Streaming Event Detection in Microblogs: Balancing Accuracy and Performance

Ozlem Ceren Sahin[1], Pinar Karagoz[1], and Nesime Tatbul[2]

[1] METU, Ankara, Turkey
{e1746668,karagoz}@ceng.metu.edu.tr
[2] Intel Labs and MIT, Cambridge, MA, USA
tatbul@csail.mit.edu

Abstract. In this work, we model the problem of online event detection in microblogs as a stateful stream processing problem and offer a novel solution that balances result accuracy and performance. Our new approach builds on two state of the art algorithms. The first algorithm is based on identifying bursty keywords inside blocks of blog messages. The second one involves clustering blog messages based on similarity of their contents. To combine the computational simplicity of the keyword-based algorithm with the semantic accuracy of the clustering-based algorithm, we propose a new hybrid algorithm. We then implement these algorithms in a streaming manner, on top of Apache Storm augmented with Apache Cassandra for state management. Experiments with a 12M tweet dataset from Twitter show that our hybrid approach provides a better accuracy-performance compromise than the previous approaches.

Keywords: Online event detection · burst detection · stream processing · data stream management · microblogging.

1 Introduction

The emergence of microblogging services such as Twitter has caused a revolution in the way information is created and exchanged on the web [16]. Microblogs are user-generated short messages, typically in textual format. Twitter is the most popular microblogging service provider, with more than 300M monthly active users posting more than 500M tweets every day. As such, it constitutes a rich source of information for wide range of use, from market studies to real-time dissemination of breaking news. There has been a plethora of research in analyzing social media data, including microblogs posted on Twitter [5], [17], [3], [27]. In this paper, we focus on one particular form of social media data analysis, that is, *event detection*. We consider an *event* as a happening that takes place at a certain time and place, causing a short window of sudden burst in attention from the microbloggers. The capability to accurately detect events as soon as they happen, i.e., in an *online* fashion, can be important in many ways, from timely access to interesting news to tracking life-critical phenomena such as natural disasters [20], [11].

In this work, we hypothesize that online event detection is fundamentally a stream processing problem. Stream processing systems have been around for more than a decade, and today many mature, industrial-quality platforms are publicly available [1], [6]. These systems are highly tuned for low-latency / high-throughput processing over real-time data, making them ideal base platforms for building online event detection dataflows. Furthermore, we believe that capturing events accurately (i.e., no false positives or false negatives) is as equally important as detecting them with low latency. Unfortunately, while online event detection in social media has seen much attention from the research community [24], [5], [22], [23], [29], [2], [7], solutions that aim at addressing both accuracy and performance are limited to only a couple [13], [28].

In this paper, we explore the tradeoff between event detection accuracy and performance through stream-based design and implementation of two well-known algorithms from the literature, with opposite characteristics: a keyword-based algorithm and a clustering-based algorithm. The keyword-based algorithm is purely a syntactical approach in that, it is based on counting the occurrence of words, without paying attention to their meanings or relationships. The clustering-based algorithm, on the other hand, groups tweets by the similarity of their contents. While the former is simpler and faster, the latter is expected to produce more accurate results at the expense of taking a longer time to compute. As a novel contribution, we then propose a two-phase, hybrid algorithm, which first applies the keyword-based algorithm as an initial filtering phase, followed by the clustering-based algorithm as the final event detection phase. All of our techniques have been implemented on top of the Apache Storm distributed stream processing system augmented with the Apache Cassandra key-value store for state management, and have been experimentally tuned and evaluated based on a 12M tweet dataset that we collected from Twitter. We find that there is a clear accuracy-performance tradeoff between the keyword-based approach and the clustering-based approach. Moreover, the experiments verify our intuition that the hybrid approach can provide a good compromise between the two. More specifically, this work makes the following contributions:

- Parallel, stateful, stream-based design and implementations of two state of the art algorithms for online event detection,
- A new hybrid algorithm that combines the advantages of these two algorithms to balance event detection accuracy and performance,
- A detailed experimental evaluation based on Apache Storm and Apache Cassandra, using a real Twitter workload,
- Revealing of new research directions for improving both the algorithmic and the systems components of the problem space.

In the rest of this paper, we first present our event detection methods, their implementation, and experimental evaluation in Sections 2, 3, and 4, respectively. We then summarize related work in Section 5, and conclude the paper with a discussion of future directions in Section 6.

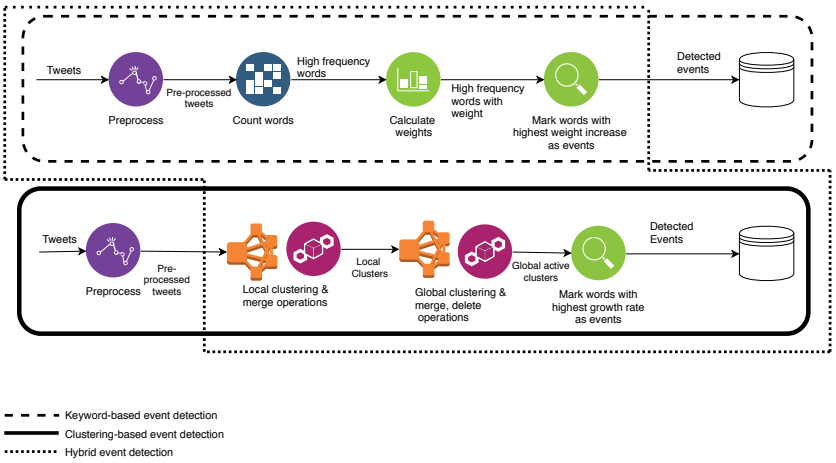


Fig. 1: Event Detection Methods

2 Event Detection Methods

In this section, we present three different event detection methods applied on streaming data. The first two are based on state of the art algorithms from the event detection literature, and the third is a hybrid extension of these that we newly propose in this paper. In all three methods, streaming posts are processed in windows of predefined time intervals, which are called *rounds*. In each round, the stream is processed as a chunk of tweets, resulting in a state. Hence, we can consider event detection as a state that is reached according to the change between two consecutive states.

For all three methods, first the following preprocessing step is applied: First, messages are tokenized and stemmed through an NLP parser [3]. Afterwards, stop words and geo-references including the phrase *"I am at"* are eliminated. Additionally, we normalize the words having characters that repeat more than two times (such as *"goooooaaaal!"*). Applying stemming and normalization are particularly important for aggregating the different occurrences of the same word.

2.1 Keyword-based Event Detection Method

Our first method relies on detecting the unexpected increase in the occurrence or observation of the words with respect to a previous round. Then, such *bursty* words are considered to express an event. This method consists of three main steps that are applied in every round: *word counting*, *word weight calculation*, and *event detection* (see Figure 1 for an overview). Hence, at the end of each round a set of event keywords are obtained.

[3] We use the Stanford NLP parser: https://nlp.stanford.edu/software/lex-parser.shtml

Word counting. Microblog postings are very short texts due to character limit. Hence, we consider the set of postings in the same round as a single document. As the postings are received from the input stream, the stemmed and normalized words are counted. In order to limit time and space complexity in the following steps, we eliminate the words whose frequency is below a given threshold.
Word weight calculation. Using word counts (i.e., frequency of words) may be misleading for detecting bursts, as some of the words may be appearing in any context. In order to normalize this effect, we measure the weight of the words in terms of *tf-idf*, instead of frequency [25]. Since all the tweets in a round are considered as a single document, frequency of a word denotes its frequency in the round.
Event detection. In order to check the increase in observation of a word, we compare the weight of the words in terms of tf-idf values in consecutive rounds. The increase is compared against a threshold in order to be considered as an *event related word*.

2.2 Clustering-based Event Detection Method

In the clustering-based method, the basic assumption is that a *cluster of tweets* with *high growth rate* corresponds to an event. As in the keyword-based method, each round is processed one by one and the resulting clusters are compared for event detection.

The method is composed of two basic steps: *cluster formation* and *event detection* (see Figure 1 for an overview). In each round, these steps are applied in sequence.
Cluster formation. We use four basic cluster operations: *creating a new cluster*, *updating a cluster*, *merging clusters*, and *deleting a cluster*. Each cluster has a representative term vector, which includes the frequent terms of the tweets in the cluster. Similarly, each tweet is represented by a term vector of stemmed words in the tweet. Hence, similarity of a tweet to an existing cluster is measured with cosine similarity between term vectors under a predefined threshold. As the tweets are received in a round, one of the cluster operations is applied.

- If the tweet content is not similar to any of the clusters, a *new cluster* is created.
- If the tweet content is similar to a cluster, then the *cluster is updated* by including the tweet in the cluster and updating the cluster's representative term vector.
- *Cluster merging* is applied in two stages. First, within each round, clusters are generated locally (i.e., only considering the tweets in the current round). Furthermore, at the end of a round, similar local clusters are merged. As the second stage of merging, the resulting local clusters are merged with the global clusters (i.e., the cumulative set of active clusters since the beginning of time) complying with the similarity threshold.
- In order to reduce the number of generated clusters, and hence improve execution time performance, *inactive clusters are deleted*. The condition for

deletion is defined as follows: If a cluster is not active (i.e., not updated) for the last two rounds, then it is deleted.

Event detection. In the clustering-based method, event detection is achieved through tracking the growth rate of clusters. The growth rate is calculated using the number of tweets that contribute to a cluster, as the ratio of the number of tweets added to the cluster to the total number of tweets in the cluster. To mark a cluster as an event, the *cluster growth rate* should be greater than a predefined threshold.

2.3 Hybrid Event Detection Method

The hybrid method combines the previous two methods in order to increase the efficiency of clustering by filtering tweets that do not include bursty keywords. First, bursty keywords are found by applying the steps used in the keyword-based event detection method, and then clustering is applied on tweets containing the bursty keywords. Finally, by using the cluster growth rates, this technique marks clusters as events (see Figure 1 for an overview; notice how the tweets with bursty words found by the keyword-based method is fed as an input to the clustering-based method).

Tweet filtering. As in the previous methods, words in a streaming tweet are tokenized and stemmed. Words are counted and those with low frequency are eliminated. By this elimination, we reduce the number of words to keep track of for burstiness. As in the first method, burstiness of a word is checked through the increase in its tf-idf value. The increase is compared against a predefined threshold in order to be considered as bursty keyword. Additionally, if the tf-idf value of a word is very high for only the last round, then we consider it as a bursty term as well.

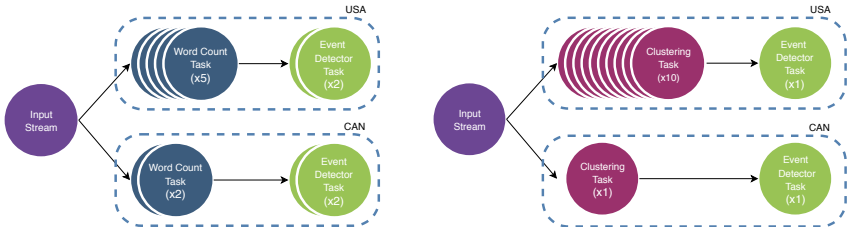
Clustering. The hybrid method uses the same clustering technique as in the clustering-based method. Similarly, two-level clustering is applied, local and global. The basic difference here is that, in a round, instead of clustering all streaming tweets, only those that include any bursty term are fed into the clustering phase. By this way, time efficiency can be significantly improved.

Detecting events using clusters. As the final step of the hybrid technique, event detection is performed by checking the growth rate of the cluster as applied in the clustering-based method.

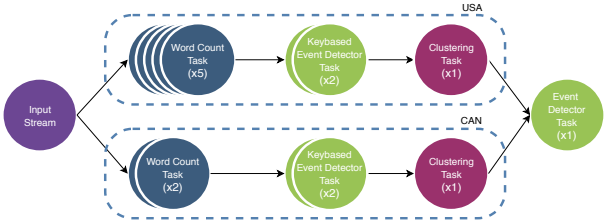
3 Implementation

The three event detection methods studied in this work are implemented on the Apache Storm stream processing framework [4]. In Storm, an application is defined as a *topology*. A topology is an arbitrarily complex multi-stage stream computation. It is a graph of spouts and bolts, such that a *spout* is a source of stream and a *bolt* is a stream processing task.

[4] http://storm.apache.org/



(a) Keyword-based Topology (b) Clustering-based Topology



(c) Hybrid Topology

Fig. 2: Storm Topologies

We use a single stream of tweets as our spout. In the experiments, we worked on the tweet stream filtered by geographical boundary. We limited the stream to the tweets posted in USA and Canada. Furthermore, we processed these two groups of tweets in parallel. Alternatively, it is possible to consider two groups of tweets as two separate streams, as well. We store the streamed data collection in the Apache Cassandra key-value store [5] to maintain each round of tweets for further processing in bolts. Within this spout, tokenization of the messages and stemming of the words are performed as well. The stream is emitted as a set of preprocessed words to the following bolt.

3.1 Keyword-based Event Detection Topology

The structure of the keyword-based event detection topology is presented in Figure 2a. The topology basically includes a single spout (denoted as *input stream*), and two types of bolts: word count bolt and event detector bolt. As described in Section 2, the tweet stream is processed in terms of rounds. We can consider the tweets in the same round as a single document. Word count bolt keeps track of the count of the words in the document of a round. When frequency counting is completed, event detector bolt calculates tf-idf values for the words, and compares the value with that of the previous round to determine *bursty keywords*.

[5] http://cassandra.apache.org/

The topology given in Figure 2a shows the parallelism and distribution applied in the experimental analysis. The upper path in the topology graph includes the bolts reserved for tweets from USA, whereas the lower path is for processing tweets from Canada. The figure shows the number of bolts allocated for each task, as well (the numbers given in parenthesis at the end of task label). The number of bolts to be allocated are determined empirically according to the work load on the task.

3.2 Clustering-based Event Detection Topology

The structure of the clustering-based event detection topology is presented in Figure 2b. As in the keyword-based approach, there is a single spout (denoted as *input stream*), and two types of bolts, but the tasks of the bolts are different than those in the previous topology.

Clustering bolt is responsible for clustering the incoming tweets on the basis of the similarity between the tweet and cluster representative vector. *Event detector bolt* keeps track of change in the size of the clusters to detect a burst in the size (hence to detect an event). As described in Section 2, various operations are applied on clusters. *New cluster construction*, and *cluster update* operations are performed in the *clustering bolt*, whereas *cluster deletion* is done as the last step in the *event detection bolt*. *Cluster merging* is applied in two stages. Within each round, clusters are generated locally. At the end of the round, similar local clusters are merged. As the second stage of merging, the resulting local clusters are merged with the global clusters complying with the similarity threshold.

Clustering in two stages, local and global, serves for two purposes. As the first one, storage access for updating the global clustering is more costly than doing so for the local clustering, due to the size of the clusters. Therefore, updating the clusters locally for processing the streaming tweets improves time performance. The second reason is due to the lack of transactional support in Storm's stream processing environment. When the streaming tweets are processed in a distributed way, different tweet processing nodes may access the same cluster for updating the cluster's term vector concurrently. This causes the loss of some of the updates and leads to incorrect clusters. While building the clusters locally, this problem is considerably reduced, since the local clusters are much smaller, and the tweets are processed in sequence within a bolt. This is an interesting technical problem that this work revealed, which we would like to study as part of our future work. For example, it would be interesting to explore the use of a transactional stream processing engine such as S-Store [15], as an alternative to Storm and Cassandra to handle both streaming and storage needs in a transactional manner. This could potentially improve both correctness and performance of our techniques.

As in the keyword-based topology, the clustering-based event detection topology given in Figure 2b shows the parallel execution paths in the experiments for processing tweets from USA and Canada. Additionally, the number of bolts created for each task is shown as well. As presented in the figure, clustering task for

the upper path has the highest requirement for distribution, due to high number of tweets and clusters.

3.3 Hybrid Event Detection Topology

The structure of the hybrid event detection topology is presented in Figure 2c. As in the previous topologies, there is a single spout (denoted as *input stream*). However, the topology includes bolts for both detecting bursty keywords (the first two tasks in the topology) and clustering-based event detection (the last two tasks). The basic idea in this method is to filter the tweets such that only those tweets that contain some bursty keyword are clustered towards event detection. By this way, the load on the clustering tasks is reduced. The reduction in the load is obvious in the smaller number of bolts for the clustering task. As another result of the reduced load, just a single event detector node is allocated for both of the tweet processing paths.

In each of the three methods, it is important to create and manage state during execution. We realize this via batch-based stream processing. More specifically, we consider tweet blocks collected within 6-minute rounds as batches. At the end of each round, *state* is generated and saved into a Cassandra store so that next round can use information resulting from the previous round. In the keyword-based method, tf of the words are stored as the state of the round in order to detect events through increment rate of tf-idf values. In the clustering-based method, clusters created/updated in each round constitute the state. Finally, for the hybrid method, both tf values and clusters created/updated in the round are stored as the state at the end of the round. Note that, event detection accuracy depends on correct state maintenance, as well as the event detection method employed.

4 Experimental Evaluation

In this section, we present an experimental evaluation of our event detection techniques in terms of accuracy and performance.

4.1 Setup

All experiments were run on a MacOS Version 10.13.3 machine with an Intel® Core™ i5 processor running at 3.2 GHz with 16 GB of memory.

We used a real Twitter dataset with nearly 12M tweets that we collected within a week, from May 31, 2016 to June 7, 2016. We filtered tweets by geographic location and worked with only the ones posted from USA and Canada. The complete dataset is stored in Cassandra and we replay it in a streaming fashion in our experiments in order to simulate a behavior similar to the real Twitter Firehose. In each of the experiments, we processed the tweets in *rounds* (i.e., time windows) of 6 minutes. We chose this window size so as to create a behavior that is as close to a realistic and stable system scenario as possible

(i.e., tweet collection rate matches the processing rate, and the total latency of buffering and processing each tweet is not too high).

Our main evaluation metrics are the well-known Precision, Recall, and F1-measure for accuracy, and throughput (i.e., total number of input tweets processed per second) and total round processing latency for performance. The clustering-based event detection method (and therefore, the hybrid method which is also based on clustering) involves several parameters used as thresholds (such as cosine similarity threshold to merge clusters or the number of tweets in a local cluster). The optimal values for these parameters are obtained through validation experiments that are conducted on a smaller sample of the whole dataset.

4.2 Event Detection Accuracy and Performance

We now analyze the event detection accuracy of our methods against the *ground truth*. Furthermore, we report our findings on their computational performance.

Ground Truth Construction. We determined the set of events that constitute the ground truth through a user study involving three judges. The following process is applied by each of the judges independently: Given all clusters generated by the cluster-based and the hybrid event detection methods, the most frequent terms in representative term vectors of each cluster is examined in detail, making use of web search with the frequent terms in order to match the cluster with a real-world event that happened within the same time interval as the dataset collection. Some of the events were very clear and well-known events, such as *Death of Muhammad Ali*, which did not need detailed examination. On the other hand, some other events, such as *Offensive Foul by Kevin Love in NBA Finals Game I*, needed a more detailed web search. After the individual evaluation session by each judge, another session is conducted to compare their results. In this second session, the final set of events for the ground truth is determined under full consensus from all three judges. As a result, the ground truth includes 21 different events for the USA tweets and 4 different events for the Canada tweets, including events from the 2016 NBA Finals, the 2016 NHL Final, events about celebrities as well as first appearances of movie trailers and music videos, and the death of Muhammad Ali.

Accuracy Comparison. For accuracy evaluation, we used the well-known relevance metrics of *Precision*, *Recall*, and *F1-measure*. Since the output of the keyword-based method is a set of bursty keywords denoting events, precision is calculated as the ratio of the number of keywords matching some event to the number of keywords found. For the clustering-based and the hybrid methods, precision is calculated as the ratio of number of clusters matching some event to number of clusters found. Recall is calculated in the same way for all three methods, as the ratio of the number of detected events to the total number of events in the ground truth set. Finally, F1-measure is calculated conventionally, as the harmonic mean of precision and recall. Precision, Recall, and F1-measure values

Table 1: Accuracy Results for the Keyword-based Method

Method	Stream	Detected Bursty Keywords	Detected Events	Undetected Events	Precision	Recall	F1
Keyword	USA	220	14 (matching 135 keywords)	7	61%	67%	64%
	CAN	17	2 (matching 7 keywords)	2	41%	50%	45%
	All	237	16 (matching 142 keywords)	9	60%	64%	62%

Table 2: Accuracy Results for the Clustering-based and the Hybrid Methods

Method	Stream	Constructed Clusters	Event Clusters	Undetected Events	Precision	Recall	F1
Clustering	USA	74	39	0	53%	100%	69%
	CAN	7	5	0	71%	100%	83%
	All	81	44	0	54%	100%	70%
Hybrid	USA	87	53	4	61%	80%	69%
	CAN	3	3	1	100%	75%	86%
	All	90	56	5	62%	79%	69%

Table 3: Clustering Analysis for the Clustering-based and the Hybrid Methods

Method	Stream	No. of Clusters	Avg. SC	Min. SC	Max. SC	Std. dev.
Clustering	USA	74	0.855	0.15	1.0	0.276
Clustering	CAN	7	1.0	1.0	1.0	0.0
Hybrid	USA	87	0.62	0.0	1.0	0.41
Hybrid	CAN	3	1.0	1.0	1.0	0.0

for the methods are shown in Table 1 and Table 2. As seen in the results, the clustering-based method provides the highest recall, whereas the hybrid method performs better in terms of precision. This result is reflected in Table 3, as well. The clusters generated by the hybrid method for the USA tweets are all event clusters with high silhouette coefficient [6] values, whereas the clusters generated by the clustering-based method have lower average silhouette coefficient as well as a higher standard deviation. Keyword-based method has an intermediate-level performance, performing slightly better for recall than for precision.

[6] The silhouette coefficient (SC) essentially measures how similar a given object is to its own cluster compared to the other clusters. Its value ranges between -1 and +1, where a higher value indicates higher clustering quality.

Table 4: Performance Results for all Methods

Method	**Number of Tweets Processed per Sec.**	**Number of Rounds Processed per Min.**	**Round Processing Time (Sec.)**
Keyword	1200	9.3	6.5
Clustering	300	2.5	24
Hybrid	797	6.7251	8.96

Performance Comparison. In stream processing, processing time is an important metric to be able to cope with continuous data. Additionally, online event detection calls for timely processing to extract and present the events with the least possible delay. Since the streaming behavior is simulated in our experiments, the tweet arrival rates are set to the same level for each of the methods. For performance, we focus on measuring the throughput in terms of number of tweets processed per second, number of rounds processed per minute and round processing time. Table 4 summarizes our results. As expected, the most efficient method is the keyword-based event detection method with 1200 tweets per second and 6.5 seconds of round processing time on average. In contrast, since the clustering-based method performs many database accesses to maintain cluster state and it has to iterate over larger amounts of data, it incurs the lowest number of tweets processed per second and longest round execution time. The hybrid method shows a major improvement over the clustering-based method, bringing the round execution time down to 7.5 seconds. This proves that filtering tweets based on bursty keywords can be effective in reducing the cost of cluster computation.

4.3 Discussion

The key takeaways from our experimental study can be summarized as follows:

- The clustering-based method provides the highest recall value for USA events and overall. This is an expected result, since this method generates more number of clusters and performs a finer-grained analysis.
- On the other hand, the clustering-based method is also the least efficient method due to higher load of cluster processing and storing state. However, the idea of pre-filtering tweets using keyword counts is a promising way to improve the performance of the clustering-based method, as the performance results of our hybrid method indicate.
- The keyword-based method processes the tweet stream faster than the other two methods, and the bursty keywords provide good hints for detecting the events. However, the same keyword may be associated with several related yet different events. For example, the bursty keyword *game* appears in several clusters' representative vectors. Therefore, it is not easy to associate a keyword with an event precisely.

- Tweet filtering applied in the hybrid method brings considerable efficiency. It also provides a rise in precision per country, and in F1-measure for Canada events. However, there is a drop in recall, due to the filtering applied. Overall, clusters generated by the hybrid method strongly indicate the occurrence of relevant events from the ground truth dataset. The improvement in the time efficiency over clustering-based method brings an advantage for practical use as well.
- In the clustering-based method, we observed cases where multiple event clusters are generated in the same round corresponding to the same event, causing *fragmented clusters*. For example, for the event *Death of Muhammad Ali*, two clusters are generated in the same round, one of them containing frequent terms *champion, rest in peace*, whereas the other one containing *float, butterfly, sting*, referring to the famous quote *"Float like a butterfly, sting like a bee"*. Another advantage of the hybrid method we observed is that, it reduces the degree of this kind of fragmentation.
- In burst detection, processing the rounds separately and keeping the state is essential, yet this incurs a cost. Overall, this study shows that using a stream processing framework for online event detection is a viable idea and can facilitate implementation and scalability, while helping control accuracy. We note that the benefit of this approach could be further improved by providing stronger support for native storage of state and transactional processing to efficiently coordinate concurrent data accesses, which we plan to investigate in more depth as part of our future work.

5 Related Work

Event detection in social networks and microblogging platforms on the web has been a popular research topic for the past decade [5], [17]. Like our work in this paper, a significant portion of previous research has focused on analyzing streams of Twitter posts [3] [8]. Some of these focus on detecting specific types of events such as earthquakes [22], [23] or crime and disasters [10], while others, like in our case, target detecting any type of event gaining interest among bloggers [24], [21], [7]. In both cases, textual messages are first pre-processed to extract important features such as geo-location, followed by a clustering/classification step, where potential events are identified and selected.

Previous online event detection techniques follow various algorithmic approaches. Sayyadi et al. and Ozdikis et al. leverage keyword co-occurrence information to discover similar tweets [26], [19], [20], whereas Zhou and Chen focus on detecting composite social events based on a graphical model [32]. Petrovic et al. propose detecting new events from tweets based on locality-sensitive hashing (LSH) [21]. Systems like EvenTweet and Jasmine exploit location information from geo-referenced messages to detect local events more accurately [2], [29], while Osborne et al. leverage Wikipedia for enhancing story detection on Twitter [18]. TwitterStand is a Twitter-based news processing system that focuses on the problem of noise removal from tweets [24]. TwitterNews+ proposes incremental clustering and inverted indexing methods for lowering the computational

cost of event detection on Twitter [7]. Others also looked into the bursty topic detection problem [30], [31]. Trend detection on Twitter is another related problem [12], [4], where the emphasis is more on identifying longer term events. In [11], Liu et al., describe a system for removing noise in order to detect news events on Twitter. As the employed technique, they focus on first story detection, rather than burst detection.

Some of the studies use machine learning techniques for detecting events, thus they need a training set or several keywords. Medvet and Bartoli's study requires set of potentially related keywords to detect trending events with their sentiment polarity [14]. In [9], Illina et al. use textual messages and n-grams to classify the social media postings as event related and non-event related.

Our work differs from the above related work in that we take a stateful stream processing approach to accurate and efficient event detection. In this approach, we leverage a state-of-the-art system infrastructure based on a distributed stream processing system (Apache Storm) for low-latency event detection combined with a scalable key-value storage system (Apache Cassandra) for maintaining state. To our knowledge, there are two approaches that are the most closely related to ours: McCreadie et al.'s work on distributed event detection [13] and the RBEDS real-time bursty event detection system proposed by Wang et al. [28]. Like in our approach, both of these also implement online event detection solutions on top of Storm. However, they tackle different aspects of the problem. McCreadie et al. focus on scaling event detection to multiple nodes using a new distributed lexical key partitioning scheme as an extension to the LSH-based algorithm previously proposed by Petrovic et al. [21], while Wang et al. focus on applying the k-means clustering algorithm to the burst detection problem on Storm. Neither of these approaches pays attention to statefulness aspect of the problem and the need for balancing event detection accuracy and performance like we do. Thus, the three approaches are complementary.

6 Conclusion and Future Work

In this work, we model event detection problem as burst detection in frequency of keywords or in size of message clusters. We analyze the performance of three methods for event detection implemented on the Apache Storm distributed stream processing framework. These methods are evaluated on a real tweet dataset collected over a week and replayed as a stream. The experimental results show the applicability of our stream-based approach for online event detection. Among the compared methods, hybrid method provides a better balance between accuracy and processing time cost. It has lower recall value than clustering based method, but can detect event with higher precision.

This work uncovers several interesting problems that can be studied as future work. Semantic similarity based measurements can be utilized to prevent fragmentation of clusters related to the same event. On the other hand, fragmentation can be useful to detect events that may have different durations, which is an interesting direction for extending our work. In our stream simulations, we

did not apply any normalization on the load of the rounds, but the topology is determined empirically to handle the average load. Automated adaptation of a topology for load balancing can be further studied. As another research direction, utilization of a transactional stream processing engine (instead of Storm and Cassandra) can be investigated and its effect on event detection accuracy and performance can be analyzed.

References

1. IEEE Data Engineering Bulletin, Special Issue on Next-Generation Stream Processing (2015)
2. Abdelhaq, H., et al.: EvenTweet: Online Localized Event Detection from Twitter. PVLDB **6**(12) (2013)
3. Atefeh, F., Khreich, W.: A Survey of Techniques for Event Detection in Twitter. Computational Intelligence **31**(1), 132–164 (2015)
4. Becker, H., et al.: Beyond Trending Topics: Real-World Event Identification on Twitter. In: International AAAI Conference on Weblogs and Social Media (ICWSM). pp. 438–441 (2011)
5. Cordeiro, M., Gama, J.: Online Social Networks Event Detection: A Survey. In: Michaelis, S., Piatkowski, N., Stolpe, M. (eds.) Solving Large Scale Learning Tasks. Challenges and Algorithms, Lecture Notes in Computer Science, vol. 9580, pp. 1–41. Springer, Cham (2016)
6. González-Jiménez, M., de Lara, J.: Datalyzer: Streaming Data Applications Made Easy. In: International Conference on Web Engineering (ICWE). pp. 420–429 (2018)
7. Hasan, M., et al.: TwitterNews+: A Framework for Real Time Event Detection from the Twitter Data Stream. In: Spiro, E., Ahn, Y.Y. (eds.) Social Informatics, Lecture Notes in Computer Science, vol. 10046, pp. 224–239. Springer, Cham (2016)
8. Hromic, H., et al.: Graph-based Methods for Clustering Topics of Interest in Twitter. In: International Conference on Web Engineering (ICWE). pp. 701–704 (2015)
9. Ilina, E., et al.: Social Event Detection on Twitter. In: International Conference on Web Engineering (ICWE) (2012)
10. Li, R., et al.: TEDAS: A Twitter-based Event Detection and Analysis System. In: IEEE International Conference on Data Engineering (ICDE). pp. 1273–1276 (2012)
11. Liu, X., et al.: Reuters Tracer: A Large Scale System of Detecting & Verifying Real-Time News Events from Twitter. In: ACM International on Conference on Information and Knowledge Management (CIKM). pp. 207–216 (2016)
12. Mathioudakis, M., Koudas, N.: TwitterMonitor: Trend Detection over the Twitter Stream. In: ACM SIGMOD International Conference on Management of Data (SIGMOD). pp. 1155–1158 (2010)
13. McCreadie, R., et al.: Scalable Distributed Event Detection for Twitter. In: IEEE International Conference on Big Data. pp. 543–549 (2013)
14. Medvet, E., Bartoli, A.: Brand-Related Events Detection, Classification and Summarization on Twitter. In: IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT). pp. 297–302 (2012)
15. Meehan, J., et al.: S-Store: Streaming Meets Transaction Processing. The Proceedings of the VLDB Endowment (PVLDB) **8**(13), 2134–2145 (2015)

16. Milstein, S., et al.: Twitter and the Micro-Messaging Revolution: Communication, Connections, and Immediacy – 140 Characters at a Time (An O'Reilly Radar Report). http://weigend.com/files/teaching/haas/2009/readings/OReillyTwitterReport200811.pdf (2008)
17. Mokbel, M.F., Magdy, A.: Microblogs Data Management Systems: Querying, Analysis, and Visualization (Tutorial). In: ACM SIGMOD International Conference on Management of Data (SIGMOD). pp. 2219–2222 (2016)
18. Osborne, M., et al.: Bieber no more: First Story Detection using Twitter and Wikipedia. In: SIGIR Workshop on Time-aware Information Access (TAIA) (2012)
19. Ozdikis, O., et al.: Semantic Expansion of Tweet Contents for Enhanced Event Detection in Twitter. In: International Conference on Advances in Social Networks Analysis and Mining (ASONAM). pp. 20–24 (2012)
20. Ozdikis, O., et al.: Incremental Clustering with Vector Expansion for Online Event Detection in Microblogs. Social Network Analysis and Mining **7**(1), 56 (2017)
21. Petrovic, S., et al.: Streaming First Story Detection with Application to Twitter. In: Human Language Technologies: Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL). pp. 181–189 (2010)
22. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In: International Conference on World Wide Web (WWW). pp. 851–860 (2010)
23. Sakaki, T., et al.: Tweet Analysis for Real-Time Event Detection and Earthquake Reporting System Development. IEEE Transactions on Knowledge and Data Engineering (TKDE) **25**(4), 919–931 (2013)
24. Sankaranarayanan, J., et al.: TwitterStand: News in Tweets. In: ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS). pp. 42–51 (2009)
25. Sarma, A.D., et al.: Dynamic Relationship and Event Discovery. In: ACM International Conference on Web Search and Data Mining (WSDM). pp. 207–216 (2011)
26. Sayyadi, H., et al.: Event Detection and Tracking in Social Streams. In: International Conference on Web and Social Media (ICWSM). pp. 311–314 (2009)
27. Sellam, T., Alonso, O.: Raimond: Quantitative Data Extraction from Twitter to Describe Events. In: International Conference on Web Engineering (ICWE). pp. 251–268 (2015)
28. Wang, Y., et al.: A Storm-Based Real-Time Micro-Blogging Burst Event Detection System. In: Wang, X., Pedrycz, W., Chan, P., He, Q. (eds.) Machine Learning and Cybernetics, Communications in Computer and Information Science, vol. 481, pp. 186–195. Springer (2014)
29. Watanabe, K., et al.: Jasmine: A Real-time Local-event Detection System based on Geolocation Information Propagated to Microblogs. In: ACM International Conference on Information and Knowledge Management (CIKM). pp. 2541–2544 (2011)
30. Xie, W., et al.: TopicSketch: Real-Time Bursty Topic Detection from Twitter. IEEE Transactions on Knowledge and Data Engineering (TKDE) **28**(8), 2216–2229 (2016)
31. Zhang, T., et al.: A Refined Method for Detecting Interpretable and Real-Time Bursty Topic in Microblog Stream. In: Web Information Systems Engineering (WISE). pp. 3–17 (2017)
32. Zhou, X., Chen, L.: Event Detection over Twitter Social Media Streams. The VLDB Journal **23**(3), 381–400 (2014)