TwitterNews: Real Time Event Detection from the Twitter Data Stream

Mahmud Hasan
Department of Computing
Macquarie University
Sydney, Australia
Email: mahmud.hasan@students.mq.edu.au

Mehmet Orgun
Department of Computing
Macquarie University
Sydney, Australia
Email: mehmet.orgun@mq.edu.au

Rolf Schwitter
Department of Computing
Macquarie University
Sydney, Australia
Email: rolf.schwitter@mq.edu.au

***Abstract*—Research in event detection from the Twitter streaming data has been gaining momentum in the last couple of years. Although such data is noisy and often contains misleading information, Twitter can be a rich source of information if harnessed properly. In this paper, we propose a scalable event detection system, *TwitterNews*, to detect and track newsworthy events in real time from Twitter. *TwitterNews* provides a novel approach, by combining random indexing based term vector model with locality sensitive hashing, that aids in performing incremental clustering of tweets related to various events within a fixed time. *TwitterNews* also incorporates an effective strategy to deal with the cluster fragmentation issue prevalent in incremental clustering. The set of candidate events generated by *TwitterNews* are then filtered, to report the newsworthy events along with an automatically selected representative tweet from each event cluster. Finally, we evaluate the effectiveness of *TwitterNews*, in terms of the recall and the precision, using a publicly available corpus.**

1. Introduction

Social media sites are continuously gaining popularity as effective means of communicating information. Millions of users share information on different aspects of everyday life through these social networking services. Information shared in these platforms range from personal status (i.e., opinions, emotions, pointless babbles) to newsworthy events, as well as updates and discussions on these events. Twitter is a popular social media site providing microblogging service, where users can 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 covering an event than the traditional news media. As Twitter users do not have to be concerned with how to structure a write up on news events or how news will be perceived by the readers, they have the advantage of spreading event related updates before they appear on the traditional news. Also, local events that have low news coverage are spread through Twitter.

In this paper we use the definition of an event, in a social media context, provided by Dou et al. [1]: "*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 accordance with this definition, the series of coordinated terrorist attacks in Paris on 13th November 2015 is an event in the context of social media, that prompted a high volume or burst of tweets as soon as the attacks took place. The various aspects of this event were tweeted by the general users and the news agencies all over the world, e.g., "*BREAKING.This is what we know: 35 dead, 100 hostages taken at a concert venue. Various drive by shootings. Explosions at a #Paris stadium.*".

Although the previous example is a high profile event of a global consequence, an event detection system should also be able to detect newsworthy events at a smaller scale with a lower burst of tweets at any given time. There are, however, a number of challenges in real time event detection from the Twitter data streams. The message length restriction of 140 characters greatly reduces the amount of textual information obtained from a tweet and the messages themselves are noisy: containing typos, grammatical errors, abbreviations, etc. In addition, most of the content found on Twitter is not related to any event and the high volume of data on a diverse number of topics poses a big challenge in terms of scalability. Despite these challenges, it would be beneficial to develop a system that can lead to real time detection and tracking of events as Twitter does not provide a tool to see event summaries except for searching on trending topics or using hashtags.

Different approaches have been taken by the researchers to deal with the event detection task. The approaches based on term interestingness [2]–[6] and topic modeling [7]–[11] suffer from high computational cost among other things. However, incremental clustering based approaches [12]–[14] usually provide a low computational cost solution. Taking this into consideration, we propose an incremental clustering based end-to-end solution to detect newsworthy events from a stream of time ordered tweets.

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/event and the second stage involves grouping/clustering the tweets that discuss the same event. The operation of our proposed system, *TwitterNews*, is therefore divided into two major stages. After the preprocessing 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 a topic/event has occurred and the output of the first stage declares the input tweet to be bursty ("*not unique*"). The operation in the first stage of *TwitterNews* is implemented by combining Random Indexing (RI) based term vector model [15] with the Locality Sensitive Hashing (LSH) scheme proposed by Petrovic et al. [12], to determine whether a tweet is "*not unique*". To the best of our knowledge, 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, 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 ensure 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. Newsworthy events are reported along with a representative tweet for each event cluster by employing a Longest Common Subsequence (LCS) based scheme that works on the word-level.

The rest of the paper is organized as follows: Section 2 contains a brief discussion on the related work, then we introduce our proposed system, *TwitterNews*, in Section 3 and expand on the various aspects of the system in Sections 4 and 5. Finally, we discuss the results of our experiments and evaluation of *TwitterNews* in Section 6.

2. Related Work

We discuss the related work by organizing them into approaches that share common traits (i.e., identifying interesting properties in tweet terms, using probabilistic topic modeling and incremental clustering). For more details we refer to the survey on event detection techniques conducted by Atefeh and Khreich [13].

Term Interestingness Based Approaches. The event detection system in Twevent [2] initially extracts continuous and non overlapping word segments 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, related event segments are clustered 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, which is calculated using the Wikipedia as a knowledge base.

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

TwitInfo [3] allows a user to input event related keywords to track an event. The system starts logging the tweets that matches the user specified keywords, and uses a streaming algorithm to detect spikes in tweet data and automatically labels them with frequently occurring meaningful terms from the tweets. The peak generated by the high volume of Twitter posts are considered as sub-events.

TwitterMonitor [4] detects emergent topics by first identifying the bursty terms from the tweets within a small time window. If the system detected high frequency terms co-occur in a large number of tweets in the given time window, they are placed in the same group. A greedy strategy is used to generate groupings, in order to avoid the high computational cost to enumerate all possible groups. Similarly, enBlogue [16] computes statistical values for tag pairs within a given time window and monitors unusual shifts in the tag correlations to detect emergent topics.

Topic Modeling Based Approaches. TwiCal [9] is an open-domain event detection system that provides a structured representation of the significant events extracted from the Twitter data stream. The proposed system extracts named entities, along with associated event phrases and dates from each of the streaming tweets. A latent variable based model is used to discover important event types from a large tweet corpus. The event types that are found to be coherent during the inspection of the discovered event types were retained, and manually annotated with informative labels. Once appropriate event types were identified, they were used to categorize event phrases extracted from the subsequent new data without requiring any further manual annotation. Events are ranked by measuring the association strength between an entity and a specific date, based on G^2 log likelihood ratio statistic.

Latent Event and Category Model (LECM) [10] is a latent variable model similar to TwiCal [9]. However, LECM incorporates semantic concept to categorize events of different type.

Incremental Clustering Based Approaches. Becker et al. [17] 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 cluster. If the similarity of a tweet is not higher than a specific threshold in any of the existing clusters, then 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 the newsworthy events and the non-events.

McMinn et al. [14] have utilized an inverted index for each named entity with its associated near neighbors to cluster the bursty named entities for event detection and tracking. The effectiveness of this approach, however, is dependent on the accuracy of the underlying Named Entity Recognizer [18] used by the system.

Petrovic et al. [12] have adapted a variant of the locality sensitive hashing (LSH) technique to determine the novelty of a tweet by comparing it with a fixed number of previously encountered tweets. A novel tweet represents a new story, which is assigned to a newly created cluster. On the other hand, a tweet determined as 'not novel' is assigned to an existing cluster containing the nearest neighbor. Event clusters are ranked based on a combination of the entropy information in a cluster and the number of unique user posts.

All of the aforementioned general approaches to event detection from Twitter have their own set of drawbacks. The approaches based on term interestingness [4]–[6] can often capture misleading term correlations, and measuring the term correlations can be computationally prohibitive in an online setting. The topic modeling based approaches [7], [8] incur too high of a computational cost to be effective in a streaming setting, and are not quite effective in handling events that are reported in parallel [19]. Incremental clustering based approaches are prone to fragmentation, and usually are unable to distinguish between two similar events taking place around the same time [12], [14], [20]. However, despite the challenges, we believe incremental clustering approach is the way to go to avoid the high computational cost associated with most of the state-of-the art approaches.

3. TwitterNews Architecture

From the continuation of our discussion in Section 1 regarding the two major stages of operation in *TwitterNews*, here we will discuss the two major components in our system, each of which deals with the operation of a specific stage (Figure. 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 fast retrieval of the neighboring tweets of the input tweet for text similarity comparison. This is achieved by using the adapted variant of the Locality Sensitive Hashing (LSH) approach employed by Petrovic et al. [12], where a set of most recent tweets are stored in a fixed number of hash tables. However, Petrovic et al. [12] have used a variable length $tf-idf$ based 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 tweet from the hash tables.

To reduce the computational cost of calculating the hash keys, *TwitterNews* uses a random indexing based term vector model. 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 fast calculation of the hash keys for an input tweet, and thus 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 higher than a specific threshold value, then the input tweet is decided to be bursty ("*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 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 LCS based scheme, along with a set of different filters to filter out some of the trivial events from the candidate event clusters, and identifies a representative tweet for each event.

4. 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 in a fast way 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 new tweet with the oldest tweet in the bucket. 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 Random Indexing (RI) based term vector model and the Locality Sensitive Hashing based (LSH) scheme of Petrovic et al. [12] with the parameter settings used by the authors.

Before discussing how and why RI is combined with LSH, we will briefly go through each of these concepts in subsections 4.1 and 4.2. Finally, in subsection 4.3 we discuss combining RI with LSH to find previously encountered tweets similar to the input tweet.

4.1. Random Indexing (RI)

Random indexing [15] is a term vector based model for semantic similarity. It deals with the high dimensionality issue faced by other word co-occurrence based models, by performing random projection for dimensionality reduction.

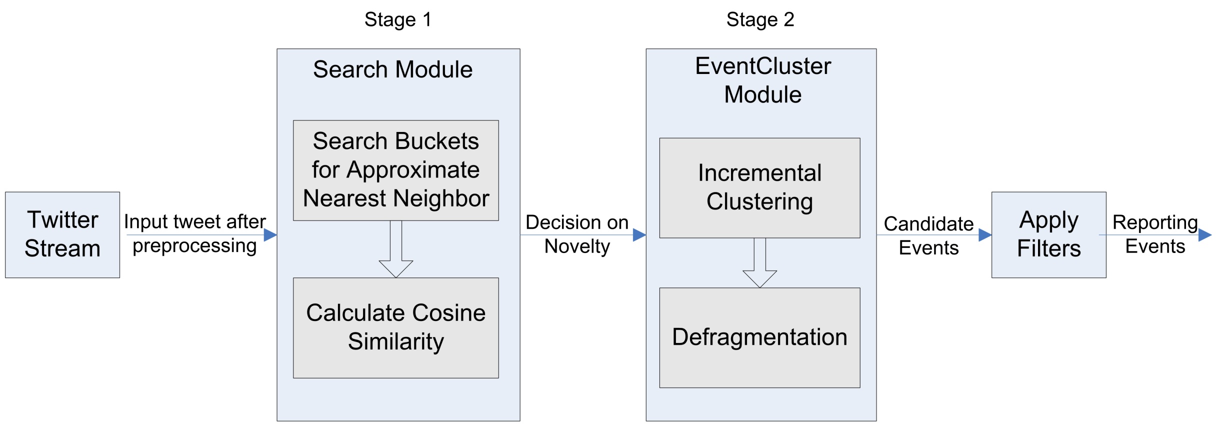

Figure 1. *TwitterNews* Architecture

Thus, making it a scalable approach. RI works by associating each term with two vectors: an index vector and a context vector. The index vector is a sparse, high dimensional, and unique random vector. In addition, 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. For each tweet in the system, a RI based incremental term vector model can be created, by processing each term of the tweet. Finally, a RI based vector representation for a tweet can be produced from an average of the vectors of each term. Having a RI 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.

4.2. Locality Sensitive Hashing (LSH)

The basic idea behind the LSH approach used by [12] 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:

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

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:

$$h_u(q) = sgn(u.q)$$

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,

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

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 (1). 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, L 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.

4.3. Combining RI with LSH

The calculation of L number of hash keys are computation intensive even with parallel processing. Generating a single hash key requires k number of independent random vectors whose dimensions are needed to be updated as more unique terms are encountered. This is done to match the dimensions of the input tweet vector which is based on an incremental $tf-idf$ weighting. To alleviate this problem we use the term vector model created with RI. This allows us to have a fixed length for every tweet vector and also the same length for the random vectors. Hence, the random vectors are created once at the initialization phase of the system, without requiring any further updates, as the dimensions for term vectors are unchanged regardless of the number of tweets encountered. Algorithm 1 shows the pseudocode for the Search Module using LSH and RI.

Algorithm 1 . TwitterNews Search Module

```
Require: $threshold$ value
 1: for each tweet $d$ in twitter-stream $D$ do
 2:     preprocess($d$)
 3:     generate vector for $d$ with $RI$
 4:     generate vector for $d$ with $tf - idf$
 5:     $S \leftarrow$set of documents that collide with $d$ in $L$ hash tables    ▷ hash calculated using RI based tweet vector
 6:     $sim_{max} \leftarrow 0$
 7:     for each tweet $d^{'}$ in $S$ do    ▷ parallel processing
 8:         $tempSim$ = CosineSimilarity($d$, $d^{'}$)    ▷ calculated using $tf - idf$ based tweet vector
 9:         if $tempSim > sim_{max}$ then
10:             $sim_{max}$ = $tempSim$
11:         end if
12:     end for
13:     if $sim_{max} > threshold$ then
14:         $d$ is "not unique"
15:     else
16:         $d$ is "unique"
17:     end if
18:     add $d$ in each colliding buckets of $L$ hash tables
19: end for
```

We have used the S-Space Package [21] to create RI based vector representation for the tweets, and a threshold value in the range of $[0.5-0.6]$ for cosine similarity is empirically set for the Search Module to determine the novelty of the input tweet. If the cosine similarity of the approximate nearest neighbor of the input tweet is below the threshold value, the input tweet is considered as "*unique*", otherwise "*not unique*".

Text similarity between two tweet vectors $d1$ and $d2$ in Algorithm 1 is computed using the cosine similarity measure:

$$\cos(\theta) = \frac{\sum_{i=1}^{n} d1_i \times d2_i}{\sqrt{\sum_{i=1}^{n}(d1_i)^2} \times \sqrt{\sum_{i=1}^{n}(d2_i)^2}}$$

In our experiments, we have found that the cosine similarity calculation is better suited with the $tf - idf$ based term vector model than the RI based vectors. Therefore we use the $tf - idf$ based term vectors for cosine similarity calculation and the RI based vectors are only used for the hash key calculation. To generate the $tf - idf$ based term vector for each tweet in Algorithm 1, the following formula is used:

$$tf - idf(t, d, D) = tf(t, d) \times idf(t, D)$$

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

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

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

5. The EventCluster Module

The EventCluster Module incrementally clusters the tweets discussing the same topic and produces a set of candidate events. Algorithm 2 shows the pseudocode for the EventCluster Module, where the event threshold (t_{ev}) value for a tweet to be assigned to a cluster is empirically set to 0.6 and the defragmentation granularity (g_{ev}) value to merge fragmented events is empirically set in the range of $[0.05 - 0.07]$.

The reason for sending only those tweets that are decided as "*not unique*" as input to the EventCluster Module, is to reduce noise. 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 members might be assigned and increasing the overall number of active clusters to be searched.

Although this strategy reduces the total number of clusters created, there are still a lot of clusters to search, 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.

The time complexity of Algorithm 2 is $O(m)$, where m is the number of clusters. However, from the point in

Algorithm 2 . TwitterNews EventCluster Module

```
Require: tweet $d$ decided as "not unique" by Algorithm 1, event threshold value $t_{ev}$, and defragmentaion granularity value $g_{ev}$
 1: $sim_{max} \leftarrow 0$
 2: for each active event cluster $c$ in $C$ do                                   ▷ parallel processing
 3:     $tempSim$ = findClosestCentroid($c$, $d_{termVector}$)  ▷ measures cosine similarity between the centroid of the cluster
        and the tweet term vector
 4:     if $tempSim > sim_{max}$ then
 5:         $sim_{max}$ = $tempSim$
 6:     end if
 7:     if $tempSim \geq (t_{ev} + g_{ev})$ then
 8:         $S_c \leftarrow$ assign $c$ to the set of fragmented clusters to be merged later
 9:     end if
10: end for
11: if $sim_{max} > t_{ev}$ then
12:     assign $d$ to the closest matching cluster $c_{simMax}$
13:     updateCentroid($c_{simMax}$, $d_{termVector}$)            ▷ cluster centroid updated by averaging with the tweet vector
14:     merge the clusters from the set $S_c$ with $c_{simMax}$
15:     update the centroid of $c_{simMax}$           ▷ cluster centroid updated by averaging with the event centroids in $S_c$
16:     update $c_{simMax}$ expiry time
17: else
18:     create a new cluster $c_{new}$ and assign $d$ to it
19:     assign $d_{termVector}$ as the centroid of $c_{new}$
20:     assign a initial expiry time to $c_{new}$
21: end if
```

time of system execution where the clusters start getting inactive, the total number of active clusters remain fairly constant. As we search only the active clusters, the cost of the EventCluster Module effectively becomes $O(1)$. Based on our experiments, we found that setting ts_i value in the range of $[8 - 15]$ minutes provides good results while reasonably limiting the total number of active clusters.

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. The defragmentation strategy is also helpful to merge clusters that are sub-events of an event. 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 $> t_{ev} + g_{ev}$, as shown in Algorithm 2. After we assign the tweet to the closest matching cluster (given that, similarity is $> t_{ev}$), all the clusters in S_c are merged to achieve defragmentation.

6. Experiment Results and Evaluation

Corpus. In our experiments, we have used the Events2012 corpus provided by McMinn et al. [22]. The corpus contains 120 million tweets from the 9^{th} of October to the 7^{th} of November, 2012. Along with this corpus 506 events, with relevant tweets for these events, are provided as a ground truth. Initially, the authors [22] have generated a set of candidate events using three different methods. The first two methods are the LSH approach of Petrovic et al. [12], and the Cluster Summarization approach of Aggarwal and Subbian [23]. The third approach is to simply extract the events found from the Wikipedia current event portal[2], which are within the dates covered by the corpus. Afterwards for each event found on the Wikipedia, relevant tweets of that event are retrieved from a Lucene[3] indexed version of the corpus. The candidate events produced by each of these three methods are combined using clustering and then the authors [22] have used crowdsourcing to generate the final set of 506 ground truth events for the time duration covered by the corpus.

Preprocessing. We have performed preprocessing on each tweet before it is sent to the Search Module. Each tweet is tokenized using Twokenize [24]. Tokens containing Username/mentions, stop words, and URLs are removed in the preprocessing phase as 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 (9^{th} to 11^{th} of October, 2012) 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 that will be reported by our system. The first level of filters retain the candidate events with an entropy > 2.5, and a user diversity > 0.0. The entropy threshold ensures that a minimum amount of information is contained in an event cluster, and a positive user diversity value ensures that the candidate event contains tweets from more than one user. Entropy [12] and User

2. http://en.wikipedia.org/wiki/Portal:Current_events
3. https://lucene.apache.org/

Diversity [20] value of an event cluster c, are computed as:

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

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.

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

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.

For the second level of filter, we 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 the news propagated by the general users or the news agencies usually follow a similar sentence structure. If we apply the traditional word-level LCS algorithm on the relevant tweets of an event cluster c, the length of the LCS can be used as a determining factor to decide whether c is about a newsworthy event. If an event cluster c has an LCS whose length is below a certain threshold, meaning there are no tweets in c with an appropriate level of similarity in their sentence structure, then 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. Table 1 shows the representative tweets of the newsworthy events reported by *TwitterNews*. Note that, only one event representative tweet from each day is shown to keep Table 1 concise.

TABLE 1. 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 Awards 2012"
10 Oct 2012	BAE-EADS merger plans are 'off': Aerospace and defence firms BAE and EADS have cancelled 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

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 the authors [22]. 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 (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.

As we have the reconfirmed ground truth events only for a specific time period, we ran our experiments on the tweets spanning the same time period from the corpus. We have used the LSH scheme proposed by Petrovic et al. [12] as our baseline, which achieved a recall of 0.52 by identifying 16 events out of the 31 ground truth events. McMinn et al., in their later work [14], have used the same baseline over the Events2012 corpus and reported the baseline system to achieve a very low precision. As calculating the precision is a time consuming task we have skipped this for our baseline which has been reported to have a low precision on the same corpus.

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. This high number of events obviously amount to a very low precision with respect to the ground truth. However, McMinn et al. [22] have noted in their later work [14] that a lot of additional events can be detected from the Events2012 corpus that were not originally on the ground truth data set they have provided along with the corpus. Hence, instead of calculating the precision with respect to the ground truth, we have used 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. The agreement between the two annotators, measured using Cohen's kappa coefficient, was 0.84 and the precision of *TwitterNews* reported by the annotators was 0.72 (72 out of 100 events were agreed as newsworthy events by both annotators). The results of our evaluation is shown in Table 2.

TABLE 2. SUMMARY OF THE EVALUATION RESULTS

Methods	Recall	Precision
Baseline [12]	0.52	-
TwitterNews	0.87	0.72

7. Conclusion

TwitterNews incorporates random indexing based term vectors with locality sensitive hashing 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 event related tweets. Both of the main components of our system maintain a constant space and processing time, which is an important requirement in a true streaming setting. We have shown that, the result achieved by *TwitterNews* outperforms the state-of-the-art baseline by a big margin. Note that, our exploration in the Events2012 corpus gave us a clear idea on the reason our system has missed some of the events from the ground truth event set. *TwitterNews* missed only those events, which are not bursty in nature, meaning, the event related tweets were spread out in the corpus with big time gaps between those tweets. In our future work, we will focus on detecting events that are less bursty in nature (e.g., events with usually non-measurable burst of few tweets, events having small amount of tweets with big gaps in successive tweet arrival). In addition, we intend to devise an approach that will allow us to discard spam and neutral event clusters in order to achieve a higher precision. Finally, we plan to perform a detailed sensitivity analysis on the different parameter settings used in our system.

References

[1] 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*, 2012, pp. 971–980.

[2] 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*, ser. CIKM '12. ACM, 2012, pp. 155–164.

[3] 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*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 227–236.

[4] M. Mathioudakis and N. Koudas, "TwitterMonitor: Trend detection over the Twitter stream," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 1155–1158.

[5] 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.

[6] R. Parikh and K. Karlapalem, "ET: Events from tweets," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13 Companion. New York, NY, USA: ACM, 2013, pp. 613–620.

[7] J. H. Lau, N. Collier, and T. Baldwin, "On-line trend analysis with topic models:\# Twitter trends detection topic model online." in *Proceedings of the International Conference on Computational Linguistics, COLING*, 2012, pp. 1519–1534.

[8] Y. Hu, A. John, D. D. Seligmann, and F. Wang, "What were the tweets about? topical associations between public events and Twitter feeds." in *Proceedings of the International AAAI Conference On Web and Social Media, ICWSM*, 2012, pp. 154–161.

[9] 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*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 1104–1112.

[10] 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*, 2015, pp. 2468–2475.

[11] 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*, vol. 7, no. 13, pp. 1637–1640, 2014, VLDB Endowment.

[12] 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*, ser. HLT '10. Stroudsburg, PA, USA: ACL, 2010, pp. 181–189.

[13] F. Atefeh and W. Khreich, "A survey of techniques for event detection in Twitter," *Computational Intelligence*, vol. 31, no. 1, pp. 132–164, 2015, Wiley Online Library.

[14] 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*, ser. CLEF '15. Springer, 2015, pp. 65–77.

[15] 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.

[16] 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*, ser. SIGMOD '11. New York, NY, USA: ACM, 2011, pp. 1271–1274.

[17] 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.

[18] 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*, 2013, pp. 198–206.

[19] 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*, vol. 15, no. 6, pp. 1268–1282, 2013, IEEE.

[20] 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.

[21] D. Jurgens and K. Stevens, "The S-Space package: an open source package for word space models," in *Proceedings of the ACL System Demonstrations*. ACL, 2010, pp. 30–35.

[22] 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*, ser. CIKM '13. New York, NY, USA: ACM, 2013, pp. 409–418.

[23] C. C. Aggarwal and K. Subbian, "Event detection in social streams." in *Proceedings of the SIAM International Conference on Data Mining, SDM*, vol. 12. SIAM, 2012, pp. 624–635.

[24] 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*, ser. HLT '13. ACL, 2013, pp. 380–391.