Information Processing and Management 56 (2019) 1146–1165

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Real-time event detection from the Twitter data stream using the TwitterNews+ Framework

Mahmud Hasan[a], Mehmet A. Orgun[*,a,b], Rolf Schwitter[a]

[a] Department of Computing, Macquarie University, Sydney, Australia
[b] Faculty of Information Technology, Macau University of Science and Technology, Macau, China

A R T I C L E I N F O

Keywords:
Event detection
Incremental clustering
Social media
Microblog
Twitter
Parameter sensitivity analysis

A B S T R A C T

Detecting events in real-time from the Twitter data stream has gained substantial attention in recent years from researchers around the world. Different event detection approaches have been proposed as a result of these research efforts. One of the major challenges faced in this context is the high computational cost associated with event detection in real-time. We propose, *TwitterNews+*, an event detection system that incorporates specialized inverted indices and an incremental clustering approach to provide a low computational cost solution to detect both major and minor newsworthy events in real-time from the Twitter data stream. In addition, we conduct an extensive parameter sensitivity analysis to fine-tune the parameters used in *TwitterNews+* to achieve the best performance. Finally, we evaluate the effectiveness of our system using a publicly available corpus as a benchmark dataset. The results of the evaluation show a significant improvement in terms of recall and precision over five state-of-the-art baselines we have used.

1. Introduction

The proliferation of social networking services has resulted in a rapid increase of their user base, spanning most parts of the world. These online platforms usually 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 services 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 (Atefeh & Khreich, 2015; Hasan, Orgun, & Schwitter, 2017).

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, Wang, Ribarsky, and Zhou (2012) 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) (Allan, 2012), an event is defined as: “Something that happens at specific time and place along with all necessary conditions and unavoidable consequences”. Becker, Naaman, and Gravano (2011) 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

[*] Corresponding author.
E-mail addresses: mahmud.hasan@students.mq.edu.au (M. Hasan), mehmet.orgun@mq.edu.au (M.A. Orgun), rolf.schwitter@mq.edu.au (R. Schwitter).

https://doi.org/10.1016/j.ipm.2018.03.001
Received 31 July 2017; Received in revised form 16 December 2017; Accepted 4 March 2018
Available online 13 March 2018

instigate a large volume of tweets that discuss the associated topics at specific times. Petrovic, Osborne, McCreadie, Macdonald, and Ounis (2013) analyzed both major and minor events reported within the duration of two months in the newswire and the Twitter streams, and 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 small or local events that fall outside the radar of the newswire sources. The authors also noted (Petrovic et al., 2013) that, in some cases, Twitter leads in reporting events related to politics and business. Moreover, Osborne and Dredze (2014) compared the coverage and latency of breaking news reported on Facebook, Google Plus, and Twitter, based on the major events[1] identified from Wikipedia between the 10th to the 31st of December, 2013. These social media sites were also compared based on the long-tailed/minor events detected by the event detection system proposed by Petrović, Osborne, and Lavrenko (2010). Osborne and Dredze (2014) found Twitter to be faster in reporting breaking news than the other social media, while being outperformed by the newswires.

The findings by Petrovic et al. (2013), and Osborne and Dredze (2014) 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. (2011) which only accounts for major events and define an event in the context of social media as 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. Therefore, our proposed event detection system, *TwitterNews+*, takes as input a time-ordered stream of tweets and generates clusters of tweets where each cluster represents a major or minor newsworthy event.

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.

We have conducted an extensive survey, reported elsewhere (Hasan et al., 2017), on the existing literature on Twitter-centric event detection systems and categorized different event detection approaches based on their common traits: (a) term-interestingness, (b) topic-modeling, and (c) incremental-clustering. One of the major drawbacks of the approaches based on term interestingness (Alvanaki, Sebastian, Ramamritham, & Weikum, 2011; Gaglio, Re, & Morana, 2016; Mathioudakis & Koudas, 2010) and topic modeling (Cai, Yang, Li, & Huang, 2015; Li, Wen, Tai, Zhang, & Yu, 2015; Xie, Zhu, Ma, Xie, & Lin, 2014) is that they are computation intensive. Whereas, incremental clustering based approaches (McMinn & Jose, 2015; Petrović et al., 2010) are quite effective in reducing the computational cost involved in the event detection task. A brief discussion on a wide range of Twitter-centric event detection techniques is provided in Section 2.

Our proposed system, *TwitterNews+* (Hasan, Orgun, & Schwitter, 2016a), briefly described in Section 3, incorporates a 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. The first stage, handled by the Search Module, discussed in Section 3.1, 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 Section 3.2, involves clustering the tweets that discuss the same events and tracking these events. Note that the preprocessing and postprocessing techniques used in *TwitterNews+* are discussed in Section 4.

In this paper, we also perform an extensive parameter sensitivity analysis on the different parameters to determine the optimal parameter settings for our system, discussed in Section 6, for which we use a one-at-a-time sensitivity measure (Crick & Hill, 1987; Hamby, 1994; Yu, Cheng, & Zielen, 1991), 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 paper investigates the performance of *TwitterNews+* and five state-of-the-art baselines that cover a wide range of event detection techniques in Section 7 based on the performance evaluation process outlined in Section 5. 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 (McMinn, Moshfeghi, & Jose, 2013), which makes the results of our experiments reasonably reproducible.

2. Related work

For the purpose of a focused discussion on the related work, we classify various event detection methods based on the common traits they share; for example, identifying interesting properties in tweet keywords/terms, using probabilistic topic modeling, and incremental clustering.

(a) Term-interestingness-based approaches. TwitterMonitor (Mathioudakis & Koudas, 2010) detects emergent topics by

[1] http://en.wikipedia.org/wiki/December2013

identifying the bursty terms from the Twitter stream. A greedy search strategy is used to generate groupings for the high frequency terms that co-occur in a large number of tweets. enBlogue (Alvanaki et al., 2011) detects emergent topics from blogs and Twitter data within a given time window, by computing statistical values for tag pairs and monitoring unusual shifts in their correlations. The strength of these shifts in tag pairs is used to rank emergent topics and the top-k ranked topics are returned by the system.

Twitter Live Detection Framework (TLDF) (Gaglio et al., 2016) adapts the Soft Frequent Pattern Mining (SFPM) algorithm (Petkos, Papadopoulos, Aiello, Skraba, & Kompatsiaris, 2014) to detect relevant topics within a generic macro event from the Twitter data stream. Unlike enBlogue (Alvanaki et al., 2011) and TwitterMonitor (Mathioudakis & Koudas, 2010), TLDF uses a dynamic temporal window size to detect events based on term co-occurrences in real time. The dynamic temporal window allows the TLDF to adapt its event detection behavior 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: first, the term being recognized as a named entity (i.e., persons, organizations and locations) by the NER (De Boom, Van Canneyt, & Dhoedt, 2015) and its $tf-idf$ score, and second, the highest ratio of the likelihood of appearance for a term in the current time window and the reference body 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, but retains the terms which are relevant in the current time window and are present in 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.

Term-interestingness-based approaches such as TwitterMonitor (Mathioudakis & Koudas, 2010), TwitInfo (Marcus et al., 2011), TLDF (Gaglio et al., 2016), and TrendingScore (Benhardus & Kalita, 2013a), which involve identifying interesting properties in tweet keywords/terms in a given time window, 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. These approaches can often capture misleading term correlations, and measuring the term correlations can be computationally prohibitive in an online setting.

(b) Topic-modeling-based approaches. TopicSketch (Xie et al., 2014) detects bursty events by detecting an acceleration on the whole Twitter stream, every word, and every pair of words. The system provides a low cost solution to maintain and update this information. The sketch-based topic modeling 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. Bursty Event dEtection (BEE+) (Li et al., 2015) is a distributed and incremental topic model that discovers bursty events by modeling the temporal information of events. The burst detection in BEE+ is similar to the approach used in TwitInfo (Marcus et al., 2011).

Topic-modeling-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) (Blei, Ng, & Jordan, 2003) 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-modeling-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 (Aiello et al., 2013). Stilo and Velardi (2015) noted that, the LDA-based methods usually can only work in an off-line manner, as the temporal aspect of the events are often not considered.

(c) Incremental-clustering-based approaches. ReDites (Osborne et al., 2014) is built on the LSH-based FSD system proposed by Petrović et al. (2010) and tailored for the security domain to detect events related to security (e.g., violent events, natural disasters and emergency situations). As the LSH-based event detection system (Petrović et al., 2010) suffers from low precision, a content classifier was built to improve the precision of ReDites, based on a passive-aggressive algorithm which was 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 modeling approach based on the Violence Detection Model (VDM) proposed by Cano Basave, He, Liu, and Zhao (2013). The VDM starts with a security-related word lexicon created from DBpedia,[2] and subsequently discovers new words related to security events for better classification.

Becker et al. (2011) 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 Support-Vector-Machine-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.

Boom and Canneyt (De Boom et al., 2015) augmented semantic information with individual tweets on the incremental clustering approach adopted by Becker et al. (2011). The authors have used TwitterLDA (Zhao et al., 2011) 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. (2011) baseline. However, the hashtag level semantics will work only when event related tweets contain hashtags. A similar observation was made by Mehrotra, Sanner, Buntine, and Xie (2013), 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. (2011) is taken by Phuvipadawat and Murata (2010). The

[2] http://wiki.dbpedia.org/

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 (2015) 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 (McMinn & Jose, 2015) 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 (Derczynski, Ritter, Clark, & Bontcheva, 2013) 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 (McMinn & Jose, 2015) 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 (McMinn & Jose, 2015) 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 (Pukelsheim, 1994), 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.

Incremental-clustering-based approaches are prone to fragmentation, and usually are 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. (2011), and McMinn and Jose (2015) incorporated a separate component to address the fragmentation issue. Despite the issues faced, we believe that an incremental-clustering-based approach can be utilized in our proposed 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. Moreover, 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.

Our proposed system, *TwitterNews+*, aims to detect both major and minor events from the Twitter data stream using an incremental-clustering-based approach that provides a low cost solution to the problem of event detection from the Twitter data stream and a substantial gain in performance in detecting newsworthy events compared to the state-of-the-art baselines used in our experiments. Further details on the literature related to the Twitter-centric event detection can be found in the survey we have conducted (Hasan et al., 2017).

3. Architecture of the TwitterNews+ System

The two main components of *TwitterNews+* are the Search Module and the EventCluster Module (Fig. 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.

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.

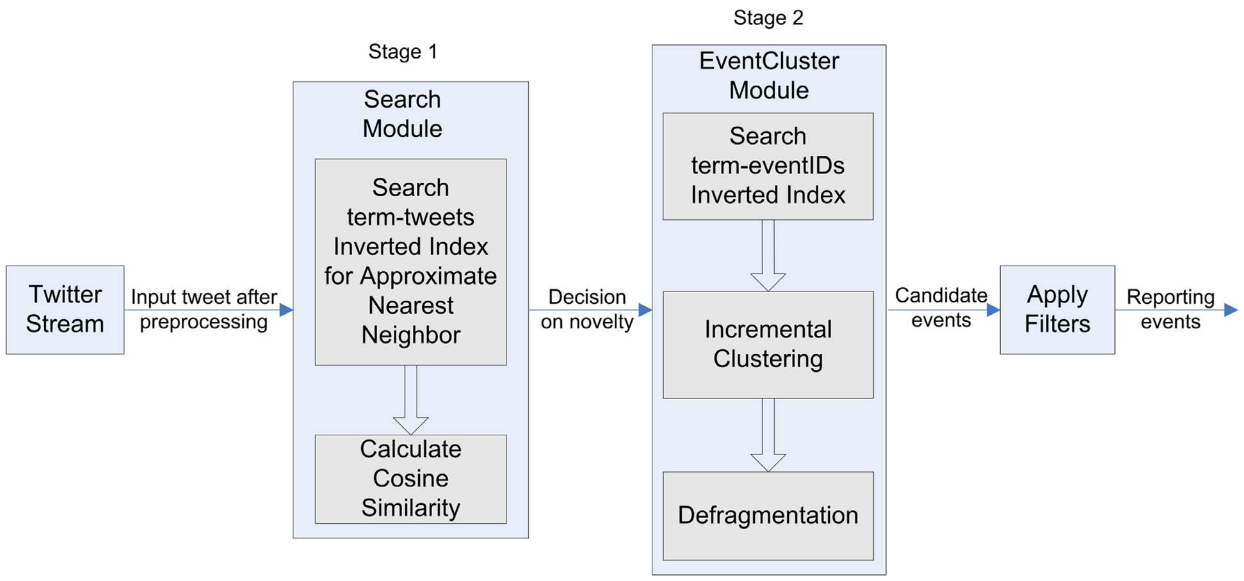


Fig. 1. *TwitterNews+* Architecture.

3.1. 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 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 d, while maintaining a constant time and space requirement, we utilize a *term-tweets* inverted index (Fig. 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 d, the tweet is first tokenized and part-of-speech tagged (Owoputi et al., 2013) as part of the preprocessing stage and an incremental $tf - idf$-based term vector is generated.

Subsequently, the top-k $tf - idf$ weighted terms are selected from d, 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 (see Fig. 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 for the cosine similarity, t_{sr}, is

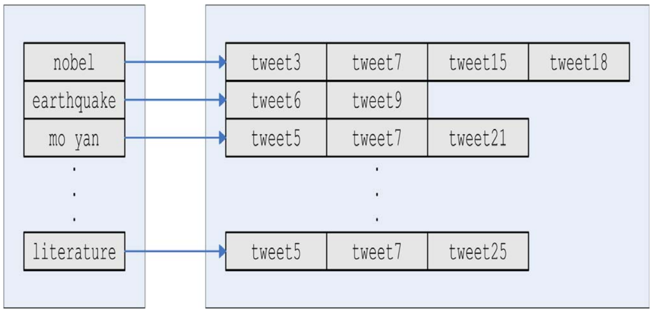


Fig. 2. *term-tweets* Inverted index.

```
Require: threshold ($t_{sr}$) value for the cosine similarity measure
 1: for each tweet $d$ in the twitter data stream do
 2:     select top-k $tf - idf$ weighted terms of $d$
 3:     $S$ ←set of tweets retrieved from the "term-tweets" inverted index containing any top-k terms of $d$    ▷ $|S| \leq K \times Q$
 4:     for each tweet $d'$ in $S$ do
 5:         calculate the cosine similarity $cosSim$ between $d'$ and $d$ using an available thread from a pool of threads in a multi-
    threaded execution environment
 6:         assign $cosSim$ to the set $S'$
 7:     end for
 8:     $novelInputTweet \leftarrow$ true
 9:     for each $cosSim$ in $S'$ do
10:         if $cosSim > t_{sr}$ then
11:             $novelInputTweet$ = false
12:         end if
13:     end for
14:     if $novelInputTweet$ then
15:         $d$ is "unique"
16:     else
17:         $d$ is "not unique"
18:     end if
19:     update term-tweets inverted index
20: end for
```

Algorithm 1. TwitterNews+ Search Module.

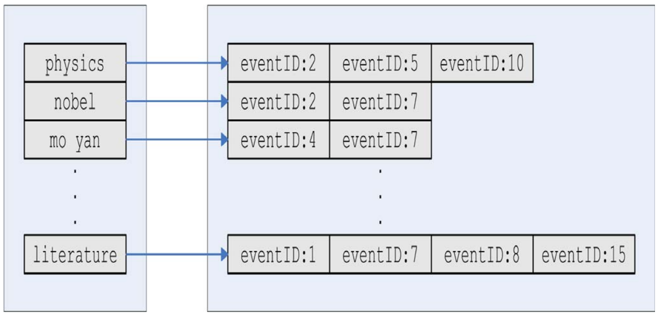


Fig. 3. *term-eventIDs* Inverted index.

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 6. 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 1) 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.

3.2. The EventCluster module

The Search Module sends the tweets that are decided as "*not unique*" to the EventCluster module. Upon receiving a tweet d, 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 d can be assigned (Fig. 3). Each entry 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 d 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 (Fig. 3). Note that the total number of event clusters with which the input tweet is compared will always 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 2) 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 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 postprocessing stage as described in Section 4.

```
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
 1: $C \leftarrow$set of events retrieved from the “term-eventIDs” inverted index containing any of the top-k terms of the input tweet $d$   ▹ $|C| \leq K \times Q$
 2: for each active event cluster $c$ in $C$ do
 3:     calculate the cosine similarity $cosSim$ between $c$ and $d$ using an available thread from a pool of threads in a multithreaded
        execution environment
 4:     assign $cosSim$ to the set $C'$
 5:     if $cosSim \geq (t_{ev} + g_{ev})$ then
 6:         $S_c \leftarrow$ assign $c$ to the set of fragmented event clusters
 7:     end if
 8: end for
 9: $cosSim_{\max} \leftarrow 0$
10: for each $cosSim$ in $C'$ do
11:     if $cosSim > cosSim_{\max}$ then
12:         $cosSim_{\max} = cosSim$
13:     end if
14: end for
15: if $cosSim_{\max} > t_{ev}$ then
16:     assign $d$ to the cluster $c_{target}$ with the maximum similarity $cosSim_{\max}$
17:     merge the clusters from the set $S_c$ with $c_{target}$
18:     update $c_{target}$ centroid by averaging with the event centroids in $S_c$ and the tweet vector
19:     update $c_{target}$ expiry time
20: else
21:     create a new cluster $c_{new}$ and assign $d$ to it
22:     assign the vector of $d$ as the centroid of $c_{new}$
23:     assign an initial expiry time to $c_{new}$
24: end if
25: update $term\text{-}eventIDs$ inverted index
```

Algorithm 2. TwitterNews+ EventCluster Module.

Table 1
Filters used in the postprocessing 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.

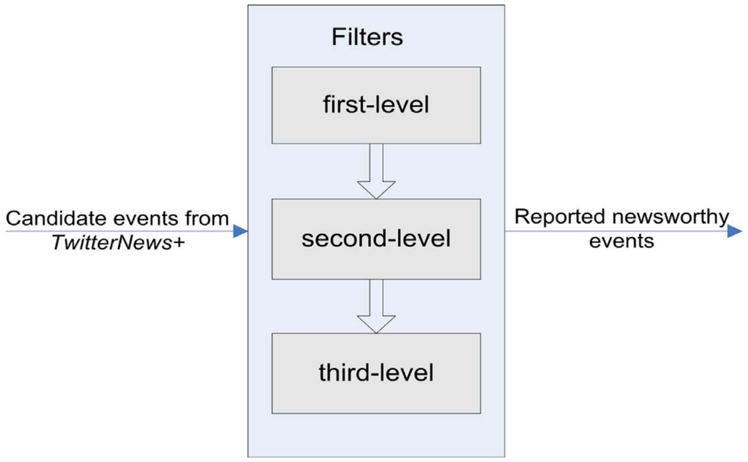

Filters
first-level
second-level
third-level
Candidate events from TwitterNews+
Reported newsworthy events

Fig. 4. Postprocessing pipeline in *TwitterNews*+.

4. Preprocessing and postprocessing operations in TwitterNews+

In this section, we discuss the main preprocessing and postprocessing operations involved in *TwitterNews*+.

Preprocessing. 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 preprocessing 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 (Owoputi et al., 2013). 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 postprocessing stage which is discussed below.

Postprocessing. A combination of three different level of filters as shown in Table 1 is applied to discard the trivial events while retaining the newsworthy events from the candidate event clusters generated by *TwitterNews*+ (Fig. 4).

The *first-level* filters use the entropy (Petrović et al., 2010) and the user diversity (Kumar, Liu, Mehta, & Subramaniam, 2014) 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 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 news entities.[3] 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*

[3] http://www.journalism.org/media-indicators/digital-top-50-online-news-entities-2015/

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* (Hasan, Orgun, & Schwitter, 2016b) (see Section 7).

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 3). 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.

The LCS-based scheme, adopted as the *third-level* filter, also selects a representative tweet from an event cluster by emitting the tweet having the maximum LCS in the 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 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 (McMinn & Jose, 2015).

The remaining events after the postprocessing stage are the newsworthy events reported by our system. Table 2 shows the representative tweets of the newsworthy events reported by *TwitterNews+* from the first three days of tweets contained in the Events2012 corpus (McMinn et al., 2013). The event representative tweets were selected using the LCS-based scheme. For each day only one event's representative tweet is shown to keep Table 2 concise.

5. 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 (9th to 11th of October, 2012) from the Events2012 corpus (McMinn et al., 2013) using different parameter settings (see Section 6). The corpus contains 120 million tweets collected from the 9th of October to the 7th 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.

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 (2017) 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 (2016).

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 (McMinn et al., 2013). The problem to deal with here is the substantial amount of time required to manually reconfirm a total of 506 ground truth events spanning from the 9th of October to the 7th of November, 2012. After a careful feasibility consideration, we have only reconfirmed the first three days (9th to 11th 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.[4] 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 (Hasan et al., 2017), 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 and Jose (2015) used automated methods at different levels of granularity which captured sub-events that describe only

[4] https://hasan922ccc.wixsite.com/tnplus

```
Require: $t_{lcs}$ value
 1: for each candidate event cluster $c$ in $C$ do                    ▹ parallel processing
 2:     $lcs_{\max} \leftarrow 0$
 3:     $D \leftarrow$ event $c$ related tweets
 4:     for $i = 0$ to $sizeof(D) - 2$ do
 5:         for $j = i + 1$ to $sizeof(D) - 1$ do
 6:             $tempLcsLength$ = calculateWordLevelLCS($D[i]$, $D[j]$)
 7:             if $tempLcsLength > lcs_{\max}$ then
 8:                 $lcs_{\max} = tempLcsLength$
 9:             end if
10:         end for
11:     end for
12:     if $lcs_{\max} < t_{lcs}$ then
13:         remove $c$ from candidate event set
14:     else
15:         report the tweet with $lcs_{\max}$ as a representative of the event cluster $c$
16:     end if
17: end for
```

Algorithm 3. LCS-based filtering and representative tweet selection.

Table 2
Representative tweets of the selective newsworthy events reported by *TwitterNews+*.

Date	Event Representative Tweet
09 Oct 2012	Retweet If You're Watching The "BET Hip Hop Awards2012"
10 Oct 2012	BAE-EADS merger plans are 'off': Aerospace and defense firms BAE and EADS have canceled their planned merger, t... http://bbc.in/Tvu31e
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/2sCWey

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 (2015) 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 (Landis & Koch, 1977). 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.

6. 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 3 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 (Crick & Hill, 1987; Hamby, 1994) where one parameter at a time is repeatedly varied while keeping the others fixed. 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+*.

Determining the Optimal M Value. 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 (Fig. 5 a) 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 Fig. 5a and a value of 50,000 for M.

Fig. 6 asummarizes the effect on *TwitterNews+*'s performance, in terms of recall and precision, for different M values. In addition, Fig. 7 a 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

Table 3
Outline of the parameter sensitivity analysis for *TwitterNews+*.

Parameter Description	Notation	Experiment
Number of most recent tweets to store for similarity comparison	M	$M0$: 50000 $M1$: 100000 $M2$: 200000 $M3$: 400000 $M4$: 800000
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
Threshold for the cosine similarity measure in the Search Module which already is fixed as 0.6, based on the findings from the related works (McMinn & Jose, 2015; Petrović et al., 2010) and our prior experiments	t_{sr}	n/a
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
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
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
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
Threshold to merge 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

(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

Fig. 5. Baseline parameter settings in different stages of the parameter sensitivity analysis of *TwitterNews+*.

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

(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

Fig. 6. *TwitterNews+* performance on different parameter settings.

the experiments, while the optimal ts_i value is being determined, the other parameter settings are kept fixed, as shown in Fig. 5b. The results for the experiments to determine the optimal ts_i value are summarized in Fig. 6b and the run-time dependency of our system on the different ts_i values is shown in Fig. 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 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 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 (McMinn & Jose, 2015; Petrović et al., 2010) 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 Fig. 5c. The results for the experiments to determine the optimal Q value are summarized in Fig. 6c. Furthermore, Fig. 7 c 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 Fig. 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 Fig. 6 d, the results of the experiments to determine optimal t_{ent} value are summarized and in Fig. 7 d 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

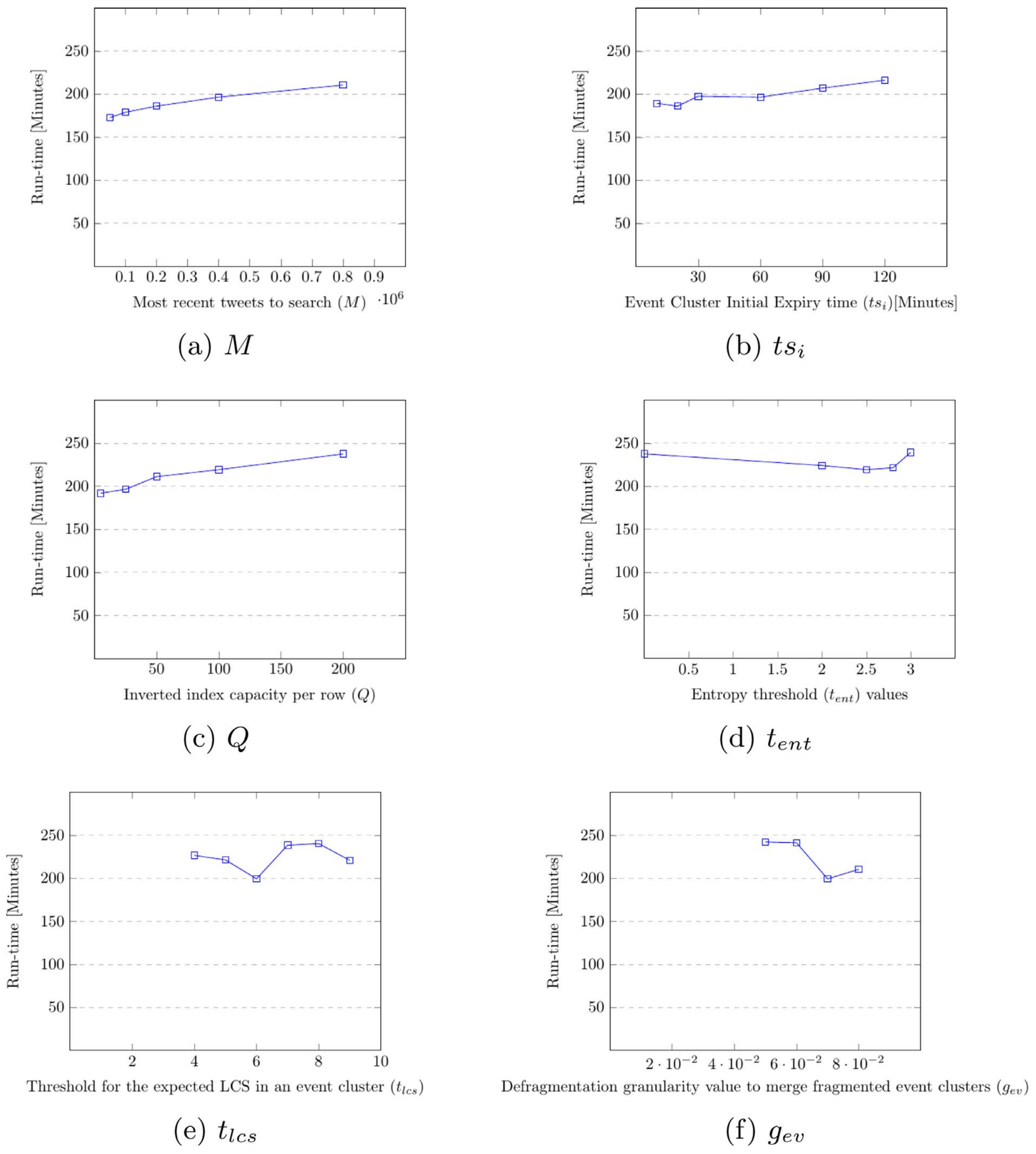


Fig. 7. *TwitterNews+* run-time dependency on different parameter settings.

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 Fig. 5e. In Fig. 6 e, the results of the experiments to determine optimal t_{lcs} value are summarized and in Fig. 7 e 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.

Table 4
Recommended parameter settings for *TwitterNews+*.

Parameter	Value
M	400,000
ts_i	60
t_{sr}	0.6
t_{ev}	0.6
Q	100
t_{ent}	2.8
t_{lcs}	6
g_{ev}	0.07

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 Fig. 5f. We refer to Fig. 7 f, 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 Fig. 6 f 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, which achieves a recall of 0.96 and a precision of 0.89.

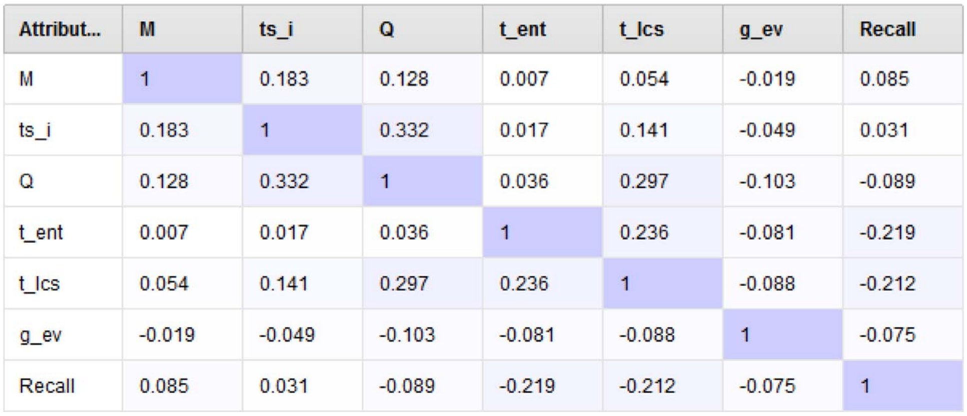
Correlation analysis. We have performed a correlation analysis using RapidMiner[5] to determine the degree of association between the different parameters and the recall values achieved by *TwitterNews+* in each parameter setting (Fig. 8). From the Pearson's correlation coefficient matrix in Fig. 8 we can determine that the parameter Q, which is used to set the row capacity of both of the inverted-indices utilized in *TwitterNews+*, 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 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 Fig. 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 (Fig. 9). From the Pearson's correlation coefficient matrix in Fig. 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 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 Fig. 9 exceeds the critical value of 0.468 at $P < 0.05$ for it to be statistically significant.

7. Performance evaluation results

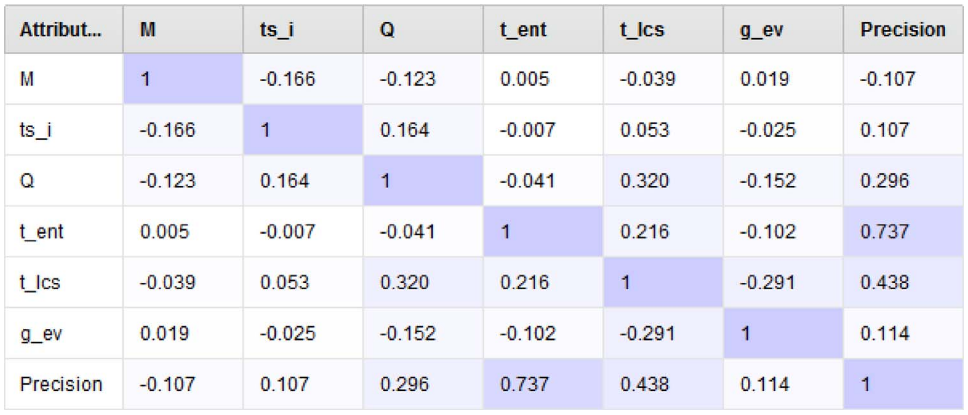
We have evaluated the events reported by *TwitterNews+* within the time window of three days. The five baselines we have used to compare with our system are: First Story Detection (FSD) system (Petrović et al., 2010), an incremental-clustering-based approach; *TwitterNews* (Hasan et al., 2016b), our previously proposed system also based on incremental clustering; LDA-SocialSensor,[6] a topic-modeling-based approach (Aiello et al., 2013); TrendingScore[7] (Benhardus & Kalita, 2013b), a term-interestingness-based approach; and mention-anomaly-based Event Detection (MABED) (Guille & Favre, 2015), 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 5 summarizes the results of our evaluation.

[5] https://rapidminer.com/
[6] http://www.socialsensor.eu/results/software/87-topic-detection-framework
[7] https://github.com/AdrienGuille/SONDY



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

Fig. 8. Pearson's correlation coefficient matrix for *TwitterNews+* parameters and recall.



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

Fig. 9. Pearson's correlation coefficient matrix for *TwitterNews+* parameters and precision.

Table 5
Summary of the evaluation results.

Method	Recall	Precision
FSD (Petrović et al., 2010)	0.52	–
LDA-SocialSensor (Aiello et al., 2013)	0.45	0.49
MABED (Guille & Favre, 2015)	0.58	0.55
TrendingScore (Benhardus & Kalita, 2013b)	0.71	0.64
TwitterNews (Hasan et al., 2016b)	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 6). To achieve a fair evaluation, the input dataset from the Events2012 corpus went through the same preprocessing 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 7 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. (2013) used automated methods at different levels of granularity which captured: 1) sub-events that 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.

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

Table 6
Parameter setting used for the baselines.

Method	Parameter Setting
FSD (Petrović et al., 2010)	cosine similarity threshold = 0.6
LDA-SocialSensor (Aiello et al., 2013)	no of topics = 500
	no of training iterations = 300
	no of keywords returned per topic = 10
MABED (Guille & Favre, 2015)	minTermSupport = .0001
	maxTermSupport = .01
	k = 500
	p = 10
	theta = 0.7
	sigma = 0.7
TrendingScore (Benhardus & Kalita, 2013b)	minTermSupport = .0001
	maxTermSupport = .01
	trendingThreshold = 10.0
TwitterNews (Hasan et al., 2016b)	similar to *TwitterNews+*

Table 7
Number of ground truth events detected by different event detection systems.

Method	Number
FSD (Petrović et al., 2010)	16
LDA-SocialSensor (Aiello et al., 2013)	14
MABED (Guille & Favre, 2015)	18
TrendingScore (Benhardus & Kalita, 2013b)	22
TwitterNews (Hasan et al., 2016b)	27
TwitterNews+	30

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 different filters in the postprocessing phase of *TwitterNews+*, different from our previously proposed system *TwitterNews* (Hasan et al., 2016b), was instrumental in the substantial improvement of precision compared to the precision of 0.72 achieved by *TwitterNews*. The postprocessing 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.

Efficiency evaluation by analyzing computational complexity mathematically is rarely done in the literature on event detection systems (Hasan et al., 2017). 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 (Aiello et al., 2013) 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.

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[8] on average tweeted on Twitter every second at the time of writing this paper 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 and EventCluster modules 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.

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

8. 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 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 an extensive 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.

To the best of our knowledge, *TwitterNews+* is one of the efficient systems to detect events from Twitter in real-time, in terms of the *number-of-tweets/second* processing capability, maintains a constant space and processing time, and achieves very good results compared to the baselines that cover a wide range of event detection techniques. The different set of filters, applied to extract newsworthy events from the set of candidate events, helps in retaining both minor and major 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 major events based on a burst detection approach that require 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. We plan to implement a sub-module for *TwitterNews+* in the future, which performs a temporal summarization of the tweets in an event cluster, instead of relying on the LCS-based scheme currently used to select a representative tweet from an event cluster.

Funding

Mahmud Hasan has been supported by a Macquarie University Research Excellence Scholarship (MQRES).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ipm.2018.03.001

References

Aiello, L. M., Petkos, G., Martin, C., Corney, D., Papadopoulos, S., Skraba, R., et al. (2013). Sensing trending topics in Twitter. *IEEE Transactions on Multimedia, 15*(6), 1268–1282 IEEE.

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

Alvanaki, F., Sebastian, M., Ramamritham, K., & Weikum, G. (2011). *Enblogue: Emergent topic detection in web 2.0 streams. Proceedings of the ACMSIGMOD international conference on management of dataSIGMOD '11*New York, NY, USA: ACM1271–1274.

Atefeh, F., & Khreich, W. (2015). A survey of techniques for event detection in Twitter. *Computational Intelligence, 31*(1), 132–164 Wiley Online Library.

Becker, H., Naaman, M., & Gravano, L. (2011). *Beyond trending topics: Real-world event identification on Twitter. Proceedings of the fifth international AAAI conference on weblogs and social media, ICWSM.* AAAI438–441.

Benhardus, J., & Kalita, J. (Kalita, 2013a). Streaming trend detection in Twitter. *International Journal of Web Based Communities, 9*(1), 122–139 Inderscience Publishers Ltd.

Benhardus, J., & Kalita, J. (Kalita, 2013b). Streaming trend detection in twitter. *Web Based Communities, 9*(1), 122–139 Inderscience Publishers Ltd.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research, 3*, 993–1022 JMLR.org.

Cai, H., Yang, Y., Li, X., & Huang, Z. (2015). *What are popular: Exploring Twitter features for event detection, tracking and visualization. Proceedings of the 23rd annual ACM conference on multimedia conference.* ACM89–98.

Cano Basave, A. E., He, Y., Liu, K., & Zhao, J. (2013). *A weakly supervised Bayesian model for violence detection in social media. Proceedings of the international joint conference on natural language processing, IJCNLP.* Asian Federation of Natural Language Processing.

Crick, M., & Hill, M. (1987). *The role of sensitivity analysis in assessing uncertainty. Uncertainty analysis for performance assessments of radioactive waste disposal systems* Nuclear Energy Agency of the OECD (NEA): Organisation for economic co-operation and development.

De Boom, C., Van Canneyt, S., & Dhoedt, B. (2015). *Semantics-driven event clustering in Twitter feeds. Proceedings of the 5th workshop on making sense of microposts1395. Proceedings of the 5th workshop on making sense of microposts* CEUR2–9.

Derczynski, L., Ritter, A., Clark, S., & Bontcheva, K. (2013). *Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. Proceedings of the recent advances in natural language processing, RANLP*198–206.

Dou, W., Wang, K., Ribarsky, W., & Zhou, M. (2012). *Event detection in social media data. Proceedings of the IEEE visweek workshop on interactive visual text analytics - task driven analytics of social media content*971–980.

Gaglio, S., Re, G. L., & Morana, M. (2016). A framework for real-time Twitter data analysis. *Computer Communications, 73*, 236–242 Elsevier.

Guille, A., & Favre, C. (2015). Event detection, tracking, and visualization in Twitter: A mention-anomaly-based approach. *Social Network Analysis and Mining, 5*(1), 1–18 Springer.

Hamby, D. (1994). A review of techniques for parameter sensitivity analysis of environmental models. *Environmental monitoring and assessment, 32*(2), 135–154 Springer.

Hasan, M., Orgun, M. A., & Schwitter, R. (Orgun, Schwitter, 2016a). *Twitternews+: A framework for real time event detection from the twitter data stream. Proceedings of the 8th international conference on social informaticsSocInfo '16*Springer224–239.

Hasan, M., Orgun, M. A., & Schwitter, R. (Orgun, Schwitter, 2016b). TwitterNews: Real time event detection from the Twitter data stream. *PeerJ PrePrints, 4*, e2297v1. http://dx.doi.org/10.7287/peerj.preprints.2297v1.

Hasan, M., Orgun, M. A., & Schwitter, R. (2017). A survey on real-time event detection from the Twitter data stream. *Journal of Information Science, 0*(0), http://dx.doi.org/10.1177/0165551517698564 0165551517698564. SAGE Publications Sage UK: London, England.

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

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics,* 159–174 JSTOR.

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

Marcus, A., Bernstein, M. S., Badar, O., Karger, D. R., Madden, S., & Miller, R. C. (2011). *TwitInfo: Aggregating and visualizing microblogs for event exploration. Proceedings of the SIGCHI conference on human factors in computing systemsCHI '11*New York, NY, USA: ACM227–236.

Mathioudakis, M., & Koudas, N. (2010). *TwitterMonitor: Trend detection over the Twitter stream. Proceedings of the ACMSIGMOD international conference on management of dataSIGMOD '10*New York, NY, USA: ACM1155–1158.

McMinn, A. J., & Jose, J. M. (2015). *Real-time entity-based event detection for Twitter. Proceedings of the experimental ir meets multilinguality, multimodality, and interaction: 6th international conference of the CLEF associationCLEF '15*Springer65–77.

McMinn, A. J., Moshfeghi, Y., & Jose, J. M. (2013). *Building a large-scale corpus for evaluating event detection on Twitter. Proceedings of the 22nd ACM international conference on information and knowledge managementCIKM '13*New York, NY, USA: ACM409–418.

Mehrotra, R., Sanner, S., Buntine, W., & Xie, L. (2013). *Improving LDA topic models for microblogs via tweet pooling and automatic labeling. Proceedings of the 36th international ACMSIGIR conference on research and development in information retrieval.* ACM889–892.

Osborne, M., & Dredze, M. (2014). *Facebook, Twitter and Google Plus for breaking news: Is there a winner? Proceedings of the international AAAI conference on web and social media, icwsm*.

Osborne, M., Moran, S., McCreadie, R., Von Lunen, A., Sykora, M. D., Cano, E., et al. (2014). *Real-time detection, tracking, and monitoring of automatically discovered events in social media. Proceedings of the 52nd annual meeting of the association for computational linguistics*. Association for Computational Linguistics/© The Authors.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., & Smith, N. A. (2013). *Improved part-of-speech tagging for online conversational text with word clusters. Proceedings of the annual conference of the north american chapter of the association for computational linguistics: Human language technologiesHLT '13*ACL380–391.

Petkos, G., Papadopoulos, S., Aiello, L., Skraba, R., & Kompatsiaris, Y. (2014). *A soft frequent pattern mining approach for textual topic detection. Proceedings of the 4th international conference on web intelligence, mining and semantics, WIMS*. ACM25:1–25:10.

Petrović, S., Osborne, M., & Lavrenko, V. (2010). *Streaming first story detection with application to Twitter. Proceedings of the annual conference of the north american chapter of the association for computational linguistics: Human language technologiesHLT '10*Stroudsburg, PA, USA: ACL181–189.

Petrovic, S., Osborne, M., McCreadie, R., Macdonald, C., & Ounis, I. (2013). *Can Twitter replace newswire for breaking news? Proceedings of the international AAAI conference on web and social media, ICWSM*.

Phuvipadawat, S., & Murata, T. (2010). *Breaking news detection and tracking in Twitter. Proceedings of the IEEE/WIC/ACM international conference on web intelligence and intelligent agent technologyWI-IAT '103. Proceedings of the IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology* Washington, DC, USA: IEEE Computer Society120–123.

Pukelsheim, F. (1994). The three sigma rule. *The American Statistician, 48*(2), 88–91 Taylor & Francis.

Repp, Ø. K. (2016). Event detection in social media-detecting news events from the twitter stream in real-time. Master's thesis. NTNU.

Sequiera, R., & Lin, J. (2017). *Finally, a downloadable test collection of tweets. Proceedings of the 40th international ACMSIGIR conference on research and development in information retrieval*. ACM1225–1228.

Stilo, G., & Velardi, P. (2015). Efficient temporal mining of micro-blog texts and its application to event discovery. *Data Mining and Knowledge Discovery,* 1–31 Springer.

Xie, R., Zhu, F., Ma, H., Xie, W., & Lin, C. (2014). CLEar: A real-time online observatory for bursty and viral events. *Proceedings of the VLDB Endowment, 7*(13), 1637–1640 VLDB Endowment.

Yu, C., Cheng, J., & Zielen, A. (1991). Sensitivity analysis of the resrad, a dose assessment code. *Transactions of the American Nuclear Society, 64*, 73–74.

Zhao, W. X., Jiang, J., He, J., Song, Y., Achananuparp, P., Lim, E.-P., et al. (2011). *Topical keyphrase extraction from Twitter. Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologiesHLT '111. Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* ACL379–388.