different parameters and the precision values achieved by *TwitterNews+* in each parameter setting (Figure 4.9). From the Pearson's correlation coefficient matrix in Figure 4.9 we can determine that the entropy threshold (t_{ent}) and the LCS-based scheme's threshold (t_{lcs}) parameters have a strong and moderate level of correlation with precision, respectively. The parameter Q to set the row capacity of the inverted indices is positively correlated to t_{lcs} and precision to a small degree. Furthermore, t_{lcs} is negatively correlated to a small degree to g_{ev} which is the threshold parameter to merge the fragmented event clusters. The highest Pearson's correlation coefficient of $r = 0.737$ is between the parameter t_{ent} and recall with a sample



Attribut...	M	ts_i	Q	t_ent	t_lcs	g_ev	Recall
M	1	0.183	0.128	0.007	0.054	-0.019	0.085
ts_i	0.183	1	0.332	0.017	0.141	-0.049	0.031
Q	0.128	0.332	1	0.036	0.297	-0.103	-0.089
t_ent	0.007	0.017	0.036	1	0.236	-0.081	-0.219
t_lcs	0.054	0.141	0.297	0.236	1	-0.088	-0.212
g_ev	-0.019	-0.049	-0.103	-0.081	-0.088	1	-0.075
Recall	0.085	0.031	-0.089	-0.219	-0.212	-0.075	1

Figure 4.8: The Pearson's correlation coefficient matrix for TwitterNews+'s parameters and recall

size of 20, which exceeds the critical value of 0.468 at $P < 0.05$. However, there is no strong evidence for the hypothesis that there exists a linear relationship between the parameters in *TwitterNews+*, as none of the correlation coefficients (besides $r = 1$ and $r = 0.737$) in Figure 4.9 exceeds the critical value of 0.468 at $P < 0.05$ for it to be statistically significant.



Attribut...	M	ts_i	Q	t_ent	t_lcs	g_ev	Precision
M	1	-0.166	-0.123	0.005	-0.039	0.019	-0.107
ts_i	-0.166	1	0.164	-0.007	0.053	-0.025	0.107
Q	-0.123	0.164	1	-0.041	0.320	-0.152	0.296
t_ent	0.005	-0.007	-0.041	1	0.216	-0.102	0.737
t_lcs	-0.039	0.053	0.320	0.216	1	-0.291	0.438
g_ev	0.019	-0.025	-0.152	-0.102	-0.291	1	0.114
Precision	-0.107	0.107	0.296	0.737	0.438	0.114	1

Figure 4.9: The Pearson's correlation coefficient matrix for TwitterNews+'s parameters and precision

4.6 Results and Discussion

We have evaluated the events reported by *TwitterNews+* within the time window of three days. The five baselines we have used to compare with our system are: First Story Detection (FSD) system [8], an incremental-clustering-based approach; *TwitterNews* [118], our previously proposed system also based on incremental clustering; LDA-SocialSensor[2], a topic-modeling-based approach [103]; TrendingScore[3] [122], a term-interestingness-based approach; and mention-anomaly-based Event Detection (MABED) [17], a statistical approach for event detection. The baselines selected for evaluation are state-of-the-art event detection systems, most of which are publicly available for use and cover a wide range of event detection techniques, so that it is easy to compare the performance of our system with that of the other systems. Table 4.4 summarizes the results of our evaluation.

Table 4.4: A summary of the evaluation results

Method	Recall	Precision
FSD [8]	0.52	-
LDA-SocialSensor [103]	0.45	0.49
MABED [17]	0.58	0.55
TrendingScore [122]	0.71	0.64
TwitterNews [118]	0.87	0.72
TwitterNews+	0.96	0.89

For all the baselines we have used the recommended and/or the best parameter setting (see Table 4.5). To achieve a fair evaluation, the input dataset from the Events2012 corpus went through the same pre-processing phase as *TwitterNews+* before conducting experiments on the baselines.

Our experiments revealed that *TwitterNews+* has detected the highest number of ground truth events (30 events out of 31), resulting in a recall of 0.96. Table 4.6 shows the number of ground truth events detected by different event detection systems that we have tested. While consolidating the ground truth events, McMinn et al. [108] used automated methods at different levels of granularity which captured: 1) sub-events that

---

[2] `http://www.socialsensor.eu/results/software/87-topic-detection-framework`

[3] `https://github.com/AdrienGuille/SONDY`

Table 4.5: The parameter settings used for the baselines

Method	Parameter Setting
FSD [8]	cosine similarity threshold = 0.6
LDA-SocialSensor [103]	no of topics = 500 no of training iterations = 300 no of keywords returned per topic = 10

MABED [17]	minTermSupport = .0001
	maxTermSupport = .01
	k = 500
	p = 10
	theta = 0.7
	sigma = 0.7

TrendingScore [122]	minTermSupport = .0001
	maxTermSupport = .01
	trendingThreshold = 10.0

TwitterNews [118]	similar to *TwitterNews+*

describe only a part of an event, 2) minor events with a small number of associated tweets, and 3) major events with a large number of associated tweets. As most of the state-of-the art approaches focus on detecting major events based on a substantial burst in the volume of tweets that discuss a particular topic, they are unable to detect minor events that generate a low volume of tweets. Our investigation into the events detected by the baselines show that they are unable to detect the minor events in most cases, while performing well in detecting major events related to sports, politics or the topics that instigated a substantial burst in the volume of tweets. As *TwitterNews+* requires a soft burst to start tracking an event, it is much more suitable to detect the minor events, compared to the baselines, which contributed to the higher recall achieved by our system.

Table 4.6: The number of the ground truth events detected by different event detection systems

Method	Number
FSD [8]	16
LDA-SocialSensor [103]	14
MABED [17]	18
TrendingScore [122]	22
TwitterNews [118]	27
TwitterNews+	30

TwitterNews+ also outperforms the baselines by achieving the highest precision of 0.89 (89 out of 100 events were agreed as newsworthy events by the annotators). We notice that the baselines perform reasonably well on very focused events, but they are susceptible to noise which degrades their performance in terms of precision as they often incorrectly detect a noisy topic as a newsworthy event. The use of various filters in the post-processing phase of *TwitterNews+*, different from our previously proposed system *TwitterNews* [118], was instrumental in the substantial improvement of precision compared to the precision of 0.72 achieved by *TwitterNews*. The post-processing phase of *TwitterNews+* allows our system to successfully filter out most of the trivial events and substantially contributes to the precision achieved by our system. Table 4.7 shows a subset of event related tweets of the selective ground truth events that were detected by *TwitterNews+* from the first three days of tweets contained in the Events2012 corpus [108].

Efficiency evaluation by analyzing computational complexity mathematically is rarely done in the literature on event detection systems [1]. Besides achieving good performance in terms detecting newsworthy events, the goal for an event detection system is also to achieve real-time processing capability, which we believe should be measured in terms of the *number-of-tweets/second* processing capability as we can always find the information on the average number of tweets posted on Twitter per second. The experiments we have conducted on the baselines required the whole dataset to be preprocessed according to some specifications before applying an event detection technique, which made it impossible to determine the *number-of-tweets/second* processing capability of most of these baselines. Although the LDA-SocialSensor [103] baseline did not require the whole dataset to be preprocessed in advance, processing around 17 million tweets was computationally prohibitive to determine its *number-of-tweets/second* processing capability.

Table 4.7: A subset of event-related tweets for selective ground truth events, reported by TwitterNews+

Event Topic	Event Tweet
Jerry Sandusky	Penn State's Sandusky gets 30-60 years prison for child abuse http://reut.rs/QQ0XqXÃĆ via @reuters
	Behind bars, Sandusky may face threats and isolation: (Reuters) - Jerry Sandusky will soon join about 6,700 othe... http://yhoo.it/WOPTeKÃĆ
	Victim: 'Because of you, I trust no one': The young man locked eyes with Jerry Sandusky in a packed courtroom Tuesday and stared him ...
Nobel prize in literature	Nobel Prize for literature awarded - Mo Yan of China won the prize for his novel "Frog", which explores the traditio... http://ow.ly/2sCWeyÃĆ
	Congrats to Mo Yan for being the 1st Chinese Nobel Prize of Literature laureate!
	"108 authors have received the Nobel Prize in Literature.... 12 women have been awarded the Nobel Prize in Literature so far".
Los Zetas	Mexico confirms death of feared Zetas boss: Mexico confirmed Tuesday that its forces killed Heriberto Lazcano La... http://bit.ly/R65HpxÃĆ
	Mexico says drug lord taken down by accident: The Mexican navy says a team of marines had no idea that they had ... http://bit.ly/R8myrEÃĆ
	Body of brutal Los Zetas drug kingpin snatched from a funeral home by armed gunmen http://bit.ly/VO0j0oÃĆ
Buster Posey	Posey's grand slam sends Giants over Reds 6-4 to win NLDS, capping comeback from 2-0 deficit http://bit.ly/VWowSiÃĆ
	Posey, Giants knock off Reds, headed to NLCS: Buster Posey hit the third grand slam in Giants' postseason histor... http://es.pn/VW8WWyÃĆ
	Buster Posey just won the MVP award
Marie Stopes	Marie Stopes private abortion clinic to open in Northern Ireland: A new Marie Stopes private abortion clinic is... http://adf.ly/DbLmqÃĆ
	A bishop has criticised the opening of a Marie Stopes clinic offering abortion in Belfast: http://jrnl.to/TCWwCgÃĆ
	Northern Ireland's first abortion clinic to open in Belfast: Marie Stopes predicts woman from Republic will also... http://bit.ly/RcmB5ZÃĆ

The average run time for the 33 experiments we have conducted during our parameter sensitivity analysis for *TwitterNews+* is approximately 212 minutes for 17 million tweets, which means our system is capable of processing 1336 tweets per second. The number of tweets[4] on average tweeted on Twitter every second at the time of writing this thesis is 6000. However, only a small sample (around 1%) of the public data is made available through the Twitter Streaming API. Therefore, in order to achieve real-time processing capability, a Twitter-centric event detection system should be able to process 60 tweets per second (1% of 6000). *TwitterNews+* easily surpasses this requirement, which makes our system capable of dealing with the Twitter data stream in real-time.

In both the Search Module and EventCluster Module of our system, we perform a cosine similarity comparison between a new tweet and a fixed number of vectors which is quite expensive if done without multi-threading. But with multi-threading, depending on the number of cores available on a computer, the computational complexity for a fixed number of independent cosine similarity calculations becomes $O(1)$. Introducing multi-threading to compute the cosine similarity reduced the overall execution time of *TwitterNews+*, on a quad-core machine, to less than a one-third of the execution time it took without multi-threading. As the cosine similarity comparisons, which are the most expensive operation in *TwitterNews+*, are designed to be processed in parallel, a more powerful processor with a higher number of cores will increase the processing capability of *TwitterNews+* if the need arises.

4.7 Conclusion

The approach taken in *TwitterNews+* yields a low computational cost solution to detect events from a high volume streaming-data source in real-time, which is lacking in most of the state-of-the-art approaches. The most expensive operations in the Search Module and the EventCluster Module algorithms of *TwitterNews+* incur a computational complexity of $O(1)$ with parallel processing. The different set of filters, applied after the candidate events generation, collectively incur a computational cost of $O(n^2)$, where n refers to

---

[4] http://www.internetlivestats.com/twitter-statistics/

the number of tweets in an event cluster. However, the filters are applied as a separate process independent of the candidate event generation stages of *TwitterNews+* and the total number of tweets (n) in the event clusters is not high enough to be a deterrent in the performance of our system which is evident from the average run time of the experiments we have conducted.

We have conducted a parameter sensitivity analysis using a one-at-a-time sensitivity measure to determine the optimal parameter settings for *TwitterNews+*. The parameter settings that we have obtained for our system substantially improves its performance in detecting newsworthy events.

TwitterNews+ is an efficient system to detect events from the Twitter data stream in real-time with a high *number-of-tweets/second* processing capability, it maintains a constant space and processing time, and achieves very good results compared to the baselines used in our experiments. The different set of filters, applied to extract newsworthy events from the set of candidate events, helps in retaining both major and minor events, and discarding a significant amount of trivial events. This is, again, where most of the state-of-the-art approaches fail, which focus on detecting events based on a burst detection approach that require a substantial burst in the volume of tweets discussing a topic. Finally, the evaluation of *TwitterNews+*, done using a publicly available corpus, will allow researchers to compare different systems fairly against our system.

5

Providing a Context to an Event

5.1 Introduction

There exists a number of different temporal patterns for how an event unfolds in online media [104, 105], which implies that any event detection approach may suffer from capturing only a portion of the whole event. As a consequence, an event cluster consisting of event-related tweets often contains information on a sub-topic of an event, leading to a lack of sufficient insight into the detected event. For example, an event, detected by the *TwitterNews+* event detection system [123], regarding an oil tanker explosion contains tweets from which it can be deduced that an Iranian oil tanker has exploded and sank off the coast of China with no survivors. The group of tweets (e.g., "#Iran oil tanker explodes off coast of #China with no survivors https://t.co/T3Xu4xrmPQ") clustered by *TwitterNews+* contain no further information about the event. The problem of providing

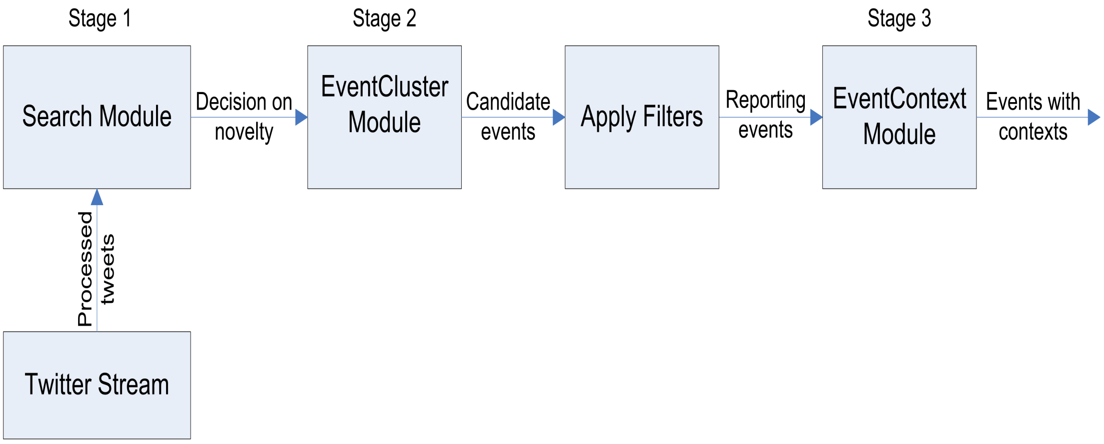


Figure 5.1: The pipeline of TwitterNews+

a context to an event, detected by an event detection system, is to provide additional information to supplement the information contained in the event-related tweets.

In this chapter we discuss a novel approach which is implemented as the EventContext Module, the third major component of *TwitterNews+*, that provides a context to an event by retrieving additional tweets from the Twitter Search API. The EventContext Module works independently of the event detection processes of *TwitterNews+* (Figure 5.1), which implies that it can be used by any event detection system to gain additional information about an event.

Unlike the Twitter Streaming API which provides a small percentage of real time feeds of Twitter's public statuses, the Twitter Search API's[1] historical access to the previous one week of the Twitter content can be utilized to retrieve additional tweets about an event so that a relevant subset of the retrieved tweets can be used as context tweets. The search query used for the Twitter Search API consists of terms which are the most frequent proper and common nouns found in the event-related tweets and retrieves tweets that contain at least one of the terms in the query. Afterwards, the EventContext Module selects a subset of the retrieved tweets that serves as a context to the detected event. Therefore, the problem of providing a context to an event is addressed by employing a method that selects the top-ranked subset of tweets from the retrieved tweets that are relevant to the

---

[1] `https://developer.twitter.com/en/docs/tweets/search/guides/standard-operators`

event. In order to rank the retrieved tweets' relevancy with the event-related tweets, using the cosine similarity measure, we have compared three different methods to create tweet vectors: 1) a probabilistic-feedback-based method, 2) a random-indexing-based method, and 3) a TF-IDF-based method, which is used as a baseline.

Traditionally, the incremental TF-IDF-based method to create tweet vectors has been utilized to measure similarity between tweets using the cosine similarity measure in a streaming setting [1]. In a non-streaming setting where a corpus consists of the event-tweets and the retrieved tweets, we have considered the use of probabilistic feedback [124] and random indexing [9] based term correlation and term co-occurrence models, respectively, as a novel approach to create tweet vectors which can be compared against the TF-IDF-based baseline. As the information contained in a tweet is formalized in a tweet vector, we have explored different methods of creating tweet vectors to decide on the method to be integrated as part of the EventContext Module that aids in selecting the maximum percentage of relevant tweets from the retrieved tweets. Our experiments revealed that the probabilistic-feedback-based method used in the EventContext Module outperforms the other two methods of creating tweet vectors as it provides the maximum percentage of relevant tweets in the top-10, top-20, and top-30 ranked context tweets of an event.

The remainder of this chapter is organized as follows: In Section 5.2, we discuss the related work. Section 5.3 contains a discussion on the operation of the proposed EventContext Module and Subsections 5.3.1, 5.3.2, and 5.3.3 deal with the three different methods to create tweet vectors. Section 5.4 contains a discussion on the experimental setup to identify the best performing method to create tweet vectors, the performance evaluation process, and the results of our experiments. Finally, we conclude in Section 5.5.

5.2 Related Work

In this section, we discuss the works that share some similarities to the problem of providing a context to an event. Sharifi et al. [125] developed an algorithm that can automatically generate one-line summaries from tweets related to a given topic. Provided with a topic, the proposed Phrase Reinforcement (PR) algorithm retrieves tweets related to the topic. From each tweet, the longest sentence containing the topic is extracted. These sentences act as an input to the PR algorithm to construct a graph that represents the common sequences of words encompassing the topic. The topic is the root node or the central node, and each input sentence is a sequence of nodes containing a single word. The weight of a node is calculated based on the frequency of the word contained in the node and the distance of the node from the root node. Once the graph is generated, the PR algorithm finds the path with the highest total weight from the root node to a leaf node, to generate the summary.

Madani et al. [47] used TextRank [126], a graph-based unsupervised ranking model for natural language processing that derives extractive summaries from the given texts. From the tweets associated with a topic cluster, a dependency graph is created where the nodes are the tweets in the cluster and the edges between the nodes correspond to the similarity between tweets. The similarity between two tweets tw_i and tw_j based on the set of terms k belonging to the tweets is defined as:

$$Sim(tw_i, tw_j) = \frac{\sum_{k \in tw_i, tw_j} (freq(k, tw_i) + freq(k, tw_j))}{(log(|tw_i|) + log(|tw_j|))} \tag{5.1}$$

TextRank is applied on the tweets dependency graph created from a cluster. In order to rank the tweets to generate a summary, the score of a given tweet tw_i within the cluster is calculated in the following manner:

$$S(tw_i) = \sum_{tw_j \in nested(tw_i)} \frac{Sim(tw_i, tw_j)}{\sum_{tw_l \in nested(tw_j)} Sim(tw_l, tw_j)} S(tw_j) \tag{5.2}$$

where $tw_j \in nested(tw_i)$ refers to the nodes that are connected to tw_i, and $tw_l \in$

$nested(tw_j)$ refers to the nodes that are connected to tw_j.

The ReDites event detection system [50] contains a Tracking and Summarization (TaS) component. When an event is detected, the real-time tracking sub-component retrieves new tweets about the event based on the most informative terms contained in the event-related tweet. In order to retrieve additional tweets, the system maintains a sliding window of the previous tweets collected from the underlying Twitter stream. The summarization sub-component of TaS removes tweets that are textually similar and retains a subset of event-related tweets that serve as a summary [127] of the event.

Khan et al. [128] proposed a system for multi-tweet summarization of those real-time events that induce a substantial response from the Twitter users and take place within a limited time span. A summary of an event is generated by initially identifying popular discussion points within the event, which is achieved by leveraging a basic LDA based topic model [102] to divide the event-related tweets into topic clusters. Subsequently, a lexical graph is created comprising of nodes which correspond to the specific unigrams (i.e., nouns, verbs, and adjectives) extracted from the tweets associated with a topic cluster. The “Likelihood Ratio” measure [129] is used to determine the statistical significance of an identified co-occurrence between unigrams and its strength of association. The measured strength of association is assigned as the edge weight between the nodes corresponding to the co-occuring unigrams. Finally, a variant of the graph-based ranking algorithm, PageRank [130], is applied on the lexical graph constructed from the tweets associated with a topic cluster. The top-k ranked tweets from each topic cluster collectively act as a summary of an event.

The works discussed in this section, focus on generating a summary of the event-related tweets that were grouped by an event detection system. The content of the summary is entirely derived from the event-related tweets. However, no additional information is provided on the detected event that can work as its context. Therefore, the problem of providing a context to an event, detected by an event detection system, is yet to be explored. As event detection systems often detect sub-events, additional information on the event can provide a better understanding of it, which is what we aim to achieve.

5.3 Architecture of the EventContext Module

The Search Module and the EventCluster Module of *TwitterNews+* are collectively responsible for event detection from the Twitter data stream. The EventContext Module (Figure 5.2) provides additional information on an event detected by *TwitterNews+* and operates independently of the event detection process of our system. Once an event is detected by *TwitterNews+*, the EventContext Module is activated which retrieves a set of tweets from the Twitter Search API containing the most frequent proper and common nouns of the event-related tweets. A subset of top ranked tweets from the retrieved set of tweets is then selected by the EventContext Module, as shown in Algorithm 6, which serves as a context to the event. The rank of a retrieved tweet is calculated, using the cosine similarity measure, against the centroid of the vectors asoociated with the event-tweets. The more similarity a retrieved tweet has with the centroid of the event-tweets' vectors, the higher the tweet is ranked by the EventContext Module. In order to calculate the rank of a retrieved tweet, each tweet in the corpus, containing the event-tweets and the retrieved tweets, needs to be assigned a vector in which the information contained in the tweet is formalized. The information contained in a tweet should be formalized in a way so that the ranking of the tweet is more accurate in determining the relevant context tweets from the retrieved tweets. Therefore, we have experimented with three different methods to create tweet vectors, discussed in Subsections 5.3.1, 5.3.2, and 5.3.3, to determine the method that provides the most relevant context tweets.

Once the tweets are ranked and sorted based on their ranks, a subset of the tweets, which are above a certain threshold (t_{con}) of similarity with the event-tweets' centroid, are considered as the candidate context tweets of the event. The j^{th} dimension of the event-tweets' centroid vector is calculated by selecting the maximum weight of the j^{th} dimension contained in the vectors associated with the event-tweets.

As the retrieved tweets suffer from the presence of near-duplicate tweets which incur substantial redundancy, the EventContext Module reports a selected subset of tweets from the candidate context tweets as an event's context tweets. In order to select the context tweets, the EventCluster Module employs a near-duplicate filter based on a $tf-idf$

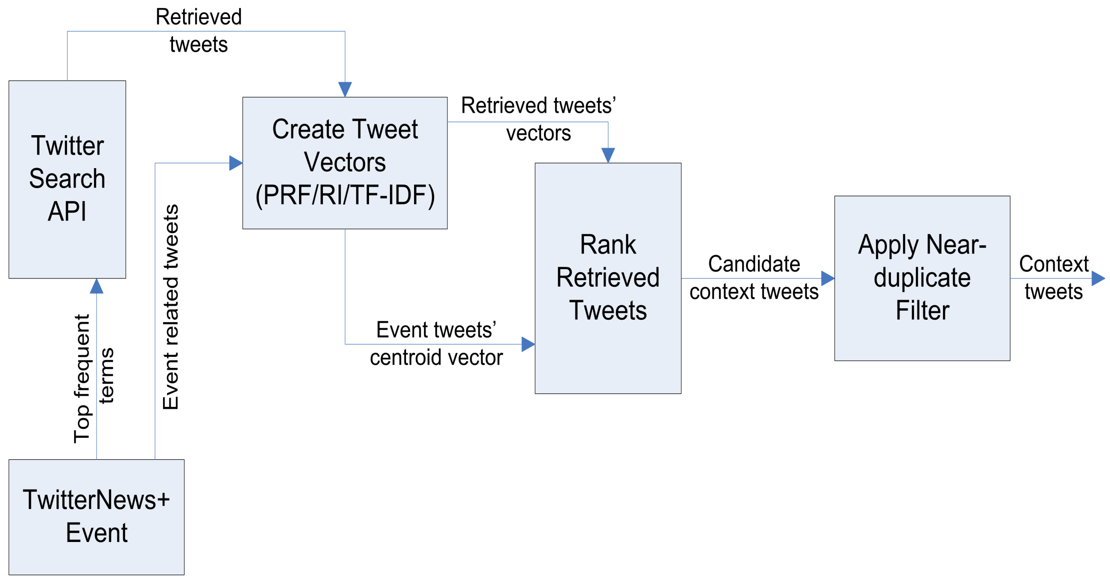


Figure 5.2: Architecture of the EventContext Module

weighted cosine similarity measure. The near-duplicate filter, which utilizes two threshold values, t_{dupE} and t_{dupC}, discards the candidate context tweets that are near-duplicates of the event-tweets or the previously selected context tweets. A candidate context tweet is considered a near-duplicate of the event-tweets, if its cosine similarity with the $tf-idf$ weighted event-tweets' centroid is above the threshold t_{dupE}. Alternatively, if the candidate context tweet's cosine similarity with the centroid of the previously selected context tweets is above the threshold t_{dupC}, then it is considered as a near-duplicate of the context tweets. The top-k ranked context tweets selected from the candidate context tweets of an event are reported by the EvenContext Module where the rank of each context tweet denotes its relevancy with the event.

5.3.1 Probabilistic-feedback-based Tweet Vector (Method-PRF)

In order to calculate the similarity between tweets using the cosine similarity measure, vectors associated with the tweets are utilized. The probabilistic feedback method (Method-PRF) to generate the tweet vectors calculates the weight of the dimensions in a tweet vector based on the term correlation vectors associated with the terms contained in the tweet. Therefore, we utilize two types of vectors: 1) a tweet vector associated with each

Algorithm 6 . TwitterNews+: EventContext Module

```
Require: context threshold ($t_{con}$) and near-duplicate filter thresholds ($t_{dupE}$ and $t_{dupC}$)
    values
$E \leftarrow$ event-related tweets
$R \leftarrow$ tweets retrieved based on $top-k$ terms in $E$
for each tweet $tw$ in $E$ do
    calculate a weighted vector using prf/ri/tf-idf and assign it to the set of
    event-related tweets' vectors $V_E$
end for
for each tweet $tw$ in $R$ do
    calculate a weighted vector using prf/ri/tf-idf and assign it to the set of
    retrieved tweets' vectors $V_R$
end for
calculate the centroid vector $\vec{c_{ev}}$ from $V_E$
for each tweet vector $\vec{tw}$ in $V_R$ do
    $rank_{tw} \leftarrow$ cosine similarity between $\vec{tw}$ and $\vec{c_{ev}}$
    if $rank_{tw} > t_{con}$ then
        assign tweet $tw$ to the set of candidate context tweets $C^{'}$
    end if
end for
sort the tweets in $C^{'}$ based on their ranks in a descending order
for each tweet $tw$ in $C^{'}$ do
    assign $tw$ to the set of context tweets $C$ if it passes the near-duplicate filter
end for
```

tweet in the corpus D which consists of the event tweets and the retrieved tweets, and 2) a term correlation vector associated with each unique term in D.

Information contained in the tweet tw_i of the corpus D is formalized in a weighted tweet vector $\vec{tw_i}$ with L dimensions:

$$\vec{tw_i} = \{d_1, d_2, \cdots, d_L\} \tag{5.3}$$

where, L is the number of unique terms contained in D. Each dimension d_j in the tweet vector $\vec{tw_i}$, where $j = term_1, term_2, \cdots, term_L$, is associated with a unique term j and assigned a weight w_j. The weight w_j is the average correlation weight of the j^{th} dimension of the correlation vectors associated with the terms contained in tw_i.

For each unique term j in D, a correlation vector $\vec{cr}_j$ is created with L dimensions:

$$\vec{cr}_j = \{c_1, c_2, \cdots, c_L\} \tag{5.4}$$

where, each dimension c_k ($k = term_1, term_2, \cdots, term_L$) of the correlation vector $\vec{cr}_j$ is assigned a weight w_k. The weight w_k of the k^{th} dimension represents a degree of correlation between the terms j and k and is calculated using a probabilistic feedback (PRF) mechanism [24, 124], where the number of tweets containing both terms serves as a positive evidence of relationship and the number of tweets containing only one of the terms serves as a negative evidence of relationship:

$$w_k = \log \frac{n_{j,k}/(n_j - n_{j,k})}{(n_k - n_{j,k})/(N - n_k - n_j + n_{j,k})} \times \left|\frac{n_{j,k}}{n_j} - \frac{n_k - n_{j,k}}{N - n_j}\right| \tag{5.5}$$

where, $n_{j,k}$ is the number of tweets in D containing the terms j and k, n_j is the number of tweets containing the term j, n_k is the number of tweets containing the term k, and $N = |D|$ is the number of tweets in D.

5.3.2 Random-indexing-based Tweet Vector (Method-RI)

The Random-indexing-based Method (Method-RI) to generate tweet vectors utilizes random indexing [9], which is a term co-occurrence based term vector model to measure semantic similarity. The RI-based term vector model associates each term with two vectors: an index vector and a context vector. The index vector is a sparse, high dimensional, and unique random vector. The index vectors for each term have the same dimension. The context vector is initialized with zeroes and its size is equivalent to that of the index vector. The context vector of a term is updated by adding the index vector of another term when it co-occurs with the other term in a sliding context window. We have used the S-Space Package [111] with a default parameter setting to process all the unique terms in the corpus D and produce a context vector for each term based on term co-occurrence. For each tweet tw_i in D, a tweet vector is then calculated based on the context vectors of the terms contained in the tweet, where the weight of the tweet vector's j^{th} dimension is the average weight of the j^{th} dimension of the context vectors associated with the terms contained in tw_i.

5.3.3 TF-IDF-based Tweet Vector (Method-TFIDF)

The $tf-idf$-based method (Method-TFIDF) to create tweet vectors works as a baseline method where each tweet in the corpus D is associated with a vector. Information contained in the tweet tw_i of the corpus D is formalized in a weighted tweet vector $\vec{tw}_i$ with L dimensions:

$$\vec{tw}_i = \{d_1, d_2, \cdots, d_L\} \tag{5.6}$$

where, L is the number of unique terms contained in D. Each dimension d_j in the tweet vector $\vec{tw}_i$, where $j = term_1, term_2, \cdots, term_L$, is associated with a unique term j and assigned a $tf-idf$-based weight if the term j is present in the tweet:

$$d_j = \begin{cases} tf-idf(j, tw_i, D) & \text{if } tw_i \text{ contains the term } j \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

The following formula is used to calculate $tf-idf(j, tw_i, D)$:

$$tf-idf(j, tw_i, D) = tf(j, tw_i) \times idf(j, D) \tag{5.8}$$

where, $tf(j, tw_i)$ is the number of times the term j is found in tw_i, and

$$idf(j, D) = \log \frac{N}{|\{tw_i \in D : j \in D\}|} \tag{5.9}$$

where, N is the total number of tweets in D, and $|\{tw_i \in D : j \in D\}|$ is the total number of tweets in D processed so far in which the term j appears.

5.4 Experiment Results and Evaluation

In order to evaluate the effectiveness of the three different methods to create tweet vectors, used in the EventContext Module, we ran *TwitterNews+* on the live Twitter data stream (1%) using the Twitter Streaming API for the first two months of the year 2018, and selected twenty events with diverse topics, detected by the event detection system, as ground truth. The topics and descriptions of the ground truth events, as shown in Table 5.1, have been generated by two human evaluators based on a manual inspection of the event-related tweets.

Table 5.1: The selected ground truth events from the first two months of the year, 2018

Event Topic	Event Description
Sanchi oil tanker	An Iranian oil tanker explodes and sinks off the coast of China.
Todd Haley	Todd Haley's contract as the offensive coordinator of the Pittsburgh Steelers will not be renewed.
Russian plane crash	A Saratov Airlines plane crashes near Moscow.
Brexit	A debate on the issue of Brexit.
Asma Jahangir	Death of a famous human rights lawyer and social activist.
Benjamin Netanyahu	Israel's prime minister says his military has dealt "severe blows" to Iran and Syria with a series of airstrikes.
Oxfam	Ministers could cut off funding for the UK based charity, Oxfam, warned by the international development secretary Penny Mordaunt.
Alexis Sanchez	A deal between Alexis Sanchez and the English football club Manchester United.
Barnaby Joyce	A petition demanding the removal of Barnaby Joyce as Deputy Prime Minister and House of Representatives MP for New England.
Boko Haram	The Nigerian army destroys a Boko Haram camp in Sambisa.
Winter Olympics	Kaitlyn Lawes and John Morris have won gold for Canada in mixed doubles curling at the Winter Olympics, 2018.
Middle-East conflict	The Syrian government warns that Israel would face "more surprises" in future attacks on Syria's territory.
Royal wedding	Public appearance of Prince Harry and Meghan Markle at Edinburgh following their engagement announcement.
Kim Jong Un	Kim Jong Un invites South Korean president to North Korea.
Indo-Pakistani conflict	Defence Minister Nirmala Sithraraman warns Pakistan in the wake of a deadly militant attack on an Indian army camp.
Poacher	Lions maul suspected lion poacher to death in South Africa.

Table 5.1. continued.

Event Topic	Event Description
Iran nuclear deal	Iranian president Hassan Rouhani made a remark regarding the involvement of US president Donald Trump in Iran nuclear deal.
Ryan Mason	English footballer Ryan Mason has been forced to retire due to fractured skull.
Donald Trump	US president Donald Trump made a derogatory comment about Haiti and Africa.
Jacob Zuma	The African National Congress (ANC) party has decided to recall president Jacob Zuma.

Once a ground truth event was detected by *TwitterNews+* from the underlying Twitter streaming endpoint, the EvenContext Module was activated which formed a search query based on the most frequent proper and common nouns found in the event-tweets and used the Twitter Search API to retrieve around 5000 tweets depending on the availability of the tweets related to the query. The EventContext Module was configured to produce three sets of the top-30 ranked context tweets from the retrieved tweets for each ground truth event. The three sets of the context tweets per event were produced by Method-PRF, Method-RI, and the baseline Method-TFIDF. A less restrictive parameter setting is used, as shown in Table 5.2, to ensure that at least thirty context tweets per method are available to facilitate evaluation. We have provided the sixty sets of the top-30 ranked context tweets associated with the twenty ground truth events to the human evaluators who helped in generating Table 5.1 and asked them to assess the relevancy of each context tweet with respect to the event. A context tweet is considered relevant with respect to an event, if it provides any insight into the event. For example, the tweet “The Iranian oil tanker #Sanchi sank last Sunday after a week of burning. Only 3 bodies of the 32 missing sailors ha.. https://t.co/kHEVP12Xud” is considered relevant to the “Sanchi oil tanker” event as it provides additional information about the event, whereas the tweet “in october 2008, please see that your daughter is off the coast of an island in the north china sea. lian yu” is considered as irrelevant to the event. Based on the evaluators’ assessment of the context tweets for each ground truth, the tweet vector creation method that produces the maximum percentage of relevant tweets as context tweets per event, is selected to be integrated as part of the EventContext Module.

Table 5.2: The parameter settings used in Algorithm 6, depending on the method to create tweet vectors being used

	Parameter Setting
Method-PRF	$t_{con} = 0.30$ $t_{dupE} = 0.30$ $t_{dupC} = 0.40$
Method-RI	$t_{con} = 0.30$ $t_{dupE} = 0.25$ $t_{dupC} = 0.40$
Method-TFIDF	$t_{con} = 0.10$ $t_{dupE} = 0.25$ $t_{dupC} = 0.25$

In Table 5.3, the number of relevant tweets in the top-10, top-20, and top-30 ranked context tweets of each ground truth event, as assessed by the evaluators, is shown.

Table 5.3: The number of the relevant tweets in the top-k ranked context tweets

Event Topic	Number of relevant tweets in top-k ranked context tweets								
	Method-PRF			Method-RI			Method-TFIDF		
	k=10	*k=20*	*k=30*	*k=10*	*k=20*	*k=30*	*k=10*	*k=20*	*k=30*
Sanchi oil tanker	9	19	29	10	17	23	6	12	19
Todd Haley	10	19	27	8	13	18	5	11	15
Russian plane crash	10	20	29	7	15	26	7	14	22
Brexit	10	18	28	9	19	28	9	19	24
Asma Jahangir	10	20	30	10	19	29	10	20	29
Benjamin Netanyahu	9	16	26	9	19	27	9	18	23
Oxfam	6	15	20	5	6	7	6	9	12
Alexis Sanchez	9	19	24	10	20	30	10	20	30
Barnaby Joyce	7	11	15	7	9	9	2	4	4
Boko Haram	8	10	11	1	4	10	5	7	7
Winter Olympics	10	20	25	10	14	15	7	12	16
Middle-East conflict	9	17	22	8	13	20	7	11	18
Royal wedding	10	19	28	10	20	30	8	12	14
Kim Jong Un	10	19	29	10	19	28	8	16	23
Indo-Pakistani conflict	9	19	26	10	18	21	10	18	25
Poacher	10	20	28	3	5	8	6	9	17
Iran nuclear deal	7	12	17	5	9	14	6	7	7
Ryan Mason	10	17	23	6	8	11	7	10	10
Donald Trump	10	18	27	10	20	27	8	18	23
Jacob Zuma	10	20	30	10	20	30	10	19	27

The performance of Method-PRF, Method-RI, and Method-TFIDF is summarized in Table 5.4, which reveals that Method-PRF outperforms the other two methods as it yields the maximum percentage of relevant tweets out of the top-10, top-20, and top-30 context tweets per ground truth event.

Table 5.4: The percentage of the relevant tweets in the top-k ranked context tweets

	k=10	k=20	k=30
Method-PRF	91.50	87.00	82.33
Method-RI	79.00	71.75	68.50
Method-TFIDF	73.00	66.50	60.83

In order to produce relevant context tweets for an event using the EventContext Module, it is an intuitive requirement that the event has to be a major event that results in a large volume of tweets being posted on Twitter. In other words, there have to be enough tweets about an event available on Twitter for the EventContext Module to do its job, which is to provide additional information on the event. It can be seen in Table 5.3 that the EventContext Module performed poorly regardless of the method being used in selecting relevant context tweets for the events “Oxfam”, “Barnaby Joyce”, “Boko Haram”, and “Iran nuclear deal”. Upon further inspection, we found that these events were minor events for which only a small number of related tweets were available in their associated set of retrieved tweets.

The results of our experiments led us to integrate the Method-PRF to create tweet vectors in the EventCluster Module as it provides the most relevant context tweets compared to the other two methods. Table 5.5 shows the selected context tweets from the top-30 context tweets generated by the EventCluster Module, integrated with Method-PRF, for three ground truth events.

Table 5.5: The selected context tweets for three ground truth events based on Method-PRF

Event Topic	Selected Context Tweet
Sanchi oil tanker	East China Sea oil tanker disaster: what it means for the environment #eastchinaoildisaster #OILSpill https://t.co/z3SRxJ7WPu
	Oil from sunken Iranian tanker spreads over 100 sq km of East China Sea: Guardian https://t.co/BlGklvO1un
	China to send in divers to plug oil leaks from Iranian tanker wreck https://t.co/PgtA4C752C
	An Iranian oil tanker sank in a commercial fishing area in the East China Sea over the weekend, potentially unleash..https://t.co/Hrxd5sbuWq
	Sunken tanker Sanchi: Four oil slicks seen, says China - BBC News https://t.co/CmLDX19o4d
Barnaby Joyce	Barnaby Joyce is a dead man walking. His political career is over. Gone by the end of the month, if not sooner.
	Do you think Barnaby Joyce and other cabinet members have acted corruptly during his affair? #BarnabyJoyce #BeetRooter #auspol
	Bye bye Barnaby? Government sources say Joyce's position is 'untenable' and he 'should go' https://t.co/sGVT0kDHuo
	Phil Coorey writes that Nationals MPs are beginning to question whether Barnaby Joyce can survive as leader..https://t.co/I148NIIPs4
	Let us now recall the glowing way Turnbull endorsed Barnaby during by election - if I recall correctly, naming him..https://t.co/XClurTIC9N
Jacob Zuma	#ZumaExit #ANCNEC Magashule has confirmed that President Jacob Zuma will respond tomorrow to the party's decision.. https://t.co/8DbBL9KlBm
	ANC NEC trying to get direction from Cyril Ramaphosa on the future of Jacob Zuma. #ANCNEC #ANC #ZumaExit https://t.co/8Yx7TzKxJh
	If the ANC wants to remove Zuma and he refuses then the ANC must do to Zuma what ZanuPF did to Mugabe Period. #ZumaExit #ZumaRecall #ANCNEC
	Basically, ANC is replacing Zuma with another Zuma, probably even worse. #ZumaRecall #ANCNEC https://t.co/MphCaBoebQ
	President Zuma has asked the ANC to give him three to six months to step down. #ZumaExit https://t.co/oTf4xpn0Zk

During our evaluation process, we found that the context tweets produced by each method to create tweet vectors contain some tweets that are unique to the method being used. For example, the context tweet "Evocative image of China Coast Guard cutters deployed to the scene of the sinking Sanchi. Despite all these new shi.." is only selected by Method-TFIDF as one of the top-30 context tweets for the "Sanchi oil tanker" event, as

Table 5.6: The selected context tweets for the ground truth event “Sanchi oil tanker” based on the three methods to create tweet vectors

Method	Selected Context Tweet
Method-PRF	East China Sea oil tanker disaster: what it means for the environment #eastchinaoildisaster #OILSpill https://t.co/z3SRxJ7WPu
	Oil from sunken Iranian tanker spreads over 100 sq km of East China Sea: Guardian https://t.co/BlGklvO1un
	China to send in divers to plug oil leaks from Iranian tanker wreck https://t.co/PgtA4C752C
	An Iranian oil tanker sank in a commercial fishing area in the East China Sea over the weekend, potentially unleash..https://t.co/Hrxd5sbuWq
	Sunken tanker Sanchi: Four oil slicks seen, says China - BBC News https://t.co/CmLDX19o4d
Method-RI	The Iranian oil tanker #Sanchi sank last Sunday after a week of burning. Only 3 bodies of the 32 missing sailors ha.. https://t.co/kHEVP12Xud
	Chinese State Oceanic Administration this am says there are 4 slicks from the sunken tanker Sanchi covering 101.. https://t.co/ho6XHkSVw9
	The real catastrophe happened before the tanker #Sanchi sank. For oil companies ecological desaster is cheaper than.. https://t.co/KDT9mRfgyv
	Iranian #Oil Tanker Fire Major #Environmental Catastrophe in Waters off #Shanghai, #China, Even Though Area Has Bee.. https://t.co/zs7rn6BIGZ
	It started as a collision out at sea. How did it become an oil spill the size of Paris? https://t.co/LC5Kz1ItYQ
Method-TFIDF	Just learnt that Ehsan Aboli, one of the 30 Iranian crew who died at the sunken #Sanchi tanker off China coast, was.. https://t.co/v1FnDEEUSs
	China says Iranian oil tanker wreck located https://t.co/PunxtqYu3
	Likely impact of oil tanker explosion at East China Sea https://t.co/sY7MAZ5PaU
	Evocative image of China Coast Guard cutters deployed to the scene of the sinking Sanchi. Despite all these new shi.. https://t.co/6lE8VG7gL3
	The oil spill area increased to 101 square km from the sinking site of the Iranian oil tanker Sanchi in the East Ch.. https://t.co/h4fn3KNE2m

shown in Table 5.6. Similarly, the tweet “China to send in divers to plug oil leaks from Iranian tanker wreck https://t.co/PgtA4C752C” is selected as one of the top-30 context tweets by all the methods except Method-RI. Therefore, as a future work we can explore additional high-performing methods, besides Method-PRF, to create tweet vectors so that more than one of these methods can be used simultaneously by the EventContext Module to generate the context tweets for an event. Combining the context tweets generated by two or more of such high-performing methods can lead to an increased amount of information in the context tweets, compared to the information obtained by using a single high-performing method.

5.5 Conclusion

As the Twitter Search API provides a historical access to the previous one week of the Twitter content, it can be utilized to retrieve additional tweets about an event which is detected by an event detection system. A subset of the retrieved tweets that provides additional information about the event can serve as a context to the event. In order to provide a context to an event from the tweets retrieved using the Twitter Search API, we have proposed and implemented the EventContext Module, the third major component of the *TwitterNews+* event detection system.

The EventContext Module operates independently of the event detection process of *TwitterNews+*, which means any event detection system can use it to gain additional insight on an event. Once an event is reported by an event detection system, the EventContext Module formulates a search query consisting of the most frequent proper and common nouns found in the event-tweets and uses the Twitter Search API to retrieve additional tweets. The EventContext Module then selects a subset of the highly relevant tweets from the retrieved tweets which serves as a context to the event.

Finally, the EventContext Module employs a near-duplicate filter to select the context tweets for an event from the candidate context tweets generated in the previous stage of the module's operation. As a number of the candidate context tweets are often redundant due to the presence of the tweets that are near-duplicates of the event-tweets or the retrieved tweets, the near-duplicate filter helps in discarding the redundant tweets.

We have experimented with the EventContext Module using three different methods: Method-PRF, Method-RI, and Method-TFIDF to create tweet vectors. Our experiments revealed that the EventCluster Module integrated with Method-PRF outperformed the other two methods by having the maximum percentage of relevant tweets in the top-10, top-20, and top-30 ranked context tweets per ground truth event.

6

Conclusion

In this thesis, we have proposed and implemented a novel end-to-end Twitter-centric event detection framework, *TwitterNews+*, which is capable of detecting newsworthy events in real-time from the Twitter data stream. The starting point of our work was to perform a survey on the existing literature on various event detection techniques. We have conducted a survey [1], disccussed in Chapter 2, on the noteworthy and recent event detection techniques that focus on detecting real world events of global and/or local interest from the Twitter data stream, and broadly categorized different approaches based on term interestingness, topic modelling, incremental clustering, and hybrid methods. Term-interestingness-based approaches usually differ on the term selection methods they employ, as well as on the way in which term correlations are computed and changes in the term correlations are tracked. The approaches based on term interestingness can often capture misleading term correlations, and measuring the term correlations can be computationally prohibitive in an online setting. Similarly, the topic-modelling-based approaches usually incur too high of a computational cost to be effective in a streaming setting. Incremental-clustering-based approaches, despite having low computational cost, are prone to fragmentation where the same event is detected multiple times as a new event.

We identified that not all the approaches found in the literature may be applicable in real-world applications. This is mostly due to the scalability issue as some event detection methods are not equipped to handle the high volume of streaming data. Moreover, the use of a clustering approach that requires the total number of clusters to be fixed is often not useful for a streaming data environment, where a wide variety of topics are discussed, thus making it difficult to predict the total number of expected event clusters in advance. Based on our findings from the survey, a novel variant of the incremental-clustering-based approach has been utilized in our event detection system, because of its inherently low computational complexity compared to the most of the state-of-the art approaches. Incremental clustering is also suitable as it does not require to have a fixed number of clusters, which is a requirement while dealing with a streaming data environment.

The initial proposed system, *TwitterNews* [118], combines the Locality Sensitive Hashing [8] scheme with a random-indexing-based term vector model [9], to provide a novel incremental-clustering-based solution to detect events from the Twitter data stream. We took an approach in *TwitterNews*, discussed in Chapter 3, where the problem of event detection from the Twitter data stream is divided into two major stages. The first stage involves detecting a burst in the number of tweets discussing an event and the second stage involves clustering the tweets that discuss the same event.

Accordingly, *TwitterNews*'s operation was divided into two major stages. After the pre-processing of a tweet, the first stage is responsible for determining the novelty of a tweet. *TwitterNews* maintains a fixed number of continuously updated tweets from the Twitter Streaming API, stored in the local machine in which it is running. If the input tweet is textually similar to a tweet stored by our system, then it means a soft tweet burst related to an event has occurred and the output of the first stage declares the input tweet to be "*not unique*".

The operation in the first stage of *TwitterNews* is implemented by combining a random-indexing-based (RI) term vector model [9] with the Locality Sensitive Hashing (LSH) scheme [8], to determine whether a tweet is "*not unique*". This is a novel approach that combines RI with LSH to reduce the time needed to determine the novelty of a tweet. Subsequently the second stage, implemented using a novel approach by adapting the generic incremental clustering algorithm, deals with generating the candidate event clusters by incrementally clustering the tweets that were determined as bursty during the first stage. The second stage also incorporates a defragmentation strategy to deal with the fragmented events that were generated when a particular event is detected multiple times as a new event. At the end of the second stage, a set of filters are applied so that *TwitterNews* reports only the candidate events as newsworthy events which can pass through the filters. From our experiments we found that *TwitterNews* performs reasonably well in detecting newsworthy events but does not scale well in a streaming setting and further work in the filtering process is required to improve its precision in detecting events.

Based on the experience gathered from developing *TwitterNews*, we were able to design an efficient solution to the problem of event detection which resulted in the implementation of *TwitterNews+* [114, 123]. Our proposed system, *TwitterNews+*, discussed in Chapter 4, incorporates a novel variant of the incremental clustering approach to provide a low computational cost solution to the problem of event detection in real-time from the Twitter data stream and operates in two major stages similar to *TwitterNews*. However, both stages of *TwitterNews+* use more efficient algorithms than *TwitterNews* and utilize specialized inverted indices to address the former system's scalability issue.

We have performed a parameter sensitivity analysis on the different parameters of *TwitterNews+*, for which we have utilized a 'local' sensitivity analysis where one parameter at a time is repeatedly varied while keeping the others fixed. A sensitivity ranking for each parameter is obtained by varying its value while leaving all the other parameters constant, and quantifying the change in terms of recall and precision, calculated from the newsworthy events reported by *TwitterNews+*. The parameter sensitivity analysis gave us the optimal parameter settings for our system to improve its performance in terms of recall and precision. We then investigated the performance of *TwitterNews+*

and five state-of-the-art baselines that cover a wide range of event detection techniques. Our experiments revealed that *TwitterNews+* outperforms the baselines by achieving the highest recall and precision in detecting newsworthy events.

TwitterNews+ is an efficient system to detect events from the Twitter data stream in real-time with a high *number-of-tweets/second* processing capability, it maintains a constant space and processing time, and achieves very good results compared to the baselines used in our experiments. The different set of filters used in *TwitterNews+*'s post-processing phase helps in retaining both major and minor newsworthy events while discarding a significant amount of trivial events in order to improve the precision of the reported events. *TwitterNews+*'s capability to retain both major and minor events based on a soft burst detection approach, is a feature which is lacking in most of the state-of-the-art approaches as they focus on detecting events based on a burst detection approach that requires a substantial burst in the volume of tweets discussing a topic. Moreover, the evaluation of *TwitterNews+*, done using a publicly available corpus, will allow researchers to compare different systems fairly against our system.

An event, which is detected by an event detection system, often consists of tweets which capture a partial story, leading to a lack of sufficient insight into the detected event. The problem of providing a context to an event is to provide additional information to supplement the information contained in the event-tweets. We have utilized the Twitter Search API, which provides a historical access to the previous one week of the Twitter content, to fetch additional tweets about an event. In order to provide a context to an event from the tweets retrieved using the Twitter Search API, we have designed a novel approach which is implemented as the EventContext Module of *TwitterNews+* and discussed in Chapter 5. The EventContext Module operates independently of the event detection process of *TwitterNews+* and thus, any event detection system can use it to gain additional insight on an event.

Once an event is reported by an event detection system, the EventContext Module formulates a search query consisting of the most frequent and informative terms (i.e., proper and common nouns) found in the event-tweets, to retrieve additional tweets through the Twitter Search API containing at least one of the terms used in the query.

The EventContext Module then selects a subset of the highly relevant tweets from the retrieved tweets which serves as a context to the event. The EventContext Module also employs a near-duplicate filter to discard redundant tweets from the candidate context tweets generated in the second-to-last stage of the module's operation and reports the remaining tweets, in descending order of their relevancy ranking, as context tweets. The relevancy of a tweet is determined based on the cosine similarity measure which requires the information contained in a tweet to be represented using a vector. We have experimented on the EventContext Module using three different methods: Method-PRF, Method-RI, and Method-TFIDF to create tweet vectors. Our experiments revealed that the EventCluster Module integrated with Method-PRF outperformed the other two methods by having the maximum percentage of relevant tweets in the context tweets produced per ground truth event.

As the focus of our thesis was design and development of an event detection system that can detect events in real-time, the literature review was also mostly focused on systems with real-time event detection capability. We believe an extensive classification of event detection systems is out of scope of this thesis. However, in future, an extensive classification of event detection systems can prove to be useful for further analysis.

Based on the study conducted on real-time event detection systems, we have chosen the commonly used cosine similarity measure to determine the textual similarity between tweets and an incremental-clustering-based approach to cluster similar tweets. In future, an extensive evaluation of the available similarity measures and clustering approaches can be useful.

A future avenue of work in *TwitterNews+* is to investigate various Twitter-centric spam detection methods and design a novel approach that can be used in the pre-processing phase of our system to achieve higher precision in detecting newsworthy events. However, this is a big challenge to overcome, which requires providing a low computational cost solution to detect spam tweets in Twitter and making sure that our system, or for that matter any event detection system, remains capable of detecting events in real-time from the Twitter data stream despite the additional computational cost incurred in the pre-processing phase.

An independent component, based on a machine learning model, can be incorporated in *TwitterNews+* to identify specific types of events (e.g., politics, security, disaster, etc.) from the reported newsworthy events. Furthermore, we can explore additional high-performing methods, besides Method-PRF, to create tweet vectors so that more than one of these methods can be used simultaneously by the EventContext Module to generate the context tweets for an event. Combining the context tweets generated by two or more of such high-performing methods can lead to an increased amount of information in the context tweets, compared to the information obtained by using a single high-performing method.

References

[1] M. Hasan, M. A. Orgun, and R. Schwitter. *A survey on real-time event detection from the Twitter data stream*. Journal of Information Science (2017). SAGE Publications, URL `http://dx.doi.org/10.1177/0165551517698564`. 1, 3, 4, 65, 79, 85, 101

[2] F. Atefeh and W. Khreich. *A survey of techniques for event detection in Twitter*. Computational Intelligence **31**(1), 132 (2015). Wiley Online Library. 1, 3, 7, 8, 35, 36

[3] W. Dou, K. Wang, W. Ribarsky, and M. Zhou. *Event detection in social media data*. In *Proceedings of the IEEE VisWeek Workshop on Interactive Visual Text Analytics - Task Driven Analytics of Social Media Content*, pp. 971–980 (2012). 2

[4] J. Allan. *Topic detection and tracking: event-based information organization*, vol. 12 (Springer Science & Business Media, 2012). 2

[5] H. Becker, M. Naaman, and L. Gravano. *Beyond trending topics: Real-world event identification on Twitter.* In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media, ICWSM* (2011). 2, 3, 22, 23, 24, 29, 31, 34

[6] S. Petrovic, M. Osborne, R. McCreadie, C. Macdonald, and I. Ounis. *Can Twitter replace newswire for breaking news?* In *Proceedings of the International AAAI Conference On Web and Social Media, ICWSM* (2013). 2, 3

[7] M. Osborne and M. Dredze. *Facebook, Twitter and Google Plus for breaking news: Is there a winner?* In *Proceedings of the International AAAI Conference On Web and Social Media, ICWSM* (2014). 2, 3

[8] S. Petrović, M. Osborne, and V. Lavrenko. *Streaming first story detection with application to Twitter*. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, HLT '10, pp. 181–189 (ACL, Stroudsburg, PA, USA, 2010). 6, 21, 22, 29, 31, 34, 38, 39, 40, 41, 43, 46, 47, 48, 51, 61, 68, 72, 77, 78, 102, 103

[9] M. Sahlgren. *An introduction to random indexing*. In *Proceedings of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE*, vol. 5 (2005). 6, 38, 39, 40, 41, 43, 85, 91, 102, 103

[10] C. Li, A. Sun, and A. Datta. *Twevent: Segment-based event detection from tweets*. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, CIKM '12, pp. 155–164 (ACM, 2012). 7, 9, 10, 12, 29, 30, 31

[11] S. Gaglio, G. L. Re, and M. Morana. *A framework for real-time Twitter data analysis*. Computer Communications **73**, 236 (2016). Elsevier. 7, 10, 13, 29, 30

[12] G. Stilo and P. Velardi. *Efficient temporal mining of micro-blog texts and its application to event discovery*. Data Mining and Knowledge Discovery pp. 1–31 (2015). Springer. 7, 11, 15, 29, 30, 33

[13] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang. *TopicSketch: Real-time bursty topic detection from Twitter*. In *Proceedings of the IEEE 13th International Conference on Data Mining, ICDM*, pp. 837–846 (IEEE, 2013). 7, 18, 31

[14] D. Zhou, L. Chen, and Y. He. *An unsupervised framework of exploring events on Twitter: Filtering, extraction and categorization*. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2468–2475 (2015). 7, 18, 29, 31, 32, 33

[15] A. J. McMinn and J. M. Jose. *Real-time entity-based event detection for Twitter*. In *Proceedings of the Experimental IR Meets Multilinguality, Multimodality, and Interaction: 6th International Conference of the CLEF Association*, CLEF '15, pp. 65–77 (Springer, 2015). 7, 22, 24, 25, 29, 31, 32, 34, 48, 50, 63, 66, 68, 72

[16] C. De Boom, S. Van Canneyt, and B. Dhoedt. *Semantics-driven event clustering in Twitter feeds*. In *Proceedings of the 5th Workshop on Making Sense of Microposts*, vol. 1395, pp. 2–9 (CEUR, 2015). 7, 22, 23, 29, 30, 34

[17] A. Guille and C. Favre. *Event detection, tracking, and visualization in Twitter: A mention-anomaly-based approach*. Social Network Analysis and Mining **5**(1), 1 (2015). Springer. 7, 27, 29, 31, 77, 78

[18] R. A. Jarvis and E. A. Patrick. *Clustering using a similarity measure based on shared near neighbors*. IEEE Transactions on Computers **100**(11), 1025 (1973). IEEE. 9, 10

[19] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller. *TwitInfo: Aggregating and visualizing microblogs for event exploration*. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pp. 227–236 (ACM, New York, NY, USA, 2011). 10, 12, 19, 29, 30

[20] M. Mathioudakis and N. Koudas. *TwitterMonitor: Trend detection over the Twitter stream*. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pp. 1155–1158 (ACM, New York, NY, USA, 2010). 10, 12, 14, 29, 30, 31, 35

[21] F. Alvanaki, M. Sebastian, K. Ramamritham, and G. Weikum. *Enblogue: Emergent topic detection in web 2.0 streams*. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pp. 1271–1274 (ACM, New York, NY, USA, 2011). 10, 13, 29, 30

[22] G. Petkos, S. Papadopoulos, L. Aiello, R. Skraba, and Y. Kompatsiaris. *A soft frequent pattern mining approach for textual topic detection*. In *Proceedings of the*

4th International Conference on Web Intelligence, Mining and Semantics, WIMS, pp. 25:1–25:10 (ACM, 2014). 10, 13

[23] S. Gaglio, G. Lo Re, and M. Morana. *Real-time detection of Twitter social events from the user's perspective*. In *Proceedings of the IEEE International Conference on Communications, ICC*, pp. 1207–1212 (IEEE, 2015). 10

[24] M. Cataldi, L. Di Caro, and C. Schifanella. *Emerging topic detection on Twitter based on temporal and social terms evaluation*. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, pp. 4:1–4:10 (ACM, New York, NY, USA, 2010). 11, 14, 34, 91

[25] J. Lin, E. Keogh, L. Wei, and S. Lonardi. *Experiencing SAX: A novel symbolic representation of time series*. Data Mining and Knowledge Discovery **15**(2), 107 (2007). Springer. 11, 15

[26] R. Parikh and K. Karlapalem. *ET: Events from tweets*. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion, pp. 613–620 (ACM, New York, NY, USA, 2013). 11, 15, 29, 31

[27] J. Weng and B.-S. Lee. *Event detection in Twitter.* In *Proceedings of the International AAAI Conference on Web and Social Media, ICWSM*, vol. 11, pp. 401–408 (2011). 11, 16, 29, 30, 34

[28] X. Zhang, X. Chen, Y. Chen, S. Wang, Z. Li, and J. Xia. *Event detection and popularity prediction in microblogging*. Neurocomputing **149**, 1469 (2015). Elsevier. 11, 16, 17, 34

[29] G. Salton and C. Buckley. *Term-weighting approaches in automatic text retrieval*. Information Processing & Management **24**(5), 513 (1988). Elsevier. 11, 14, 16, 23, 25

[30] *IETF network working group. rfc 2988: Computing TCP's retransmission timer*. `http://tools.ietf.org/html/rfc2988` (2015). 12

[31] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. *Indexing by latent semantic analysis*. Journal of the American Society for Information Science **41**(6), 391 (1990). American Documentation Institute. 13

[32] J. R. Finkel, T. Grenager, and C. Manning. *Incorporating non-local information into information extraction systems by Gibbs sampling*. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 363–370 (ACL, 2005). 14

[33] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank citation ranking: bringing order to the web*. Technical report, Stanford InfoLab (1999). 14, 16

[34] G. Kaiser. *A Friendly Guide to Wavelets* (Birkhauser Boston Inc., Cambridge, MA, USA, 1994). 16

[35] A. Ritter, Mausam, O. Etzioni, and S. Clark. *Open domain event extraction from Twitter*. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pp. 1104–1112 (ACM, New York, NY, USA, 2012). 17, 18, 29, 31, 32

[36] A. Ritter, S. Clark, Mausam, and O. Etzioni. *Named entity recognition in tweets: An experimental study*. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pp. 1524–1534 (ACL, Stroudsburg, PA, USA, 2011). 17, 32

[37] I. Mani and G. Wilson. *Robust temporal processing of news*. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pp. 69–76 (ACL, Stroudsburg, PA, USA, 2000). 17, 32

[38] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. *Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data*. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 282–289 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001). 17

[39] Y. You, G. Huang, J. Cao, E. Chen, J. He, Y. Zhang, and L. Hu. *GEAM: A general*

and event-related aspects model for Twitter event detection. In *Proceedings of the Web Information Systems Engineering*, WISE '13, pp. 319–332 (Springer, 2013). 18, 32

[40] R. Xie, F. Zhu, H. Ma, W. Xie, and C. Lin. *CLEar: A real-time online observatory for bursty and viral events*. Proceedings of the VLDB Endowment **7**(13), 1637 (2014). VLDB Endowment. 18, 31

[41] J. Li, J. Wen, Z. Tai, R. Zhang, and W. Yu. *Bursty event detection from microblog: A distributed and incremental approach*. Concurrency and Computation: Practice and Experience (2015). Wiley Online Library. 19, 29, 31

[42] H. Cai, Y. Yang, X. Li, and Z. Huang. *What are popular: Exploring Twitter features for event detection, tracking and visualization*. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pp. 89–98 (ACM, 2015). 19

[43] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. *Caffe: Convolutional architecture for fast feature embedding*. In *Proceedings of the ACM International Conference on Multimedia*, pp. 675–678 (ACM, 2014). 20

[44] D. Shepard. *Nonparametric Bayes Pachinko allocation for super-event detection in Twitter*. In *Proceedings of the IEEE Region 10 Conference, TENCON*, pp. 1–5 (IEEE, 2014). 20

[45] Q. Diao and J. Jiang. *A unified model for topics, events and users on Twitter*. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1869–1879 (ACL, 2013). 20

[46] W. Li, D. Blei, and A. McCallum. *Nonparametric Bayes Pachinko allocation*. arXiv preprint arXiv:1206.5270 (2012). 20

[47] A. Madani, O. Boussaid, and D. E. Zegour. *Real-time trending topics detection and description from Twitter content*. Social Network Analysis and Mining **5**(1), 1 (2015). Springer. 20, 32, 86

[48] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. *Hierarchical Dirichlet processes*. Journal of the American Statistical Association (2012). Taylor & Francis. 20

[49] M. Osborne, S. Petrovic, R. McCreadie, C. Macdonald, and I. Ounis. *Bieber no more: First story detection using Twitter and Wikipedia*. In *SIGIR Workshop on Time-aware Information Access* (ACM, 2012). 21

[50] M. Osborne, S. Moran, R. McCreadie, A. Von Lunen, M. D. Sykora, E. Cano, N. Ireson, C. Macdonald, I. Ounis, Y. He, *et al.* *Real-time detection, tracking, and monitoring of automatically discovered events in social media*. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (Association for Computational Linguistics/© The Authors, 2014). 21, 22, 34, 87

[51] A. E. Cano Basave, Y. He, K. Liu, and J. Zhao. *A weakly supervised Bayesian model for violence detection in social media*. In *Proceedings of the International Joint Conference on Natural Language Processing, IJCNLP* (Asian Federation of Natural Language Processing, 2013). 21, 22

[52] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. *LIBLINEAR: A library for large linear classification*. Journal of Machine Learning Research **9**, 1871 (2008). JMLR.org. 22

[53] S. Phuvipadawat and T. Murata. *Breaking news detection and tracking in Twitter*. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3 of *WI-IAT '10*, pp. 120–123 (IEEE Computer Society, Washington, DC, USA, 2010). 22, 24, 34

[54] S. Unankard, X. Li, and M. A. Sharaf. *Emerging event detection in social networks with location sensitivity*. World Wide Web **18**(5), 1393 (2015). Springer. 23, 25, 29, 30, 32, 34

[55] R. O. Duda, P. E. Hart, *et al.* *Pattern classification and scene analysis*, vol. 3 (Wiley, New York, USA, 1973). 23, 25

[56] S. B. Kaleel and A. Abhari. *Cluster-discovery of Twitter messages for event detection and trending*. Journal of Computational Science **6**, 47 (2015). Elsevier. 23, 26

[57] C.-H. Lee and T.-F. Chien. *Leveraging microblogging big data with a modified density-based clustering approach for event awareness and topic ranking*. Journal of Information Science **39**, 523 (2013). SAGE Publications. 23, 26

[58] C.-H. Lee. *Mining spatio-temporal information on microblogging streams using a density-based online clustering method*. Expert Systems with Applications **39**(10), 9623 (2012). Elsevier. 23, 26

[59] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. *Incremental clustering for mining in a data warehousing environment*. In *Proceedings of the 24th International Conference on Very Large Databases*, VLDB '98, pp. 323–333 (1998). 23, 26

[60] C.-H. Lee, H.-C. Yang, and T.-F. Chien. *DBHTE: A novel algorithm for extracting real-time microblogging topics*. In *Proceedings of the 23rd International Conference on Computer Applications in Industry and Engineering*, pp. 55–60 (2010). 23, 26

[61] C.-H. Lee, T.-F. Chien, and H.-C. Yang. *An automatic topic ranking approach for event detection on microblogging messages*. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, SMC '11, pp. 1358–1363 (IEEE, 2011). 23

[62] W. X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.-P. Lim, and X. Li. *Topical keyphrase extraction from Twitter*. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, vol. 1 of *HLT '11*, pp. 379–388 (ACL, 2011). 23

[63] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie. *Improving LDA topic models for microblogs via tweet pooling and automatic labeling*. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 889–892 (ACM, 2013). 24

[64] L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva. *Twitter part-of-speech tagging*

for all: Overcoming sparse and noisy data. In *Proceedings of the Recent Advances in Natural Language Processing, RANLP*, pp. 198–206 (2013). 24, 32

[65] F. Pukelsheim. *The three sigma rule*. The American Statistician **48**(2), 88 (1994). Taylor & Francis. 25

[66] G. A. Miller. *WordNet: a lexical database for English*. Communications of the ACM **38**(11), 39 (1995). ACM. 25

[67] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)* (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005). 26

[68] F. Chierichetti, J. M. Kleinberg, R. Kumar, M. Mahdian, and S. Pandey. *Event detection via communication pattern analysis*. In *Proceedings of the International AAAI Conference on Web and Social Media, ICWSM* (2014). 26

[69] J. Huang, M. Peng, and H. Wang. *Topic detection from large scale of microblog stream with high utility pattern clustering*. In *Proceedings of the 8th Workshop on Ph.D. Workshop in Information and Knowledge Management*, PIKM '15, pp. 3–10 (ACM, New York, NY, USA, 2015). 27

[70] J. Han, J. Pei, and Y. Yin. *Mining frequent patterns without candidate generation*. ACM Sigmod Record **29**(2), 1 (2000). ACM. 27

[71] M. E. Newman. *Modularity and community structure in networks*. Proceedings of the National Academy of Sciences **103**(23), 8577 (2006). National Academy of Sciences. 27

[72] M. Adedoyin-Olowe, M. M. Gaber, C. M. Dancausa, F. Stahl, and J. B. Gomes. *A rule dynamics approach to event detection in twitter with its application to sports and politics*. Expert Systems with Applications **55**, 351 (2016). Elsevier. 27

[73] M. Adedoyin-Olowe, M. M. Gaber, and F. Stahl. *TRCM: A methodology for temporal analysis of evolving concepts in Twitter*. In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, pp. 135–145 (Springer, 2013). 27

[74] Q. Dang, F. Gao, and Y. Zhou. *Early detection method for emerging topics based on dynamic Bayesian networks in micro-blogging networks*. Expert Systems with Applications **57**, 285 (2016). Elsevier. 27

[75] K. P. Murphy. *Dynamic Bayesian networks*. Probabilistic Graphical Models, M. Jordan **7** (2002). 27

[76] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al. A density-based algorithm for discovering clusters in large spatial databases with noise.* In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 226–231 (1996). 28

[77] Y. Fang, H. Zhang, Y. Ye, and X. Li. *Detecting hot topics from Twitter: A multiview approach*. Journal of Information Science **40**, 578 (2014). SAGE Publications. 28

[78] A. Kumar and H. Daumé. *A co-training approach for multi-view spectral clustering*. In *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pp. 393–400 (2011). 28

[79] A. Y. Ng, M. I. Jordan, Y. Weiss, *et al. On spectral clustering: Analysis and an algorithm*. In *Advances in Neural Information Processing Systems 2*, pp. 849–856 (MIT, 2002).

[80] A. Kumar, P. Rai, and H. Daume. *Co-regularized multi-view spectral clustering*. In *Advances in Neural Information Processing Systems 24*, pp. 1413–1421 (Curran Associates, Inc., 2011). 28

[81] N. Thapen, D. Simmie, and C. Hankin. *The early bird catches the term: Combining Twitter and news data for event detection and situational awareness*. arXiv preprint arXiv:1504.02335 (2015). 28

[82] L. Hutwagner, W. Thompson, G. M. Seeman, and T. Treadwell. *The bioterrorism preparedness and response early aberration reporting system (EARS)*. Journal of Urban Health **80**(1), i89 (2003). Springer. 28

[83] B. Robinson, R. Sparks, R. Power, and M. Cameron. *Social media monitoring for health indicators*. In *Proceedings of the 21st International Congress on Modelling and Simulation*, MODSIM '15, pp. 490–496 (Modelling and Simulation Society of Australia and New Zealand, 2015). 28

[84] R. Sparks, A. Adolphson, and A. Phatak. *Multivariate process monitoring using the dynamic biplot*. International Statistical Review **65**(3), 325 (1997). Wiley Online Library. 28

[85] R. Sparks, C. Carter, P. Graham, D. Muscatello, T. Churches, J. Kaldor, R. Turner, W. Zheng, and L. Ryan. *Understanding sources of variation in syndromic surveillance for early warning of natural or intentional disease outbreaks*. IIE Transactions **42**(9), 613 (2010). Taylor & Francis. 28

[86] J. Yin, A. Lampert, M. Cameron, B. Robinson, and R. Power. *Using social media to enhance emergency situation awareness*. IEEE Intelligent Systems **27**(6), 52 (2012). IEEE. 28

[87] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. *Parameter free bursty events detection in text streams*. In *Proceedings of the 31st International Conference on Very Large Databases*, pp. 181–192 (VLDB Endowment, 2005). 28

[88] T. Sakaki, M. Okazaki, and Y. Matsuo. *Tweet analysis for real-time event detection and earthquake reporting system development*. IEEE Transactions on Knowledge and Data Engineering **25**(4), 919 (2013). IEEE. 28

[89] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. *Bayesian filtering for location estimation*. IEEE Pervasive Computing (3), 24 (2003). IEEE. 28

[90] H. Sayyadi, M. Hurst, and A. Maykov. *Event detection and tracking in social streams*. In *Proceedings of the International AAAI Conference on Web and Social Media, ICWSM* (2009). 29, 30

[91] O. Ozdikis, P. Senkul, and H. Oguztuzun. *Semantic expansion of hashtags for*

enhanced event detection in Twitter. In *Proceedings of the 1st International Workshop on Online Social Systems* (2012). 29, 30

[92] S. Unankard, X. Li, and M. A. Sharaf. *Location-based emerging event detection in social networks*. In *Web Technologies and Applications*, vol. 7808, pp. 280–291 (Springer, 2013). 29, 30

[93] C. C. Aggarwal and K. Subbian. *Event detection in social streams.* In *Proceedings of the SIAM International Conference on Data Mining, SDM*, vol. 12, pp. 624–635 (SIAM, 2012). 29, 31, 46

[94] T. Hofmann. *Probabilistic latent semantic indexing*. In *Proceedings of the 22nd annual International ACM SIGIR Conference on Research and Development in Information retrieval*, pp. 50–57 (ACM, 1999). 29, 31

[95] J. Li, Z. Tai, R. Zhang, W. Yu, and L. Liu. *Online bursty event detection from microblog*. In *Proceedings of the IEEE/ACM 7th International Conference on Utility and Cloud Computing, UCC*, pp. 865–870 (IEEE, 2014). 29, 31

[96] J. Benhardus and J. Kalita. *Streaming trend detection in Twitter*. International Journal of Web Based Communities **9**(1), 122 (2013). Inderscience Publishers Ltd. 29, 31

[97] A. Weiler, M. Grossniklaus, and M. H. Scholl. *Evaluation measures for event detection techniques on Twitter data streams*. In *British International Conference on Databases*, pp. 108–119 (Springer, 2015). 31

[98] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. *Part-of-speech tagging for Twitter: Annotation, features, and experiments*. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, vol. 2 of *HLT '11*, pp. 42–47 (ACL, 2011). 32

[99] A. X. Chang and C. D. Manning. *SUTime: A library for recognizing and normalizing*

time expressions. In *Proceedings of the Language Resources and Evaluation Conference, LREC*, pp. 3735–3740 (2012). 32

[100] M. F. Porter. *An algorithm for suffix stripping*. Program **14**(3), 130 (1980). MCB UP Ltd. 32

[101] G. Miller and C. Fellbaum. *WordNet: An electronic lexical database* (MIT Press Cambridge, 1998). 32

[102] D. M. Blei, A. Y. Ng, and M. I. Jordan. *Latent dirichlet allocation*. Journal of Machine Learning Research **3**, 993 (2003). JMLR.org. 33, 87

[103] L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Goker, I. Kompatsiaris, and A. Jaimes. *Sensing trending topics in Twitter*. IEEE Transactions on Multimedia **15**(6), 1268 (2013). IEEE. 33, 77, 78, 79

[104] J. Lehmann, B. Gonçalves, J. J. Ramasco, and C. Cattuto. *Dynamical classes of collective attention in Twitter*. In *Proceedings of the International Conference on World Wide Web*, pp. 251–260 (ACM, 2012). 34, 83

[105] J. Yang and J. Leskovec. *Patterns of temporal variation in online media*. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pp. 177–186 (ACM, 2011). 34, 83

[106] C. Castillo, M. Mendoza, and B. Poblete. *Information credibility on Twitter*. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pp. 675–684 (ACM, New York, NY, USA, 2011). 34

[107] A. Gupta and P. Kumaraguru. *Credibility ranking of tweets during high impact events*. In *Proceedings of the 1st Workshop on Privacy and Security in Online Social Media*, PSOSM '12, pp. 2:2–2:8 (ACM, New York, NY, USA, 2012). 35

[108] A. J. McMinn, Y. Moshfeghi, and J. M. Jose. *Building a large-scale corpus for evaluating event detection on Twitter*. In *Proceedings of the 22nd ACM International*

Conference on Information and Knowledge Management, CIKM '13, pp. 409–418 (ACM, New York, NY, USA, 2013). 35, 46, 48, 54, 63, 65, 66, 77, 79

[109] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning. Adaptive computation and machine learning series* (2006). The MIT Press. 35

[110] S. J. Pan and Q. Yang. *A survey on transfer learning*. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345 (2010). IEEE. 35

[111] D. Jurgens and K. Stevens. *The S-Space package: an open source package for word space models*. In *Proceedings of the ACL System Demonstrations*, pp. 30–35 (ACL, 2010). 43, 91

[112] O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith. *Improved part-of-speech tagging for online conversational text with word clusters*. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, HLT '13, pp. 380–391 (ACL, 2013). 46, 56, 61

[113] S. Kumar, H. Liu, S. Mehta, and L. V. Subramaniam. *From tweets to events: Exploring a scalable solution for Twitter streams*. arXiv preprint arXiv:1405.1392 (2014). 47, 61

[114] M. Hasan, M. A. Orgun, and R. Schwitter. *Twitternews+: A framework for real time event detection from the twitter data stream*. In *Proceedings of the 8th International Conference on Social Informatics*, SocInfo '16, pp. 224–239 (Springer, 2016). 53, 103

[115] D. Hamby. *A review of techniques for parameter sensitivity analysis of environmental models*. Environmental monitoring and assessment **32**(2), 135 (1994). Springer. 54, 67

[116] M. Crick and M. Hill. *The role of sensitivity analysis in assessing uncertainty*. In *NEA Workshop on Uncertainty Analysis for Performance Assessments of Radioactive Waste*

Disposal Systems (Nuclear Energy Agency of the OECD (NEA): Organisation for Economic Co-operation and Development, 1987). 67

[117] C. Yu, J. Cheng, and A. Zielen. *Sensitivity analysis of the resrad, a dose assessment code*. Transactions of the American Nuclear Society **64**, 73 (1991). 54

[118] M. Hasan, M. A. Orgun, and R. Schwitter. *TwitterNews: Real time event detection from the Twitter data stream*. PeerJ PrePrints **4**, e2297v1 (2016). 63, 77, 78, 79, 102

[119] R. Sequiera and J. Lin. *Finally, a downloadable test collection of tweets*. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1225–1228 (ACM, 2017). 64

[120] Ø. K. Repp. *Event Detection in Social Media-Detecting News Events from the Twitter Stream in Real-Time*. Master's thesis, NTNU (2016). 64

[121] J. R. Landis and G. G. Koch. *The measurement of observer agreement for categorical data*. Biometrics pp. 159–174 (1977). JSTOR. 67

[122] J. Benhardus and J. Kalita. *Streaming trend detection in twitter*. Web Based Communities **9**(1), 122 (2013). Inderscience Publishers Ltd. 77, 78

[123] M. Hasan, M. A. Orgun, and R. Schwitter. *Real-time event detection from the Twitter data stream using the TwitterNews+ framework*. Information Processing & Management (2018). Elsevier, URL `https://doi.org/10.1016/j.ipm.2018.03.001`. 83, 103

[124] I. Ruthven and M. Lalmas. *A survey on the use of relevance feedback for information access systems*. Knowledge Engineering Review **18**(2), 95 (2003). Cambridge University Press. 85, 91

[125] B. Sharifi, M.-A. Hutton, and J. Kalita. *Summarizing microblogs automatically*. In *Proceedings of the Annual Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, HLT '10, pp. 685–688 (ACL, Stroudsburg, PA, USA, 2010). 86

[126] R. Mihalcea and P. Tarau. *TextRank: Bringing order into texts*. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (ACL, 2004). URL `http://www.aclweb.org/anthology/W04-3252`. 86

[127] A. Nenkova and K. McKeown. *A survey of text summarization techniques*. Mining Text Data pp. 43–76 (2012). Springer. 87

[128] M. A. H. Khan, D. Bollegala, G. Liu, and K. Sezaki. *Multi-tweet summarization of real-time events*. In *Proceedings of the International Conference on Social Computing*, SocialCom '13, pp. 128–133 (IEEE, 2013). 87

[129] T. Dunning. *Accurate methods for the statistics of surprise and coincidence*. Computational Linguistics **19**(1), 61 (1993). MIT Press. 87

[130] S. Brin and L. Page. *The anatomy of a large-scale hypertextual web search engine*. Computer Networks ISDN Systems **30**(1-7), 107 (1998). Elsevier. 87