Exploring a Scalable Solution to Identifying Events in Noisy Twitter Streams

Shamanth Kumar†, Huan Liu†, Sameep Mehta⋆, and L. Venkata Subramaniam⋆
† Computer Science & Engineering, CIDSE, Arizona State University, Tempe, AZ
⋆ IBM India Research Lab, New Delhi, India
Email:{ shamanth.kumar, huan.liu}@asu.edu, {sameepmehta, lvsubram}@in.ibm.com

***Abstract*—The unprecedented use of social media through smartphones and other web-enabled mobile devices has enabled the rapid adoption of platforms like Twitter. Event detection has found many applications on the web, including breaking news identification and summarization. The recent increase in the usage of Twitter during crises has attracted researchers to focus on detecting events in tweets. However, current solutions have focused on static Twitter data. The necessity to detect events in a streaming environment during fast paced events such as a crisis presents new opportunities and challenges. In this paper, we investigate event detection in the context of real-time Twitter streams as observed in real-world crises. We highlight the key challenges in this problem: the informal nature of text, and the high-volume and high-velocity characteristics of Twitter streams. We present a novel approach to address these challenges using single-pass clustering and the compression distance to efficiently detect events in Twitter streams. Through experiments on large Twitter datasets, we demonstrate that the proposed framework is able to detect events in near real-time and can scale to large and noisy Twitter streams.**

I. Introduction

Social networking sites like Twitter have proven to be popular outlets for information dissemination during crises. It has been observed that information related to a crisis is released on social media sites before traditional news sites [1]. This motivates us to study the problem of event detection, which is an interesting and important problem in this domain.

Event detection approaches designed for documents cannot be directly applied to tweets due to the difference in the characteristics of tweets. Unlike a traditional document stream, a Twitter stream suffers from the informality of language, and differs in both volume and velocity characteristics. Existing approaches to event detection in tweets focus on the problem in an offline setting, where the corpus is static and multiple passes can be employed in the solution. However, event detection in streaming environment presents unique challenges, which prevent the direct application of existing approaches. Detecting events in streaming Twitter data have the following new challenges: **Informal use of language:** Twitter users produce and consume information in an informal manner [2]. Misspellings, abbreviations, contractions, and slang are rampant in tweets, which is promoted by the length restriction (a tweet can have no more than 140 characters). **Noise:** While traditional event detection approaches assume that all documents are relevant, Twitter data typically contains a vast amount of noise and not every tweet is related to an event [3]. **Dynamicity:** Twitter streams are highly dynamic with high volume and high velocity. Approximately 400 million tweets are now posted on Twitter every day [4]. Event detection methods need to be scalable to handle this high volume of tweets.

Social media such as Twitter empower their users to publish information as soon as a real-world event occurs. However, this information is not curated as in the case of traditional documents, such as a news article. Whereas, each news article is part of an event, not every tweet is expected to be part of an event, as there is a significant amount of noise and interpersonal communication. In this paper, we address the above challenges through a novel approach which can:

- Effectively handle the informality of language in a Twitter stream through an appropriate distance measure;
- Capture evolving events without the need for labeled events; and
- Scale to high-volume streaming Twitter data.

II. Related Work

Event detection in traditional media is known as Topic Detection and Tracking (TDT). In [5], news articles were modeled as documents to detect topics. The documents were transformed into vector space using TF-IDF and evaluated the Group-Average Agglomerative Clustering (GAAC) for retrospective event detection, and Incremental Clustering for new event detection. Allan et al. [6] focused on online event detection. The authors constructed a query from the k most frequent words in a story. If a new document did not trigger any existing queries, then it was considered to be part of a new event. In [7], the authors addressed the problem of detecting *hot* bursty events. They introduced a parameter-free clustering approach called feature-pivot clustering, which attempted to detect and cluster bursty features into hot stories. Similarly, [8] interpreted events as hashtag clusters and propose a hierarchical spatio-temporal clustering of tweets into events.

An attempt to detect earthquakes using Twitter users as social sensors was made in [9]. The temporal aspect of an event was modeled as an exponential distribution, and the probability of the event was determined based on the likelihood of each sensor being incorrect. In [10], the authors constructed word signals using the Wavelet Transformation and applied a modularity-based graph partitioning approach on the correlation matrix to get event clusters. [11] identified bursty segments in tweets and clustered them to identify events.

In [12], the authors model the social text streams including blogs and emails as a multi-graph and cluster the streams using textual, temporal, and social information to detect events. A hybrid network and content based clustering approach was

employed in [13] to identify a fixed number of events in a labeled Twitter stream. Typically, the number of events is unknown and obtaining the user network is expensive. In [14], the authors recognized the need for faster approaches for first story detection in streams. They proposed a two-step process to identify first stories in streaming data. First, the nearest neighbor of each tweet is identified using locally sensitive hashing in constant time and space. Second, a clustering approach called Threading is applied to group related tweets into event clusters. Here a thread is considered an event. Subsequently, this was converted to a distributed method in [15].

III. Problem Definition

Given an ordered stream of tweets $T = t_1, t_2, t_3, ...,$ where each t_i is associated with a timestamp indicating its publication time, we need to detect events $E = e_1, e_2, ...e_m$, where $e_j = t_1, t_2, ..., t_k$, where $t_k \in T$ and $j \in [1, m]$.
Event: An event is formally defined as a set of similar tweets $E = t_1, t_2, ..., t_k$ with high user diversity.
User Diversity: of an event is the diversity of the user population who contribute to the event. The intuition here is that a diverse user population lends credibility to the event and helps us filter out noise. Entropy is a measure commonly used to compute the amount of information in a text and here we reformulate it to measure user diversity of an event. Given an event e, its *User Diversity* $H(e)$ is

$$H(e) = -\sum_i \frac{n_{u_i}}{n} log \frac{n_{u_i}}{n}, \quad (1)$$

where u_i is the ith user in the cluster. Here, n_{u_i} is the number of tweets published by user u_i, which are part of the cluster C, and n is the total number of tweets in the cluster.

IV. Identifying Events

People use Twitter to tweet and retweet their experiences. This published information may be aggregated/clustered to detect events. For streaming Twitter data, we must consider that (1) traditional multi-pass clustering cannot be applied, and (2) the informal language of tweets defies the standard preprocessing of text such as stemming and vectorization.

To handle high-volume and high-velocity streams, clustering approaches must return the clusters in a single pass. We seek to avoid multiple passes over data. In this paper, we employ a single-pass clustering technique to group related tweets into clusters as they arrive. Additional information on the method and additional results can be found in [16]

To cluster the tweets, we must choose an appropriate distance measure which is scalable to high-volume streaming data; avoids the need for expensive data transformations, be robust to informal language, and avoids the maintenance of a vocabulary as the language is constantly evolving.

A. Tackling Data Informality

We use the compression distance to compute the distance between two texts by measuring the compression gain achieved from merging the two texts. This distance measure has been shown to be both efficient and effective for clustering text in [17]. Unlike traditional measures, such as cosine similarity which requires the maintenance of a vocabulary and data transformation, compression distance can be directly applied to text. In this paper, we consider each tweet as a document and use the compression distance to compute the distance between two events $D(e_1, e_2)$ as the maximum pairwise distance between any pair of tweets in the events. We use DEFLATE as the compressor to compute the distance.

B. Scaling to High-Volume Data

Twitter users currently generate more than 400 million tweets a day [4]. Using publicly available Twitter APIs, one can access a sample of (1%) tweet stream, which can lead to as many as several million tweets a day. Thus, detecting events in a stream necessitates a scalable solution. Here, we present detailed solutions to scalability.

Events are dynamic and it is essential to consider the temporal evolution of the events in streaming data. As we create clusters of tweets, they be active or inactive at a given time based on the arrival of new tweets. Here, we propose a temporal model which can be used to make this decision. We model events as a Poisson process, which have been traditionally used to model the number of objects in an event at time t. In a Poisson process, the rate of arrival of tweets can be modeled as an exponential distribution. This rate is represented by the parameter λ.

Let us consider a random variable X, where X measures the time between successive tweets. The variable X is modeled as an exponential distribution with the parameter λ as $X \propto exp(\lambda)$. Given an event e and the number of tweets in each time interval in the event $x_1, x_2, ..., x_n$, the likelihood function for the inter-arrival time is

$$L(\lambda | x_1, x_2, ..., x_n) \propto f(x_1, x_2, ..., x_n | \lambda) \propto \prod_{i=0}^{n} \lambda e^{-x_i \lambda}. \quad (2)$$

To obtain λ_{MLE}, by taking the derivative of the log-likelihood with respect to λ and setting it to zero. Then, $\lambda_{MLE} = \frac{1}{\bar{x}}$, where $\bar{x}$ is the mean of the distribution. For each cluster c, if a tweet does not arrive in λ_c time units, the current estimate for cluster c, then c is considered inactive and removed from memory. The estimate for λ_c is updated every time a new tweet is added to the cluster.

C. Identifying Events from Noisy Clusters

Tweets are noisy and not every tweet in the stream is expected to be part of an event. Therefore, not every cluster identified by the algorithm can be an event. The volume of a cluster can help us identify events, but this is susceptible to noise. As a crowdsourced information sharing platform, the diversity of the users (or the number of unique users) who publish tweets in a cluster lends credibility to the information within the cluster. Therefore, we measure the user diversity of a cluster to determine whether it is an event. A cluster is classified as an event, if its Diversity Score $H(c)$ is above the *Diversity Threshold* H_t.

V. Event Detection Framework

Using the strategies to handle informal language in tweets, temporal dynamics of events, and handling noise, we can

Algorithm 1: Event Detection in Twitter Streams

Input: A stream of tweets T and the Cluster Limit (k), the Tweet Limit (l), the Distance Threshold (D_t), and the Diversity Threshold (H_t).
Output: Detected events E.

```
E ← {};
C ← {};
while tweet t ∈ T do
    Identify active cluster c ∈ C, where D(t, c) ≤ D_t;
    if c exists then
        Add t to c;
        Update expected time of next tweet λ_c;
        Update User Diversity (H(c)) of cluster c;
        if H(c) ≥ H_t then
            Mark cluster as an event, Add c to E;
        end
    end
    else
        Create new cluster c with t as its first member;
        Add c to C;
        5
    end
end
```

detect events in Twitter streams. A high-level description of the algorithm is presented in Algorithm 1. To improve the efficiency of the algorithm and to scale it to large Twitter streams we propose two heuristics: *Cluster Limit:* sequentially searching through all active clusters can be prohibitive. As a tweet is more likely to be similar to clusters with overlapping content we limit the comparisons to these candidate clusters. We pick the top k clusters as the candidate clusters. Here $k = 100$. *Tweet Limit:* due to the timely nature of tweets we restrict the comparisons to at most l recent tweets in a cluster. In our implementation, we set $l = 1000$.

Time complexity: given the number of tweets in the stream as N, the Cluster Limit (k) and the Tweet Limit (l), the time complexity of our algorithm is $O(Nkl)$. The most expensive operation in our algorithm is the assignment of tweets to clusters. The values of k and l are much smaller than N, thus allowing us to process the tweets in near real-time. We will present empirical evidence in support of this claim lated.

Selection of parameters: two thresholds are used in our framework to identify events. First, the distance threshold D_t is used to determine assignment of tweets to clusters. In a study on 20,000 random tweets, we found that the average self-similarity of tweets was 0.54 and a value of 0.8 was empirically found to be a suitable value to obtain reasonable clusters. Second, the diversity threshold H_t, which is used to decide which clusters can be labeled as events to remove noise. This threshold is set empirically as outlined later.

VI. Evaluation Strategy

There are two challenges in evaluating events from Twitter: 1) unlike traditional media such as broadcast news, where every event is reported, on Twitter there is less likelihood of finding tweets related to minor events, and 2) while traditional research on event detection has relied upon the availability of labeled corpora such as the TDT corpus for evaluation, no such corpus exists for Twitter. Therefore it is difficult to determine the exact number or nature of the events and manual labeling is impractical. In this section we evaluate our approach on two forms of Twitter streams: **topic streams** containing tweets related to a specific topic, where the number and type of events can be verified using external sources, and **random streams**, which contain randomly sampled tweets, where the number and type of events must be manually determined.

TABLE I. Efficiency of event detection: Earthquake

Day	#tweets	Total processing time (Min)	Collection rate (Tweets/Min)	Processing rate (Tweets/Min)
7/19/2011	880	0.04	0.613	23,498.00
9/5/2011	2,712	0.18	1.88	14,788.69
9/18/2011	465	0.02	0.32	18,699.73
10/23/2011	5,253	0.49	3.65	10,646.89
11/9/2011	2,712	0.17	1.89	15,611.63
2/6/2012	13,586	4.79	13.72	2,834.92
4/11/2012	28,182	10.61	19.57	2656.06
5/20/2012	20,509	6.40	14.33	3,204.44

TABLE II. Events detected in the Earthquake dataset

Day	Earthquake Location	Key Event Terms
9/5/2011	Indonesia	sumatra, western, indonesian, island, #breakingnews
10/23/2011	Turkey	#turkey, eastern, turkey, magnitude, news
11/9/2011	Turkey	turkey, eastern, magnitude, rocks, usgs
2/6/2012	Philippines	pray, visayas, philippines, struck, earlier
4/11/2012	Indonesia	#indonesia, tsunami, magnitud, indonesia, sacudi
5/20/2012	Italy	sentito, emilia, sono, cosa, chies

A. Detecting Events in Topic Streams

As a representative topic stream, we introduce the Earthquake topic stream which consists of tweets related to earthquakes around the world. Due to the existing research demonstrating the use of Twitter during earthquakes [9], [18], we collected tweets referring to earthquakes between June, 2011 to May, 2012 by monitoring the hashtags: #earthquake, #terremoto, and #quake using the Twitter streaming API and the techniques recommended in [19]. The data comprises of 1,007,417 tweets from 317,564 users. In general, the volume of tweets during the earthquakes in 2011 was significantly lower than those in 2012.

To obtain ground-truth we seek an independent and external source. One such source is Wikipedia where the information is curated by the community. Therefore, we select the major earthquakes in 2011 [20] and 2012 [21] listed on Wikipedia as the ground-truth. These reports were manually compiled from several major news sources. Here, we focus on 10 days when a major earthquake resulted in at least 10 casualties. We found that for most events in 2011, only a few hundred tweets were collected. Therefore, we set $H_t = 5$ for this dataset.

1) Evaluating Scalability: To verify the scalability of our approach we compare tweet processing rate with the tweet generation rate. Table I presents the results of the comparison for the Earthquake dataset. We find that the approach is capable of handling high-volume topic-specific Twitter streams by being able to process the tweets at a rate higher than the rate at which tweets are generated.

2) Quality of Detected Events: Events are typically described using the frequent keywords extracted from the tweets [5], [14], [7]. We employ the same procedure to verify

TABLE III. EFFICIENCY OF THREADING TECHNIQUE: EARTHQUAKE

Day	#tweets	Total Processing Time (Min)	Tweet collection rate (Tweets/Min)	Processing rate (Tweets/Min)
7/19/2011	880	1.11	0.613	793.40
9/5/2011	2,712	3.99	1.88	678.68
9/18/2011	465	0.88	0.32	527.10
10/23/2011	5,253	2.65	3.65	1,984.97
11/9/2011	2,712	2.54	1.89	1,068.13
2/6/2012	13,586	38.36	13.72	354.19
4/11/2012	28,182	135.27	19.57	208.34
5/20/2012	20,509	210.32	14.33	97.51

whether the detected events matched the ground-truth. In Table II, we present the most representative event corresponding to the known events.

To quantify the effectiveness of our approach, we compute the F_1 score which captures both the Precision and Recall. Precision is computed as the number of detected events that match the ground truth including sub-events. Recall is computed as the number of known events which were successfully detected. The F_1 Score for the Earthquake dataset was 0.77.

B. Detecting Events in a Random Stream

Twitter streams can also be collected without any topic bias using the Twitter Sample API. The task of event detection is harder in this case due to the presence of noise in the form of interpersonal tweets. We collected a 1% random sample of tweets from 11:02 AM on Apr 15 to 9:16 AM on Apr 16, 2013. The data consisted of 4,212,333 tweets from 3,322,379 users. Due to the lack of ground truth, we verify that we can detect the top stories of the day. Here we set $H_t = 6.3$.

Evaluating scalability: to evaluate scalability we once again compare the tweet generation rate and the tweet processing rate and found that the processing speed for a majority of the processing speed was on par with the collection speed and it often exceeded the tweet collection rate significantly.

Quality of detected events: we detected 167 events in this corpus. But, a manual investigation revealed 4 major kinds of events: the Boston bombing incident, the Presidential elections in Venezuela, a music festival, and events representing banal Twitter chatter, such as those of Justin Bieber fans.

VII. DISCUSSION AND FUTURE WORK

We compare the performance of the proposed technique with the *Threading* technique proposed in [14]. Using the configuration recommended by the authors, we applied this technique to the Earthquake dataset. The results for the scalability experiment are presented in Table III. A comparison with our approach in Table I shows that we can process and detect events faster. Next, we evaluate the quality of the events. On all days, the Threading approach detected a greater number of events. Even using the ranking strategy proposed by the authors to retrieve the top 10 fastest growing events, we found that the F_1 score for the Threading technique was 0.64 compared to 0.77 for the proposed approach. As the Earthquake dataset was the smallest among our datasets, the results show that our framework outperforms this approach.

Event detection has several potential applications, which we intend to investigate as part of our future work. Investigating the relationship between an event's rate of growth and its impact in the real-world is one. Another direction of future study is the categorization of events based on two characteristics: the volume of the event defined by the number of tweets, and the rate of the event. By organizing the events in this fashion, we can provide a value added service to users by facilitating the tracking of specific types of events.

ACKNOWLEDGMENTS

This work was sponsored, in part, by the Office of Naval Research grant N000141410095.

REFERENCES

[1] D. Gilgoff and J. J. Lee, "Social Media Shapes Boston Bombings Response," http://bit.ly/1iESExb, Apr. 2013, [Online; accessed 27-February-2015].

[2] C. Paris, P. Thomas, and S. Wan, "Differences in Language and Style Between Two Social Media Communities," in *ICWSM*, 2012.

[3] "Twitter Study," http://www.pearanalytics.com/wp-content/uploads/2012/12/Twitter-Study-August-2009.pdf, 2009, [Online; accessed 27-February-2015].

[4] H. Tsukayama, "Twitter turns 7: Users send over 400 million tweets per day," http://articles.washingtonpost.com/2013-03-21/business/37889387_1_tweets-jack-dorsey-twitter, 2013, [Online; accessed 25-February-2015].

[5] Y. Yang, T. Pierce, and J. Carbonell, "A Study of Retrospective and On-Line Event Detection," in *SIGIR*. ACM, 1998, pp. 28–36.

[6] J. Allan, R. Papka, and V. Lavrenko, "On-Line New Event Detection and Tracking," in *SIGIR*. ACM, 1998, pp. 37–45.

[7] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu, "Parameter Free Bursty Events Detection in Text Streams," in *VLDB*. VLDB Endowment, 2005, pp. 181–192.

[8] W. Feng, J. Han, J. Wang, C. Aggarwal, and J. Huang, "STREAMCUBE: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration over the Twitter Stream," in *ICDE*. IEEE, 2015.

[9] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake Shakes Twitter Users: Real-Time Event Detection by Social Sensors," in *WWW*. ACM, 2010, pp. 851–860.

[10] J. Weng and B. S. Lee, "Event Detection in Twitter," in *ICWSM*, 2011.

[11] C. Li, A. Sun, and A. Datta, "Twevent: Segment-Based Event Detection from Tweets," in *CIKM*. ACM, 2012, pp. 155–164.

[12] Q. Zhao, P. Mitra, and B. Chen, "Temporal and Information Flow Based Event Detection From Social Text Streams," in *AAAI*, vol. 7, 2007, pp. 1501–1506.

[13] C. C. Aggarwal and K. Subbian, "Event Detection in Social Streams," in *SDM*, 2012, pp. 624–635.

[14] S. Petrovic, M. Osborne, and V. Lavrenko, "Streaming First Story Detection with Application to Twitter," in *HLT workshop at NAACL*, vol. 10. Citeseer, 2010.

[15] R. McCreadie, C. Macdonald, I. Ounis, M. Osborne, and S. Petrovic, "Scalable Distributed Event Detection for Twitter," in *Big Data*. IEEE, 2013, pp. 543–549.

[16] S. Kumar, H. Liu, S. Mehta, and L. V. Subramaniam, "From Tweets to Events: Exploring a Scalable Solution for Twitter Streams," *CoRR*, vol. abs/1405.1392, 2014. [Online]. Available: http://arxiv.org/abs/1405.1392

[17] E. Keogh, S. Lonardi, and C. Ratanamahatana, "Towards Parameter-Free Data Mining," in *KDD*. ACM, 2004, pp. 206–215.

[18] M. Mendoza, B. Poblete, and C. Castillo, "Twitter Under Crisis: Can We Trust What We RT?" in *Proceedings of the First Workshop on Social Media Analytics*, 2010.

[19] S. Kumar, F. Morstatter, and H. Liu, *Twitter Data Analytics*. Springer, 2014.

[20] "Earthquakes in 2011," http://en.wikipedia.org/wiki/Earthquakes_in_2011, 2011, [Online; accessed 27-February-2015].

[21] "Earthquakes in 2012," http://en.wikipedia.org/wiki/Earthquakes_in_2012, 2012, [Online; accessed 27-February-2015].