ONLINE EVENT DETECTION FROM STREAMING DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZLEM CEREN ŞAHİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY 2018

Approval of the thesis:

ONLINE EVENT DETECTION FROM STREAMING DATA

submitted by **ÖZLEM CEREN ŞAHİN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün _____
Head of Department, **Computer Engineering**

Prof. Dr. Pınar Karagöz _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Assoc. Prof. Dr. İsmail Sengör Altıngövde
Computer Engineering, METU _____

Prof. Dr. Pınar Karagöz
Computer Engineering, METU _____

Assist. Prof. Dr. Orkunt Sabuncu
Computer Engineering, TEDU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ÖZLEM CEREN ŞAHİN

Signature :

ABSTRACT

ONLINE EVENT DETECTION FROM STREAMING DATA

ŞAHİN, ÖZLEM CEREN
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. Pınar Karagöz

May 2018, 103 pages

The purpose of this study is detecting events from social media in an online fashion where event is a happening that takes place at a certain time and place that attracts attention within a short period of time. By doing so, it is aimed to provide a system both accurate and efficient at the same time. The problem studied in this thesis is modeled as a stream processing problem and three alternative methods are proposed. The first event detection method is keyword-based and works with bursty keywords inside social media messages. The second method is clustering-based method and suggests an improved version of hierarchical clustering algorithm. The last one is hybrid method which merges the previous two methods. All the methods introduced are implemented on top of Apache Storm and Cassandra to provide a distributed and scalable system, and each method has the ability to distinguish data belonging to different countries and events are tagged with country information. Each method is evaluated experimentally in terms of both accuracy and performance based on a real dataset with 12M tweet messages collected from Twitter.

Keywords: Online event detection, Microblogging, Real-time Evaluation, Stateful stream processing, Distributed system, Keyword-based event detection, Clustering-based event detection, Hierarchical clustering, Twitter, Apache Storm, Apache Cassandra

ÖZ

AKAN VERİ ÜZERİNDEN ÇEVRİMİÇİ OLAY BELİRLEME

ŞAHİN, ÖZLEM CEREN
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Pınar Karagöz

Mayıs 2018 , 103 sayfa

Bu çalışmanın amacı meydana gelen olayları kısa bir sürede belirleyebilmektir. Bunu yaparken, aynı zamanda hem doğru hem de verimli bir sistem sağlamayı amaçlamaktadır. Bu tezde çalışılmış olan problem, bir akış işleme problemi olarak modellenmektedir ve üç alternatif yöntem önerilmektedir. İlk olay algılama yöntemi, anahtar kelimeye dayalıdır ve sosyal medya iletilerinde patlama yapmış anahtar kelimelerle çalışır. İkinci yöntem kümeleme tabanlı bir yöntemdir ve hiyerarşik kümeleme algoritmalarının geliştirilmiş bir versiyonunu önerir. Sonuncusu, önceki iki yöntemi birleştiren bir melez yöntemdir. Tüm yöntemler, dağıtılmış ve ölçeklendirilebilir bir sistem sağlamak için Apache Storm ve Cassandra'nın üzerine uygulanmaktadır ve her bir yöntem farklı ülkelere ait verileri ayırt etme yeteneğine sahiptir, olaylar ülke bilgileriyle etiketlenmektedir. Her bir yöntem hem doğruluk hem de performans açısından deneysel olarak değerlendirilmektedir.

Anahtar Kelimeler: Çevrimiçi olay belirleme, Mikroblog, Gerçek zamanlı değerlendirme, Durumsal akış işleme, Dağıtık sistemler, Anahtar kelimeye dayalı olay be-

lirleme, Kümelemeye dayalı olay belirleme, Hiyerarşik kümeleme, Twitter, Apache Storm, Apache Cassandra

dedicated to my loved ones...

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Prof. Dr. Pınar Karagöz. The door to Prof. Karagöz's office was always open whenever I was in a trouble or had a question about my study. I always feel her support, encouragement and motivation besides me.

I like to thank Nesime Tatbul for the useful comments and remarks. I am gratefully indebted to her for very valuable comments on this thesis. I also like to thank Alper and other participants in the ground truth survey, who have willingly shared their precious time during the process of survey.

I would like to show my gratitude to defense jury members, Assoc. Prof. Dr. İsmail Sengör Altıngövde and Assist. Prof. Dr. Orkunt Sabuncu for evaluating my thesis and their valuable feedbacks.

I would like to thank to all my friends and my colleagues for their support throughout my thesis. I owe special thanks to Güneş, who is involved in the entire process, both by keeping me harmonious and helping me to put pieces together. I will be grateful forever for his patience and support.

Finally, I must express my very profound gratitude to my parents, Deniz and Necdet, to my little sister, Ece, and to my grandparents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

TDT	Topic Detection and Tracking
CEP	Complex Event Processing
API	Application Programming Interface
SAX	Symbolic Aggregate ApproXimation
LSH	Locality-Sensitive Hashing
CDE	Crime and Disaster related Events
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
TEDAS	A Twitter-based Event Detection and Analysis System
RBEDS	Real-time Bursty Event Detection System
NLP	Natural Language Processing
URL	Uniform Resource Locator
TF-IDF	Term Frequency-Inverse Document Frequency
DB	Database
CAN	Canada
USA	United States of America

CHAPTER 1

INTRODUCTION

1.1 Overview

The rise of social media and microblogging services started out as a way to share information and news quickly on the web, and has become the largest source of publicly accessible data especially with widespread use of smart phones, tablets, etc. Today, people can use social media anytime, anywhere and social media has become the main platform where people express their thoughts and react to current problems. People broadcast the events occurring around them or the situation they are in when something happens. This allows social media to take an important role in event detection and analysis.

Twitter is the most popular micro-blogging social network service having more than 300 million monthly active users posting about 340 million tweets in a day where people can post 140-character long messages[2]. Twitter provides data with geo-location tags which are obtained from devices having location services such as smart phones or tablets and that makes Twitter data extremely valuable for many disciplines from market studies to real-time trend detection.

Social media has also been used in many academic studies due to the variety and abundance of data [3, 4, 5, 6]. Up to today social media data including microblogs posted on Twitter have been used to detect earthquakes, disasters, political topics, traffic, etc. Nevertheless, detecting events by using social media is still an active and popular research problem. In computer science literature, one can find different definitions for the concept of an *event*. In this work, we follow the event definition that

comes from Topic Detection and Tracking (TDT) research, which applies data mining techniques on textual documents [7]. In such studies, *event* is defined as an activity that happens at a specific time and place and that attracts attention in short time[3]. Following this, *event detection* is detecting events using textual content [5]. We adapt these definitions within the following context:

- We work on social media posts, more specifically tweets, as the textual media.
- We assume that the time of an event constitutes a certain time interval.

In terms of event detection process, the literature contains *off-line*, or *retrospective* techniques, and *online* techniques. We work on online event detection.

We assume that the *sudden increase* in the amount of mentioning about the same topic is a strong indicator of an event. This is denoted as a *burst* in the mention of the event. The burst in the mention can be measured in several ways. In this work, we focus on two alternative approaches: measuring *burst in words* and measuring *burst in collection of tweets*.

In order to clarify the scope of the work, it is important to note the difference between *event detection* and *complex event processing (CEP)*. In event detection, although the definition of the event is clear, its structure is not clear; whereas in CEP, the structure of the event is complex, but is defined through patterns. Hence, the focus in CEP is the extraction of the instances complying with the predefined pattern, under low latency (e.g., [8, 9, 10]). In this work, the focus is on event detection, rather than CEP.

Under these given assumptions, we can define the *online event detection* problem as follows: Given a stream of microblog posts, *detect all events (i.e., "bursts") in the stream* both accurately and efficiently, where a burst manifests itself either as an increase in word occurrences or as an increase in the message cluster sizes which can be worthy in many ways from being aware of breaking news to being informed about a natural disaster.

While plenty of researches have been made since the emergence of Twitter [3, 11,

12, 13, 14], only a handful of them make improvements on both accuracy and performance [15, 16]. Along with the obvious importance of accuracy, performance takes an important place with the considerable increase in data volumes, e.g. Twitter provides over 1000 tweets per second. Therefore, the ideal solution and the purpose of this study is maintaining high accuracy with low latency under high input rates.

This study considers event detection as a stream processing problem. Low-latency, high-throughput stream processing platforms are in use for years and many mature platforms are present [17]. These platforms are laying the groundwork for high performance event detection studies. Stream processing platforms are studied with event detection applications in a few researches[15, 16], but there is a lack of researches studying high-accuracy event detection methods on top of stream-based platform. Therefore, in this study, stream-based design and implementation of three different event detection algorithms are explored: keyword-based method, clustering-based method and a hybrid method. While keyword-based method examines the words by itself in a collection of splitted tweets, clustering-based method groups tweets by the similarity of their contents. Despite the opposing characteristics of key-based and clustering-based methods, hybrid methodology combines them by using the former technique to filter dataset followed by grouping the remaining dataset.

1.2 Contributions

The main purpose of this study, as mentioned above, is capturing events accurately by using different types of detection methods on top of stream processing system to accelerate the processing of huge data by distributing. For this purpose three different methods are hypothesized: keyword-based method, clustering-based method and hybrid method. Keyword-based method depends on splitted words of tweets and their frequency changes in time, but this method does not spot the relation of words or tweets and results in set of words as events which should be associated with others manually as a latter step. On the other hand clustering method operates with whole tweets and groups tweets by their similarity and relation. Therefore keyword-based method stays as a simpler and faster method with less sensible results, but clustering

based method produces more accurate results in a longer time. While keyword-based method takes advantage of being a syntactical approach by means of performance, clustering-based method produces more accurate results by paying attention to meanings and relations between data. As a combination of these two methods, hybrid method uses the initial approach to filter data including bursty words and afterwards group these filtered tweets by their similarity as in second method. Therefore, hybrid method of event detection stays in the middle of those two approaches by producing more expressive events with above average performance.

Along with the solution for accuracy of this study, performance requirements are met by using a stream processing platform, Apache Storm. This study considers the problem as a *stateful stream processing* problem, i.e. all the algorithms defined above uses fixed-size time windows, namely *rounds* to process data as a chunk at every round and some state is transferred between each consecutive rounds. The need for stateful stream processing becomes the main issue when merged with a distributed system since correct state transaction is crucial for accuracy of methods and system natively has no support for batch processing or state transactions. For the state transactions and data storage, Apache Cassandra, a key-value store is deployed.

The need for stateful stream processing has directed us to improve existing techniques or experience different ways of execution steps for keyword-based and clustering-based methods. Additionally, hybrid methodology has come into scene to cross in the middle for accuracy and performance. Since event detection mostly depends on the growth on words or clusters, it is important to find a way to break stream at some point and evaluate the growth and compare with the previous ones. For this purpose we have used the time window and control the growth rates at the end of time windows. Hence, it is important to start/stop streaming and executions appropriately. Keyword-based method has two different approaches for that purpose, first one is waiting input streams for a predefined time and the other one is taking control of distribution mechanism from Apache Storm so that stream can be easily controlled by us. Thus, since keyword-based method for event detection is widely used, these improvements and different handling of state transactions and their comparisons are novel. Similarly, clustering based approaches are studied before but in this disserta-

tion, hierarchical clustering is splitted into two steps and it is applied on a distributed system. Hybrid method of these two approach is also a novelty for this study. Additionally, all of these three methods are adapted to multi country examination so that events from two countries are evaluated separately on the same infrastructure and the system is capable of increase this number easily by adding more parallelism.

In experiments, real Twitter dataset is replicated in real time order and processed by taking their geo-location information into account. As the last contribution, system is designed so that the place affected by an event can be detected.

All three applications have implemented on top of Apache Storm & Cassandra and experimentally tuned by using various scenarios. From the experiments, it is found that there is an explicit accuracy-performance trade-off between keyword-based and clustering-based methods. On the other hand, hybrid method of event detection provides a settlement between these two techniques.
The contributions of this thesis work can be summarized as follows:

- Three event detection methods are realized on stream processing framework and their performances are compared in terms of accuracy and time efficiency.
- The first and second methods are previously used in the literature. Within the scope of this work, these two methods are modified and improved to adapt the distributed environment and also a new hybrid method is developed. This hybrid method combines the advantages of the other two methods.
- All these three methods are built on the stateful stream processing platform, and they use fixed-size time windows to process data as chunks and track the change between rounds.
- The system is designed as a real-time computation system. Thus, a real Twitter data set is replicated for experiments and the execution time of the system is far below the interval of data set; while 7 days of Twitter data is used, maximum execution time is less than 12 hours.
- The designed system is capable of processing multiple countries separately on

the same platform by using parallelism. For the experiments two different countries are evaluated.

- All three approaches have been implemented on top of Apache Storm & Cassandra to take advantage of the distributed systems and increase the performance of system.

1.3 Organization of Thesis

This dissertation is divided into six chapters starting with this introductory chapter. This chapter is followed by the related work chapter which describes the previous researches related to event detection techniques and about the technologies used in our study. At the end of related work section, the most similar researches and difference of this study is discussed. Chapter 3 contains the detailed description of three different methods present in this thesis, conceptually. This chapter ends with an illustrative example to simulate the methods. Afterwards in Chapter 4 the equivalent implementation details of theoretical method definitions are given in detail. At the beginning, the Twitter API, used to collect Twitter data; and Apache Storm, preferred as a distributed real-time computation system, are introduced briefly. At the end of this chapter, the event detection by geolocation implemented in this study is described and finally the brief introduction and the usage of Apache Cassandra in our system is given. Chapter 5 contains the experiment environment and results. Firstly setup is described in this section, then preprocess input and outputs are given, afterwards it is explained how the parameters are determined and finally statistical information about performance and accuracy of system is given and results are discussed as conclusion. The last chapter is composed to give final discussions about methods, system and future work. There are 4 appendices added to this thesis where the first one contains tabular data showing events detected by clustering and hybrid method, second one contains the detailed description of Apache Cassandra table descriptions, third one contains three sample figures of events detected by keyword-based method visualizing the tf-idf increase by time and the last one contains the execution times of each task for each method.

CHAPTER 2

RELATED WORK

Twitter is used by millions of users around the world to exchange information on web and many studies use Twitter data for information retrieval after it came into scene in 2006 [18, 19, 5, 6]. The focus of this study is *event detection*, also known as *event tracking* where *event* can be described as a happening that takes place at a certain time and place, causing a short window of sudden burst in attention from the microbloggers and event detection aims to detect events by processing textual materials like social media[20]. Atefeh and Khreich [5] studied on Twitter streaming, syntactic filtering and correction of tweets and classified different event detection techniques by using the types of event and event detection methods. Cordiero and Gama [3] studied on an improvement of event detection methods from Twitter data using network analysis by means of user relations and interactions.

As a common point to all researches, initially pre-process step takes place and it is followed by clustering or classification procedures to identify potential events. Hasan et al. [21] make a survey about Twitter streams having different features and using different techniques. In the survey both term-based and cluster-based methods take place. For example Li et al. [22] detect events by top-k bursty word segments in a specific time windows, Marcus et al. [23] make detection by keywords provided by users to find peaks, Mathioudakis and Koudas [24] detect terms with high frequency as events, Alvanaki et al. [25] use the correlation of tag pairs, Gaglio et al. [26] work on a specialized term scoring to identify top-k terms within a dynamic temporal window, Cataldi et al. [27] detect events by user authorities and previous usage informations, Stilo and Velardi [28] use Symbolic Aggregate ApproXimation (SAX) to make terms discrete and use Wikipedia Events4 to remove non-event terms, Parikh

and Karlapalem [29] take leverage of increase in frequency and appearance of term patterns, Weng and Lee [30] use discrete wavelet signals to filter out trivial words, Zhang et al. [31] identify events by augmented normalized term frequency and user authority, Zie et al. [32] also suggest a system offering real-time bursty topic detection.

All of the above studies are term-based approaches. Besides the following works stand on clustering based approaches: Hasan et al. [14] use defragmentation submodule to handle cluster fragmentation with incremental clustering to lower the computational cost, Petrovic et al. [33] use expiration mechanism and locality-sensitive hashing(LSH), Osborne et al. [34] classify security related events and in another study Osborne et al. [35] take leverage of Wikipedia for story detection, Becker et al. [36] execute a periodic second pass to handle cluster fragmentation and mostly focus on trend detection which interests in longer-term events and similarly Mathioudakis and Koudas [37] improve a system called TwitterMonitor which also focuses on trend detection, De Boom et al. [38] improve the approach introduced by Becker et al. by adding hashtag-level semantics, Phuvipadawat and Murata [39] apply similarity-threshold-based incremental clustering, McMinn and Jose [40] utilize inverted indices for named entities and aggressive tweet filtering for scalability, Unankard et al. [41] use leader-follower clustering algorithm and take advantage of content and concept similarity, Kaleel and Abhari [42] prefer LSH-based incremental clustering improved by prefix-tree structure, Lee and Chien [43] work on a modified Incremental DBSCAN clustering algorithm. Sankaranarayanan et al. [44] receive tweets of specified users from different parts of the world; cluster them for event detection, and assign a geographic location to the event in order to display it on a map.

While all the above previous studies focus on any type of events occurred, there are also some studies which are specialized on a decided topic. Sakaki et al. [45, 11] detect earthquakes and their locations by following tweets. Li et al. [46], on the other hand focus on Crime and Disaster related Events (CDE), such as shooting, car accidents, or tornado, as they are important types of events. In another work, Park et al. [47] detect events related to a baseball game and list people who watch the game on TV.

There are also some improvements about event detection on Twitter, for example returning the first tweet posted about an event [48]. In several studies, semantics in word co-occurrences has been used to find the similarities. In the study "Dynamic Relationship and Event Discovery" [49], burst detection and co-occurrence methods are used for event detection. They prepare lists to identify words which belong to same event. To achieve this they use the burst approach, i.e. if two words have burst in same window, they point the same event. A similar approach is used in "Event Detection and Tracking in Social Streams" [50]. They generate a graph where terms are represented as nodes and co-occurrences are represented as edges, then connected sub-graphs are labeled as event clusters.

Methods identifying hashtag or word associations are also presented in some studies. They aim to detect hashtags with an increasing co-occurrence value within a period of time. Plachouras and Stavrakas make users select data and try to explore the context of the selected data by defining associations between words and entities using predefined queries. In another work, Huang et al. [51] aims to predict the name of the product sold on online-store by using previous queries so that sellers can give a more appropriate titles to their products instead of shortened and non-descriptive ones.

Ozdikis et al. [52] proposed a method to enhance the event detection techniques by using lexico-semantic expansion of tweets. To achieve this they use document similarity techniques and clustering algorithms. Similar to the approach by Ozdikis et al., Sayyadi et al. [50] also leverage word co-occurrences to detect events. Zhou and Chen [53] suggest a graphical model to detect social events.

Other than detection methods, studies differ as being online or offline. The studies of Feng et al. [54] focus on hierarchical spatio-temporal hashtag clustering techniques by using an offline technique, ie. using historical data. Similarly, Singh et al. [55] collect spatio-temporal-thematic data from Twitter and Jaiku to detect events. On the other hand there are many researches using real time event detection approaches. Abdelhaq et al. [56] create a system called EvenTweet which detects localized events in real time and also follows the change of event over time. Hasan et al. [14] offer by TwitterNews+ a scalable event detection system detecting events in real time, Sakaki et al. [45, 11] similarly use real-time event detection techniques for earthquake de-

tection and Sankaranarayanan et al. [57] suggest a real-time localized event detection called TwitterStand and focus on the noise removal problem from tweets. Watanabe et al. [12] also suggest a system called Jasmine which is a real-time event detection system based on geolocation information. There are also some studies applying both online and offline techniques like TEDAS [46].

The emergence of Twitter opens a research area on event detection from tweets in real time and detecting events from Twitter is much faster than any news channel since tweets are sent at the time event is happening. However character limitation, volume of data, location sensors, connections between data sources, performance of system and accuracy of system are all study cases for the offered systems and all of them focused on different type of problems, or they improve each other in a sense. This study differs from other related work in that we suggest a distributed, scalable stateful stream processing system which detects localized events and focuses on accuracy as well. 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, McCreadie et al.'s and Wang et al.'s works are the closest approaches to ours. McCreadie et al.'s work [15] also suggests a distributed event detection environment and Wang et al. [16] propose RBEDS which is a real-time bursty event detection system. Similar to our approach, these two approaches also built on Apache Storm, but they focus on different aspects of the problem. McCreadie et al. [15] extend Petrovic et al.'s LSH-based approach by scaling event detection to multiple nodes using a new distributed lexical key partitioning, while Wang et al.'s work applies k-means clustering to detect bursts on Storm. As distinct from these two approaches we propose a stateful system with three different methods, and we balance the accuracy and performance issues. The methods compared in this study are burst detection on keywords by using tf-idf values, burst detection on clusters created by hierarchical clustering and hybrid method of these two. Thus, McCreadie et al.'s, Wang et al.'s and our studies are complementary.

CHAPTER 3

EVENT DETECTION METHODS

This chapter contains the detailed description about three different event detection techniques, namely keyword-based event detection, clustering-based event detection and hybrid methodology with two of the previous methods. As an enhancement, all of these three methods start with a preprocess step using Stanford NLP parser to increase accuracy and process data with regard to geolocation. For the performance and handling big data, distributed data computation tools as Apache Storm and Apache Cassandra, explained in following sections, are used.

Another important point is that streaming posts are processed in windows of predefined time intervals, which are called *rounds*. In each round, the stream is processed and a state is generated. Hence, at each round, the previous state is preserved, and there is a state transition between consecutive rounds. We can consider event detection as a state that is reached according to the change between two consecutive states.

Preprocessing. In the experiments and method evaluations, data are collected from Twitter where Twitter provides the collection of limited size of text which includes the opinions, sentiments, etc. of different people with different writing skills and styles. This situation arises a need to normalize and standardize the data which contain shortened, extended, derived words or unrelated texts like website links. Therefore, as a first step data are preprocessed to increase the accuracy and performance by using "Stanford Natural Language Processing Library". Preprocess of a tweet includes the following points,

- **Tokenize** the sentence into words using spaces and punctuation for the use of

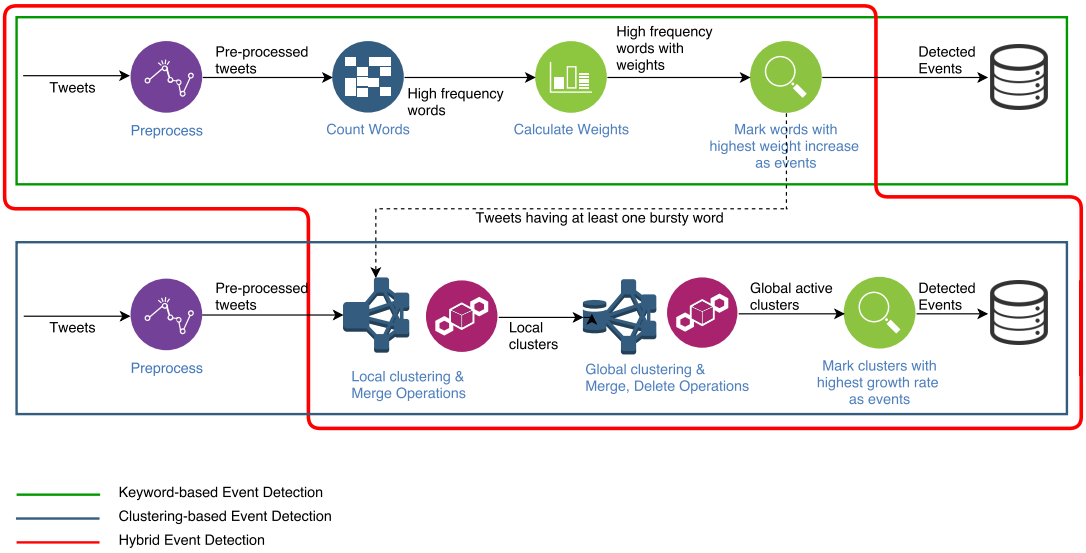


Figure 3.1: Event Detection Methods

further preprocessing.

- Apply **stemming and stop word elimination** using NLP library [58]. The reason for stemmer and stop word elimination is increasing the accuracy and performance. While stemming reduces the word diversity, stop word elimination eliminates the unnecessary and irrelevant words.

- Remove **geo-references** which are sentences started with "I am at" and URLs.

- **Normalize** words by removing characters which repeats more than 2 times consecutively. For example replace "noooooooooooo" with "no".

Stemming and normalization takes the most important roles aggregating the different occurrences of the same word. "firing", "fireeeee" and "fire" should be counted as same word which will result in better performance and more accurate detection.

3.1 Keyword-based Event Detection Method

Keyword-based event detection method intends to reveal the uncommonly common words that show high occurrence only in certain rounds by evaluating the usage of each word in each round. The words which are frequently used in many documents

or the words which are underused are not the interest of this algorithm. The main interest is detecting the unexpected increase in the occurrence or observation of the words with respect to the 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 3.1 for an overview). Hence, at the end of each round a set of event keywords are obtained.

3.1.1 Word Counts

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 through stream, the stemmed and normalized words are counted. In order to limit time and space complexity in the following steps, there is a need for initial elimination of words whose frequency is below a given threshold. By this pre-elimination we can define the most common words in current time-block and avoid unnecessary calculations for the uncommon words.

3.1.2 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 [59]. Since all the tweets in a round are considered as a single document, frequency of a word denotes its frequency in the round.

Tf-idf is the short form of term frequency-inverse document frequency which shows a numerical statistic pointing out the importance of a word to a document in a collection of documents. This technique is mostly used in information retrieval and text mining as a weighting factor. The tf-idf value of a word increases proportionally to tf values which calculates the number of times a word appears in the document, however it has offset by the frequency of the word in the corpus, which helps to eliminate words

which appear more frequently in general [60].

The value term frequency in a document can be calculated as follows:

$$\text{tf}(t, d) = \frac{f_{t,d}}{|\{t' \in d\}|} \tag{3.1}$$

where,

- $f_{t,d}$: the number of times that term t occurs in document d.
- $|\{t' \in d\}|$: total number of terms in document d.

Then, the value idf among all documents [1] can be calculated as:

$$\text{idf}(t, D) = log\frac{N}{1 + |\{d \in D : t \in d\}|} \tag{3.2}$$

where,

- N: total number of documents in the corpus N = $\{|D|\}$
- $|\{d \in D : t \in d\}|$: the number of documents where the term t appears. *i.e.*, $\text{tf}(t, d) \neq 0$. If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

And finally the overall tf-idf value can be computed as follows:

$$\text{tf-idf}(t,d,D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{3.3}$$

where,

- $\text{tf}(t, d)$: tf value calculated by using equation 1
- $\text{idf}(t, D)$: idf value calculated by using equation 2

[1] For efficiency and feasibility reasons, we consider previous 2 rounds for *idf* calculation.

3.1.3 Detecting Events Using TF-IDF

As a final step of this method, the increase in observation of a word is checked by comparing the weight of the words in terms of tf-idf values in consecutive rounds. The increase is compared against a threshold in order to consider a keyword as an *event related word*.

$$\text{increment\_rate} = \frac{tf\text{-}idf(t, d_{current}, D)}{tf\text{-}idf(t, d_{previous}, D)} \tag{3.4}$$

where,

- $tf\text{-}idf(t, d_{current}, D)$: the tf-idf value of the word for current document.
- $tf\text{-}idf(t, d_{previous}, D)$: the tf-idf value of the word for previous document.

Using this formula the percentage showing the increment rate can be calculated and this rate is pointing out even if the word represents an event or not. To sign a word as an event, the increment_rate of the word should be above the specified threshold. If the tf-idf value of a common word for the current document is much higher than the tf-idf value of the word for the previous document, then obviously the word is an uncommonly-common word and it is marked as an event. Otherwise, i.e if the tf-idf value of a word is high or low for each document or if it has high tf-idf value for the previous documents, it isn't marked as event since it is not uncommonly common word for the current round.

3.2 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. This methodology creates clusters from tweets hierarchically, starts from zero cluster and ends with unknown number of clusters. When the event detection procedure starts, there does not occur any clusters and the number of clusters at the end

of algorithm is unpredictable. During the stream of tweet vectors, new clusters are created and existing clusters are updated. At the beginning of clustering methodology, the number of clusters is undefined (unlike k-means clustering) and clusters are shaped during tweet streaming for each tweet which causes some problems discussed below.

The method is composed of two basic steps: *cluster formation* and *event detection* (see Figure 3.1 for an overview). In each round, these steps are applied in sequence.

3.2.1 Cluster formation

This method starts with an empty set of clusters and clusters are formed by using four basic cluster operations:

- creating a new cluster
- updating an existing cluster
- merging two clusters
- deleting a cluster

Creation of new clusters, update of an existing cluster and merge of existing clusters operations are used to follow the growth rate of clusters where the growth rate will be used to decide even if a cluster is an event or not. On the other hand, deletion operation is needed for efficiency and storage issues since the number of clusters directly affect the performance because of that each fetch of cluster information refers to database transaction. Therefore, it is required to remove clusters which are not active lately.

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, clusters from previous rounds don't take into account). 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.

One-stage clustering which stores clusters into one place and access there for each tweet is not applicable in distributed and high-performance systems. The reason of this is that to make parallel tasks synchronize, it is needed to use a shared memory which is Cassandra database in our case and provide database access for each tweet (to get the latest version of clusters to each task to avoid data loss) and this causes an exponential increase of execution time as the number of clusters increase. Therefore, another algorithm is evolved from the clustering method defined above and clustering step is divided into two steps:

- Sub-clustering (local clustering) during a round, starts with an empty set of clusters and local set of clusters are formed using tweets only emitted to owner task. At the end of the round, each task has their own cluster set and emits these sets to event detector.
- As the final step of the current round, local clusters are merged within themselves first and finally global clusters are retrieved from database and updated by using local clusters of each parallel task.

This division of clustering logic provides high performance improvements.

3.2.2 Cosine Similarity of Tweets

Cosine similarity is used for two non-zero vectors to measure the angle between them. Cosine similarity of 1 means that two of the vectors are same and 0 means they are totally different. Cosine similarity is popular in a wide area of researches since it is very efficient for sparse vectors.

The cosine similarity of two vectors can be calculated as follows:

$$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.5}$$

where,

- A_i: First tweet as vector.
- B_i: Second tweet as vector.
- $cos(\theta)$: Cosine similarity of two vectors.

Clustering-based event detection method computes the cosine similarity for two operations. First one is to assign a tweet to a cluster. For that purpose cosine similarity is calculated between the current tweet vector with each cluster vector until it finds the similar cluster or it may create a new cluster. Second one is to merge clusters. For this merge operation cosine similarity is calculated between two cluster vectors. The cosine similarity is used with a threshold value while merge and update operations.

3.2.3 Detecting Events Using Clusters

At the end of the current time-block operations, each cluster is evaluated to decide even if it is an event or not. For this aim the growth rate of cluster is used. The growth rate is calculated using the number of tweets which shapes a cluster using the

equation below:

$$\text{cluster\_growth\_rate(C)} = \frac{|\{t_{added} \in C\}|}{|\{t_{all} \in C\}|} \tag{3.6}$$

where,

- $|\{t_{added} \in C\}|$: the number of tweets merged to cluster C during the last round execution.
- $|\{t_{all} \in C\}|$: total number of tweets belongs to the cluster C from the beginning of the execution.

To sign a cluster as an event, the cluster_growth_rate of the word should be greater than the specified threshold.

3.3 Hybrid method for event detection

Our last method combines the previous two methods in order to increase the efficiency of clustering by pre-elimination using *tf-idf* values of words and by filtering tweets not including bursty keywords. First, bursty keywords are found by using 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 3.1 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 after an additional pre-processing step to turn them into tweet vectors). Similar two previous methods, hybrid method also processes data in rounds.

3.3.1 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 common words for burstiness. As in the first method, burstiness of a word is checked through the increase in its tf-idf value. The calculation of tf, idf, and tf-idf are as given in Equations 3.1, 3.2 and 3.3. The increment rate of the tf-idf values should n-be greater than a threshold value in order to consider a keyword 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.

3.3.2 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 one or more bursty terms are fed into the clustering phase. By this way, time efficiency can be significantly improved.

3.3.3 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 technique.

3.4 Illustrative Example

We illustrate the basic processing for each of the three methods on a simple, hypothetical example. As given in Table 3.1, our example includes four rounds. Thresholds used in illustrative example are:

- threshold to be a bursty word: at least two same words should be appeared in tweets
- threshold to be a bursty cluster: double the number of tweets included.
- threshold to be an inactive cluster: inactive for one round.
- threshold to be an underweighted word cluster: weight is below 0.5.

Table 3.1: A Simplified Example

Rounds	**Tweets**	**Bursty Keywords (Keyword)**	**All Detected Clusters (Clustering)**	**Bursty Clusters (Clustering)**	**All Detected Clusters (Hybrid)**	**Bursty Clusters (Hybrid)**
$Round_1$	t_1 includes *superbowl* t_2 includes *patriots* t_3 includes *oscars* t_4 includes *shooting*	—	*C1*: $\{t_1, t_2\}$ *C2*: $\{t_3\}$ *C3*: $\{t_4\}$	—	—	—
$Round_2$	t_5 includes *superbowl* t_6 includes *oscars* t_7 includes *superbowl* t_8 includes *oscars*	oscars superbowl	*C1*: $\{t_1, t_5, t_7\}$ *C2*: $\{t_3, t_6, t_8\}$	*C1*: about superbowl *C2*: about oscars	*C1*: $\{ t_5, t_7\}$ *C2*: $\{ t_6, t_8\}$	—
$Round_3$	t_9 includes *oscars* t_{10} includes *iphone* t_{11} includes *oscars* t_{12} includes *apple*	oscars	*C2*: $\{t_3, t_6, t_8, t_9, t_{11}\}$ *C4*: $\{t_{10}, t_{12}\}$	*C2*: about oscars	*C2*: $\{ t_6, t_8, t_9, t_{11}\}$	*C2*: about oscars
$Round_4$	t_{13} includes *oscars* t_{14} includes *iphone* t_{15} includes *hurricane* t_{16} includes *apple*	—	*C2*: $\{t_3, t_6, t_8, t_9, t_{11}, t_{13}\}$ *C4*: $\{t_{10}, t_{12}, t_{14}, t_{16}\}$ *C5*: $\{t_{15}\}$	*C4*: about iphone	—	—

The streaming tweets for each round are shown in the second column of the table. The third column shows the result for the keyword-based method. Being the initial one, in the first round, there are no detected bursty keywords yet. However, in this round, the weight of each term is calculated for comparison in the following rounds. As the weight value, in this example, we simply consider *frequency* of the words, and focus on the words given in the second column. Due to the increase in the frequency of *superbowl* and *oscars* in second round, these two words are detected as bursty keywords, possibly indicating the events of *Superbowl 2018 Finals* and *2018 Oscar Awards*. In the third round, the keyword *oscars* has still considerable presence, hence it is marked as bursty keyword in this round as well. In round 4, there is a drop in the frequency of this term, hence it is not detected as a bursty term anymore.

The clustering-based method is illustrated in the fourth and fifth columns. The column with label *All Detected Clusters* includes all the constructed clusters, whereas the other one shows those that are detected as *event clusters* in the current round. In the first round, three clusters are created, which are candidates for the event detection. The details of the similarity calculation are not given, but we can assume that t_1 and t_2 are about Superbowl finals and have similar content. In the second round, out of these three clusters, the first and the second ones grow further, and they are detected as bursty clusters, but $C3$ is eliminated, since it is inactive for the last round. In the third round, $C1$ is deleted from cluster list since it is inactive for the current round. In this round, a new cluster, $C4$, is detected. Since $C2$ includes two more tweets in round 3, it is detected as a bursty cluster in this round, as well. In the last round, $C2$ and $C4$ are still kept as detected clusters, and additionally, a new cluster, $C5$ is generated. In this round, as a difference from the keyword-based method, cluster $C4$, which includes tweets on iphone and apple, is detected as bursty cluster, possibly denoting an event of *Launching of a new iphone model*.

Finally, in the last two columns, the basic steps of the hybrid method is shown. Its processing is similar to the clustering-based method. The basic difference is that clusters are always constructed out of tweets containing bursty keywords. Therefore, cluster $C3$ is not generated at all and $C1$ and $C2$ are generated at round 2 and detected as event when it doubles its size at round 3. As in $C3$, since it does not include any bursty keywords, in the last round, $C4$ is not generated or detected as event cluster.

CHAPTER 4

IMPLEMENTATION

In this section Apache Storm topologies and the components of topology for each event detection technique, namely keyword-based event detection, clustering-based event detection and event detection with a hybrid technique are described and discussed.

4.1 Twitter API

Twitter provides REST APIs and Streaming APIs to share the some percentage of the public user's tweets so that researches and analyses can be carried out by using these data. REST API is used to send request to have information about tweets, users, locations or other objects of Twitter data and get JSON or XML formatted responses as common. On the other hand Streaming API provides a stream of Twitter data which can be filtered by a desired criterion.

In this thesis, the Java library of Twitter Streaming API, namely Twitter4j [61] is used, and the location filter property of Twitter4j is used to eliminate data came from other countries rather than USA and Canada. For the experiments and evaluation phase, the data came from streaming api is saved to Apache Cassandra database for 10 days and each experiment runs on the same part containing 7-days data of this collected data. Another important point is that we have collected any kind of data, collection is not topic based, i.e during data collection from Twitter no filter other than location filter is used. The variety of data makes the execution more complex and makes the system heavily loaded.

4.2 Apache Storm

Due to the performance constraints we use Apache Storm as real-time distributed computation system which is originally created by Nathan Marz [62] and team at BackType [63] and become open source after being acquired by Twitter [64]. Storm makes processing unbounded streams of data in real-time possible. Storm can be used in any programming language. Some use cases of Storm are real-time analytics, online machine learning and distributed remote procedure call. The main advantage of Storm is its efficiency, i.e Storm can process a million tuples per second per node as a benchmark. Besides Storm is scalable, fault-tolerant, guarantees your data will be processed and is easy to set up and operate.

At the beginning of the project, we decided to use a distributed computing platform to process big data by using the latest technologies and adapt event detection algorithms on top of one of the real-time distributed computing framework for efficiency and scalability. Apache Storm and Spark are two powerful candidates for this purpose. Since Apache Storm fastens up the traditional processing and design as a real-time distributed system, it provides all requirements for implementing business intelligence and analytics in real time. On the other hand, Spark also has similar capabilities but it is more of a general-purpose distributed computing platform while Storm is a stream-oriented distributed computing platform. Additionally, Spark works by chaining successive method calls as opposed to the Storm model driven by creating classes and implementing interfaces. Another advantage of Storm is that there are number of existing spouts which can directly be used in projects like Twitter streaming spout. Therefore, Apache Storm is deployed as real-time distributed stream processing platform in this study.

For an efficient and reliable computation, we have used Apache Storm in our project. Apache Storm can be used in local mode during implementation since it creates Storm nodes on local machine and makes implementation and debugging easier or it can be used in cluster mode which enables the distributed computation. When Storm is used in cluster mode, it creates distributed clusters and manage them automatically after their configurations are done.

There are just three abstractions in Storm: spouts, bolts, and topologies.

- Storm uses "spouts" as source of streams in a computation. Spout even reads from a queuing broker such as Kafka or RabbitMQ or generates its own stream or read from somewhere like the Twitter streaming API or from a database like Apache Cassandra. For our project we replicate the tweets stored in Apache Cassandra database in time order.

- Storm uses "bolt" to process any number of input streams and produce any number of new output streams. Most of the logic of a computation goes into bolts, such as functions, filters, streaming joins, streaming aggregations, talking to databases, and so on.

- Storm uses "topology" as a network of spouts and bolts, with each edge in the network representing a bolt subscribing to the output stream of some other spout or bolt. A topology is an arbitrarily complex multi-stage stream computation. Topologies run indefinitely when deployed.
  Each spout or bolt executes as many tasks across the cluster. Each task corresponds to one thread of execution, and stream groupings define how to send tuples from one set of tasks to another set of tasks. Parallelism for each spout or bolt is set by developer.

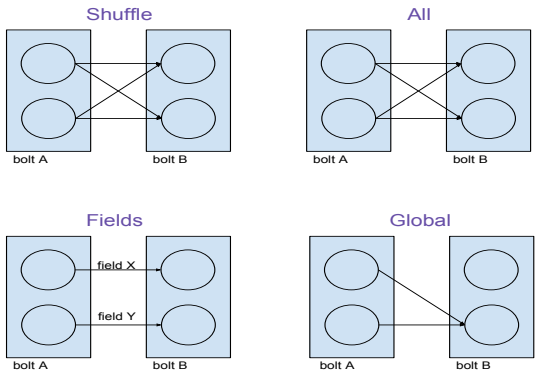


Figure 4.1: Storm grouping types (taken from [1])

As seen in Figure 4.1 Apache storm provides five grouping types:

  - Shuffle grouping: This type of grouping distributes stream among bolt

tasks randomly and fairly. Distribution is handled by Apache Storm.

- All grouping: This type of grouping replicates stream to all bolt tasks.
- Fields grouping: This type of grouping partitions stream on a user-specified field.
- Global grouping: This type of grouping gets entire stream to single task.
- Direct grouping: This is a special kind of grouping. A stream grouped this way means that the producer of the tuple decides which task of the consumer will receive this tuple. Therefore, distributing tuples among tasks fairly depends on the developer.

4.3 Keyword-based Event Detection Method

The Apache Storm topology of keyword-based event detection method can be seen in Figure 4.2. The user parameters and the equivalents of Apache Storm abstractions in experiments are given in the following subsections.

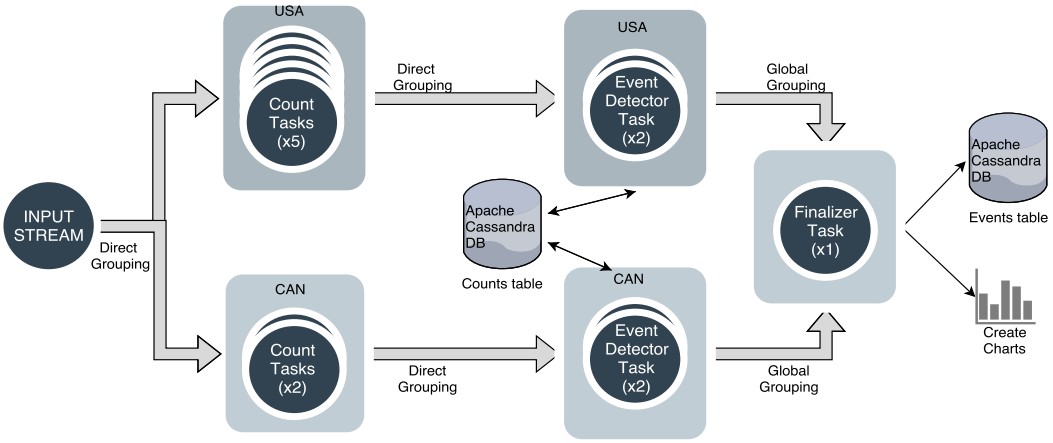
4.3.1 User defined parameters

There are 2 parameters defined by the user:

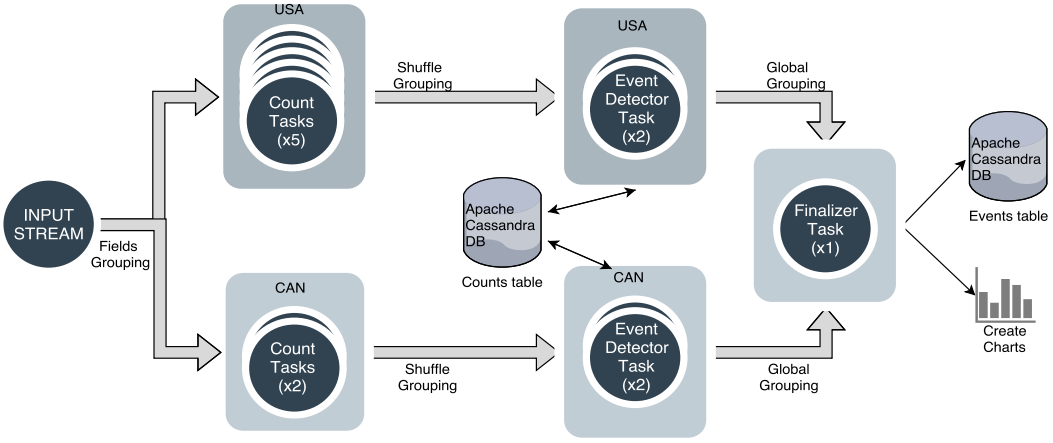
- TF-IDF_EVENT_RATE: This parameter is used to define the increment rate of tf-idf value between the last two rounds to be an event. i.e. If this parameter is 10 and tf-idf values for last two are 0.001 and 0.015 in order, then this keyword should be marked as event since 0.015/0.001 greater than 10.
- COMMON_WORD_THRESHOLD: This parameter defines the threshold number which is used to presume a word as common word. Only the common words are subject to the keyword-based event detection algorithm.

4.3.2 Source Of Stream

As a source of stream, data saved to Apache Cassandra database is replicated document by document in order where document corresponds to a collection of tweets



(a) Keyword-based Event Detection Topology with Direct Grouping



(b) Keyword-based Event Detection Topology with Sleep Intervals

Figure 4.2: Keyword-based Storm Topologies

in a 6-min time-block. When replication of tweets inside a document is finished, spout does not start replication of the next document immediately. Next document replication is suspended until the process of the current document is completed. The reason of suspension between document processes is needed for the reliability of the system and the correctness of the event detection, i.e. if the next document streaming is started immediately, before the current document has finished processing, there is a high possibility that the words in the next document will interfere the current document which will result in wrong event detection for current document. Therefore, it is needed to define a streaming protocol which handles suspension. Apache Storm does not provide a methodology for this purpose, hence in our project we describe

two different approaches. First one is sleeping spout for some time determined by experiments between rounds and the second one is using direct grouping defined by Storm so that we are able to check whether current round is finished.

Our first approach uses shuffle grouping and fields grouping where scheduling and distribution of data among tasks is handled by Apache Storm and use sleep intervals between rounds to prevent mixing multiple rounds. This approach leads to a more efficient distribution and processing time since data is distributed by Storm scheduling mechanism, but the sleeping buffer used between rounds turn the advantage to the disadvantage. The main reason of the low efficiency of this first approach is that the process time of each round differs from each other since the volume of data for each 6-min interval in a day differs and the sleeping buffer is selected as the longest time. In summary, spout needs to wait(sleep) some time decided by experiments so that at the end of this time all tasks will finish their executions for the current document.

Second approach uses direct grouping techniques and direct streaming for each task where Storm gives the control of distribution and scheduling mechanism to developer. In this approach we simply distribute the data in turn to each task without a task availability check. Because of this distribution mechanism, efficiency is not as well-handled as first approach, because the availability of the tasks can not be controlled while distributing the tuples in order. On the other hand the advantage of this method is not losing time by sleeps. Therefore, for the experiments second approach is selected since it increases performance.

To summarize, the spout is labeled as 'INPUT STREAM' in Figure 4.2 and it reads tweets from Apache Cassandra database one by one in order, split tweets into words, preprocess them and emits them to next bolt.

4.3.3 Word Counts

Streaming tuples firstly splitted into words and emitted to word count bolt. Tasks of word count bolt simply count the number of words for the current document. Count of the words are used to decide whether the word may represent an event or not. Since performance is important for big data analysis methodologies, there is a need

to initial elimination of words. For our approaches we use a word count threshold which is defined by COMMON_WORD_THRESHOLD to eliminate some of the words which occur rarely in documents. By this pre-elimination we can define the most common words in current document and avoid unnecessary calculations for the uncommon words.

For the first approach(using sleep intervals), fields grouping is preferred, i.e for each tweet same words are emitted to same tasks by Apache Storm's distribution mechanism (For example, if the first occurrence of word "hello" is emitted to Task 1 and word "world" is emitted to Task 2; then for the following tweets, spout always emits word "hello" to Task 1 and word "world" to Task 2). We have imitated this behavior for the second approach which uses direct grouping and the distribution of tuples is handled by us. For this purpose we have keep a list of task and word list so that same words are emitted to the same task.

To summarize, word count bolts are responsible to hold count of each word separately until the end of current round. This bolt emits the word to next bolt for tf-idf calculation phase *only once* when count of the word reached to the common word threshold *immediately*.

4.3.4 Event Detector Bolt

This bolt stores the common words emitted from word count bolt until word count bolt finishes its process for each word in current document. Then tasks of this bolt starts to calculate the tf-idf value of each common word for the last two rounds and these tf-idf values are used to calculate the increment rate of common words to decide if this word is an event or not.

4.3.5 Event Compare Bolt

This bolt gets all events detected by Event Detector Bolt and has two simple jobs. First one is storing event keywords to Apache Cassandra database and drawing line chart for the event keyword which shows the counts of the word for the last 10 rounds.

Three different charts created by this bolt are concatenated and given in Figure 4.3.

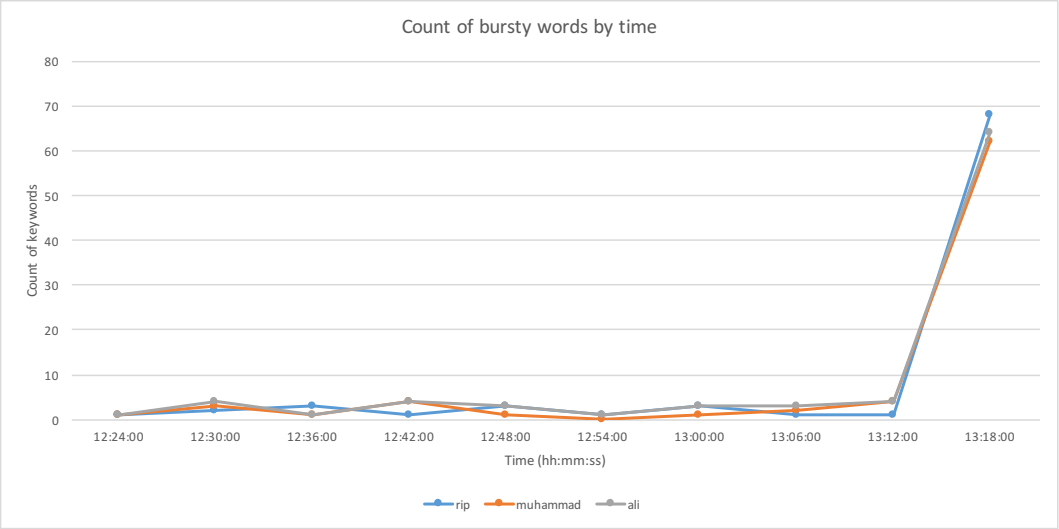


Figure 4.3: Count of *bursty* words by time

4.4 Clustering-based event detection method

The Apache Storm topology of cluster-based event detection can be seen in Figure 4.4. The user parameters and the equivalents of Apache Storm abstractions in experiments are given in the following subsections.

4.4.1 User defined parameters

There are 1 parameter defined by the user:

- NUM_TWEET_THRESHOLD: During the computation of rounds, many clusters are created. Therefore, it is not efficient to evaluate each cluster at the end of each round and this threshold is used to eliminate clusters containing fewer tweets.

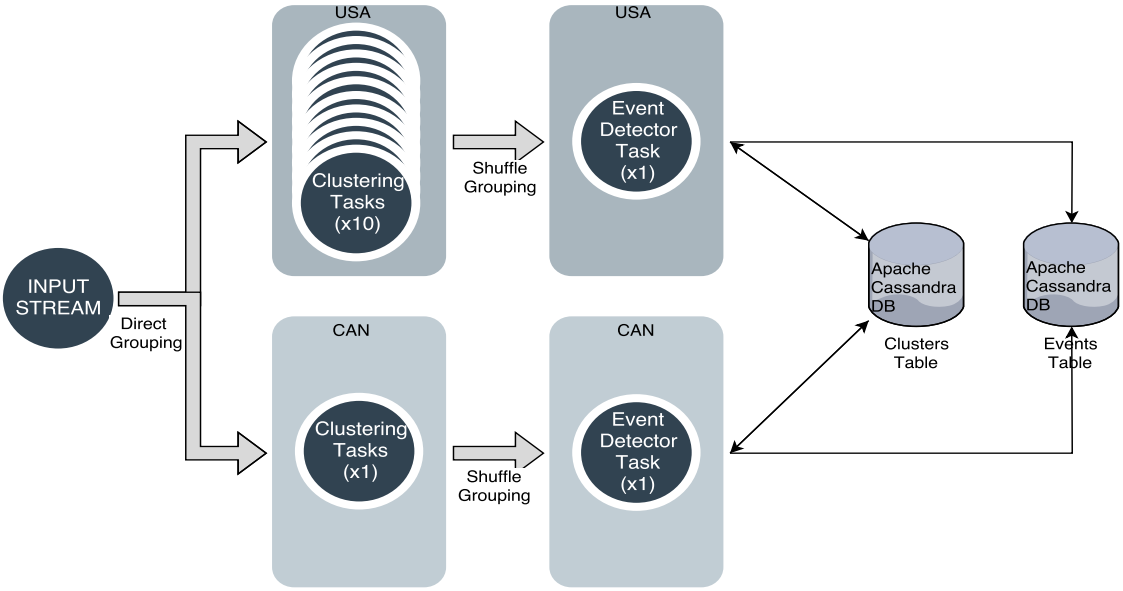


Figure 4.4: Clustering-based Storm Topology

4.4.2 Source Of Stream

"INPUT STREAM" of this method works very similar to the keyword-based event detection topology. Spout gets tweets from Apache Cassandra database in tweet-time order and emit them by direct grouping to the next bolt. Similar to previous method, spout waits until all bolt tasks finish their process for current document before starting to replicate the next document. The difference is that spout does not split tweet into words for this methodology. Spout vectorizes the tweets and directly emits the tweet vector since clustering is based on the cosine similarity of the whole tweet vector. Tweet vector shows the words inside a tweet with the normalized weight as a map. For example the vector of the tweet "RIP Muhammed Ali RIP" is {"RIP":0.5, "Muhammed":0.25, "Ali":0.25}. Clusters are also represented with weighted vectors. Therefore, these weighted vector representations are used to calculate the cosine similarity between tweets and clusters, too.

To summarize, the spout is labeled as "INPUT STREAM" in Figure 4.4 which reads tweets from Apache Cassandra database one by one in time order, vectorizes tweets into weighted vector maps and emits tweet vectors to next bolt.

4.4.3 Clustering Bolts

Clustering bolts are responsible to assign tweets into clusters by using cosine similarity. As mentioned in the previous chapter, there are two clustering algorithms. First one connects to database at each time when a tweet vector is evaluated for cluster assignment which results in very low performance and blocks the real time processing. Second method uses a two-step clustering for performance improvement. In this section, clustering bolt using two-step clustering methodology is explained.

This bolt is responsible for only local clustering step which means at the beginning of a document process, there exists no clusters in the local list of tasks. Each task handles its own cluster list, creates/updates clusters with the tweet vectors distributed by input stream and at the end of process each clustering task emits the list of clusters to next bolt for evaluation. For cluster assignment this bolt calculates the cosine similarity between tweet vector and existing local clusters and if the cosine similarity of a tweet vector and a cluster vector is higher than the specified threshold, tweet is assigned to the cluster and cluster vector is updated accordingly. If cosine similarity constraint is not met for any clusters, then new cluster is created for the tweet vector. At the end of the round, all the clusters created during block streaming is emitted to the next bolt which is event detection bolt explained below.

4.4.4 Event Detector Bolt

This bolt is activated at the end of the 6-min block streaming. Each clustering bolt task sends the local cluster list to event detector bolt and event detector bolt stores them until all tasks send their lists for the current round. When all the local cluster lists of each task of previous bolt has arrived, event detector bolt starts evaluation of local clusters. This evaluation consists of two steps:

- Local Cluster Evaluation: First of all this bolt merges local clusters created by different tasks. Merge operation takes place if the cosine similarity of two local clusters is higher than or equal to the specified threshold. During merge operation cluster word map is updated by re-calculating the weight of each

word and reducing two cluster vectors to one. Also word map of merged cluster is re-evaluated to avoid the sparse word maps by deleting insignificant words from cluster word map. For this purpose after merge operation, word map is examined and the words which has weight smaller than a specified threshold are deleted from map. At the end of merge operation, clusters representing less than a specified number of tweets are also deleted from local cluster list.

- Global Cluster Evaluation: After local evaluation, this bolt gets the list of previous clusters from Apache Cassandra database and this time, it merges the local clusters come from previous tasks and global clusters held by database. For this step, the cosine similarity of each global cluster is calculated for each local cluster one by one and the global cluster is updated locally (without affecting database[1]) until all the similar local clusters are merged into this cluster. This merge operation is also followed by the cluster word map re-evaluation step where words having smaller value than a threshold as weight are deleted from map. After each similar local cluster is merged into the global cluster and weight updates, database entry of this global cluster is updated. In case of merge operation, this cluster also becomes an event candidate. For the updated global cluster, growth rate is calculated by using Equation 3.6 and if growth rate is bigger than the specified threshold, it is marked as event for the current round. After merge operations, remaining local clusters are updated and added to database. As the final step, global clusters are eliminated for performance. The database entry of inactive clusters which are not updated for the last 3 rounds are deleted from database.

4.5 Hybrid method for event detection

The Apache Storm topology of hybrid method of event detection can be seen in Figure 4.5. The user parameters and the equivalents of Apache Storm abstractions in experiments are given in the following subsections.

[1] This detail is very important since database interactions are costly and this system connects to db at most one time for each cluster in a round

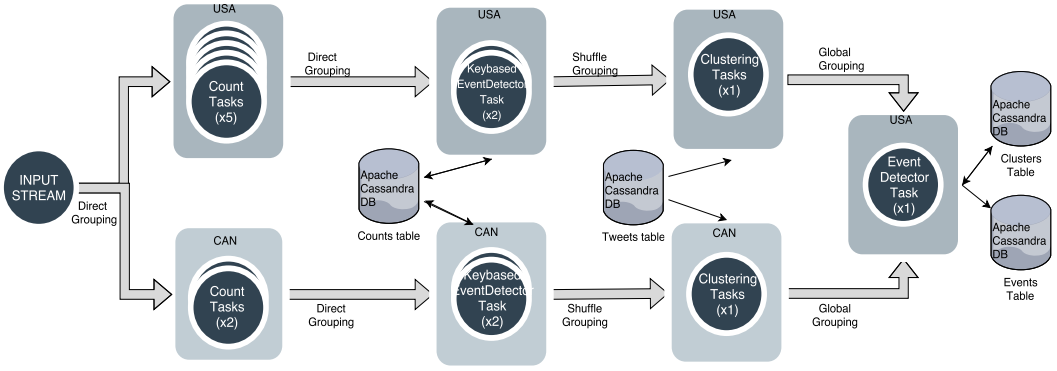


Figure 4.5: Hybrid Storm Topology

4.5.1 User defined parameters

There are 3 parameters defined by the user:

- TF-IDF_EVENT_RATE: This parameter is used to define the increment rate of tf-idf value between the last two rounds to be an event. i.e. If this parameter is 10 and tf-idf values for last two are 0.001 and 0.015 in order, then this keyword should be marked as event since 0.015/0.001>10.
- COMMON_WORD_THRESHOLD: This parameter defines the threshold number which is used to presume a word as common word. Only the common words are subject to the uncommonly common algorithm.
- NUM_TWEET_THRESHOLD: During the computation, many clusters are created. Therefore, it is not efficient to evaluate each cluster at the end of each round and there should be a threshold which points out the number of tweets in a cluster.

4.5.2 Source Of Stream

"INPUT STREAM" of this method works same as the keyword-based event detection topology of direct grouping. Spout gets tweets from Apache Cassandra database in order, splits tweet into words, preprocess them and emit words by direct grouping to the next bolt. Similar to other methods, spout waits until all bolt tasks finish their

process for current round before starting to replicate the next round.

4.5.3 Word Counts

This word count bolt works same as the word count bolt of keyword-based methodology. Words are splitted from tweets and emitted from spout to this word count bolt by direct grouping. These word count bolts is responsible to count the words occurred in the current round and emit them when the count reaches the specified threshold. By using this bolt, we only work with the words which are very commonly used in the current round for performance issues.

To summarize, count bolts receive and store the count of each word separately and emits the word to next bolt for tf-idf calculation phase once when count of the word reached to the threshold immediately.

4.5.4 Keyword-based Event Detector Bolt

This bolt stores the words emitted from word count bolt and when the previous bolt finishes its task for current round, this bolt starts to calculate the tf-idf value of last two documents for each common word. By using the tf-idf values calculated, this bolt calculates the increment rate and decides whether this word is an event or not. If the common word is marked as event keyword, it is emitted to next bolt.

4.5.5 Clustering Bolts

This bolt is activated at the end of the 6-min block streaming. Clustering bolt of hybrid approach, firstly gets the event keywords and eliminate the tweets which does not contain any of the common words. Then using the filtered tweets, it starts two-step clustering algorithm: local and global clustering same as clustering-based method. Local and global clustering algorithms are executed on different bolts: Clustering and event detector bolts. At the end of the global clustering, events are also stated by using growth rate 3.6 of cluster. Both of the clustering steps use cosine similarity for merge operations again. While local clustering step uses cosine similarity between

tweet and cluster vectors, global clustering step uses cosine similarity between two clusters.

- Local Clustering: Clustering bolts are responsible to get the list of event candidate words, eliminate current block tweets containing any of these words and then start local clustering phase. Steps of local clustering is as follows:
  - Tweet elimination: Eliminate the tweets which does not contain any of the event candidate words.
  - Local clustering: This steps starts with an empty list of local clusters and creates/updates clusters by cosine similarities and also updates vector representation of clusters by using the same operations of clustering-based method.
  - Merging cluster: At the end of the current round before emitting the list to next bolt, this bolt merges local clusters. For this step one by one each cluster is compared with other clusters and when cosine similarity of two clusters are greater than a specified threshold, these clusters are merged and the vector representation of merged clusters is updated same as the clustering-based method.
- Global Clustering: After local clustering, cluster list is emitted to the event detector bolt and this bolt compares the global clusters with local clusters and updates the global clusters. Clusters are marked as events regarding to the growth rate of the cluster. Steps of global clustering is as follows:
  - Eliminate local clusters: If a local cluster represents less than a specified number of tweets, it is very low probability to be an event. Therefore, it is removed from local cluster list.
  - Assign local clusters to global clusters: This step is same as the clustering-based method, i.e. all the local clusters are examined and if the cosine similarity of the global cluster and the local cluster is above the specified threshold, local cluster is assigned to global cluster and global cluster is updated. After a global cluster goes over the whole local cluster lists, the word map of the global cluster is cleaned by removing the under-weighted

words and Apache Cassandra table entry of this global cluster is updated. Finally the updated global cluster is evaluated to be an event or not by looking at the growth rate.

- Inactive global cluster removal: During global cluster assignment, if there is no assignment to a global cluster and if it is inactive for the last two blocks, cluster is removed from global cluster list.

- New cluster addition to global list: After relevant local clusters are assigned to global clusters, there may occur some new clusters which are not related to any of the global clusters. The word map of this new clusters are also updated by removing the words having low weights and added to database as global clusters.

4.6 Geolocation

Since Twitter is used all over the world, Twitter data contain all kind of events like new album announced by an artist who is known world-wide or an election of a country which can be labeled as local event. Therefore, tweets posted from each country or may be each city can be processed separately to identify if it is local event or it is an event which concerns the whole world. If a user on Twitter gives permission to use their location information, Twitter provides this by API. By using the filtering option of Twitter Streaming API, we have stored tweets posted from Canada and USA into Apache Cassandra with their location information. In the system proposed in this study processes tweets in two parallel streams and these two streams have exactly same process steps with different parallelism setup according to data volume. As seen in Figures 4.2, 4.4, 4.5, input stream can identify where tweet is posted from and direct tweets due to their location information to the related bolts. By doing so events are detected with their locations. Despite this study is interested in only Canada and USA, it is easy to plug new processing line for different countries.

4.7 Database

Apache Storm topologies are executing the methodology in a distributed fashion and the setup contains a single local cluster with many tasks on a single machine, but still since the system hires many tasks for each process steps and creates many threads, the need for a storage to handle state of the system and related data rises and this need is fulfilled by Apache Cassandra. Apache Cassandra is a NoSQL database providing scalability and high availability. Since data is replicated to multiple nodes, the system provides fault tolerance. Every Apache Cassandra node is identical on the system, therefore there is no single point of failure and there are no network bottlenecks. Apache Cassandra is in use at big companies like Netflix, Instagram, Reddit, eBay, etc [65].

There are two use cases Apache Cassandra is integrated to system. First one is storing data used or created from system like tweets or events detected at the end of process. Second one is to provide stateful stream processing, for example to store created information at the current state like count of words or global clusters at current time window. Besides the state of tasks are stored in Cassandra like if the task finished its job for the current window or not or how long does it take to finish its process. The detailed description of tables is given in Appendix B.

CHAPTER 5

EXPERIMENTS AND RESULTS

In this section, we present the experimental study. The goal of this study is to test and compare the three online event detection methods in terms of their accuracy and performance. In what follows, we first describe our experimental setting. Then we discuss an initial set of experiments that we conducted to tune our algorithm parameters along with the methodology that we used to validate their settings. Finally, we present our results for event detection accuracy and performance.

5.1 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.

For the experiments, nearly 12M tweets collected from Twitter within one week from May 31, 2016, to June 7, 2016, is used. 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 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 Precision/Recall/F-measure for accuracy, and total execution time of processing the whole dataset for performance. Specific details of

these metrics are discussed in the following subsections.

5.2 Preprocessing

Preprocessing step is applied in all three methods and the aim of this step is making tweets more meaningful by converting verbs into base forms and filter unimportant words like at, the or urls, etc. Some examples from the preprocessing step can be found in Table 5.1.

Table 5.1: Preprocessing input and outputs

Input	**Output**
We're proud to join 160+ companies in @HRC's petition to #RepealHB2, because we believe in equal rights for all:	be, proud, join, company, @hrc, petition, #repealhb, believe, equal, rights
TODAY WAS MY GRANDPARENTS 65TH WEDDING ANNIVERSARY AND I THREW FIRST PITCH AT @GoSquirrels AND I DEADLIFTED 295 SO IT WAS A GOOD DAY.	oday, grandparents, 65th, wedding, anniversary, throw, first, pitch, @gosquirrels, deadlift, good, day
Adding space. . . in space. Watch as BEAM module attaches to @Space_Station tomorrow at 5:30am ET http://go.nasa.gov/1SgkTSv	add, space, space, watch, beam, module, attach, @space_station, tomorrow, 5:30, et
Donate to HRC for a chance to win 2 tickets to @springsteen's NYC show. Enter on @crowdrise at http://crowdrise.com/nohate #nohateinmystate	donate, hrc, chance, win, ticket, @springsteen, nyc, show, enter, @crowdrise, #nohateinmystate
Made the ray marcher into a 3.5 KB html page: http://doodle.notch.net/unmandelboxing/ IE seems to dislike it.	make, ray, marcher, 3.5, kb, html, page, ie, seem, dislike

Table 5.1 continued

Woke up before 8 am for the first time in a long time, and I kind of like it. So many hours of sunlight ahead of me! Coffee? Need coffee.	wake, first, time, long, time, kind, like, many, hour, sunlight, ahead, coffee, need, coffee
I only follow'ed you bc I was waiting for you to finish http://cliffhorse.com . Now, I follow bc you're damn funny.	follow, ', ed, wait, finish, now, follow, be, damn, funny
lol afaik this is the best thing related to league of legends omg ahahah	best, thing, related, league, legend, ahahah
I liked a @YouTube video http://youtu.be/Vi1tkUNpohM?a Cologne Shop Roleplay (ASMR)	-
I'm at @AstirBeach in Vouliagmeni, Athens https://www.swarmapp.com/c/7lwQrizYC06	-

5.3 Parameter Tuning and Validation

The clustering-based event detection method (and therefore, the hybrid method which is also based on clustering) involves several parameters used as thresholds. In order to determine optimal values for these parameters that will create a fair and comparable setting against the keyword-based method, we conducted an initial set of tuning and validation experiments. These threshold parameters include:

- Cluster clean-up: To keep a cluster clean and simple, it is needed to eliminate unnecessary or irrelevant words when a cluster gets crowded. A word in a cluster is removed if:
  - a. Number of tweets forming the local cluster $> p_1$ and
  - b. Weight of word $< p_2$

- Cosine similarity threshold(p_3): This threshold shows the minimum cosine similarity score needed between two clusters to merge them.

- Number of tweets forming the cluster(p_4): If the number of tweets residing in a cluster is less than this threshold, the cluster cannot be a candidate of an event, it is removed.

- Minimum weight of a word in a new cluster assignment(p_5): When a new cluster is added to global list, this cluster is also cleaned up. Words having smaller value than this threshold are eliminated from cluster.

The first set of validation experiments are conducted on a smaller sample of the whole dataset with 15 rounds, including the tweets about the event *Death of Muhammad Ali* (i.e., a significant event that took place on June 3, 2016 – during the week that we collected data from Twitter), and the resulting clusters are manually analyzed. Tested parameter value settings and the number and quality of clusters that each setting led to are shown in Table 5.2 along with our observations. For measuring cluster quality, we use the *silhouette coefficient*. The silhouette coefficient essentially measures how similar a given object is to its own cluster compared to the other clusters [66], given in Equation 5.1. Its value ranges between -1 and +1, where a higher value indicates higher clustering quality. Silhouette coefficient is used for both parameter tuning and accuracy comparisons.

$$SC_{cluster}(C) = \frac{\sum_{t \in C} SC_{tweet}(t)}{|\{t \in C\}|} \tag{5.1}$$

where,

- t: current tweet inside cluster C, similarity of which is calculated.

- C: current cluster, cosine similarity of which is calculated.

- $|\{t \in C\}|$: the number of tweets shaping the cluster C.

Table 5.2: Parameter Tuning and Validation Results (small dataset)

	p_1	p_2	p_3	p_4	p_5	**Number of Clusters**	**Silhouette Coefficient**	**Comments**
Setting 1	30	0.005	0.3	30	0.05	23	0.75	Cluster term vectors include high number of words. Many non-event clusters are formed. This setting leads to the lowest accuracy for event detection.
Setting 2	40	0.01	0.4	50	0.04	7	0.79	A small number of crowded clusters containing irrelevant word sets are generated. Many non-event clusters are formed.
Setting 3	50	0.01	0.5	100	0.05	3	**1.0**	Three clusters are formed and each of them is related to the event *Death of Muhammad Ali*. No non-event clusters are formed.
Setting 4	60	0.025	0.6	120	0.06	1	**1.0**	Only one cluster is generated. It is related to the event *Death of Muhammad Ali*.
Setting 5	70	0.05	0.7	150	0.07	0	0.0	No clusters are generated.

Table 5.3: Detailed Validation Results for Setting 3 and Setting 4 (full dataset)

	Method	Country	Number of Clusters	Avg. S. C.	Std. dev.
Setting 3	Clustering	USA	74	**0.855**	0.276
	Clustering	CAN	7	**1.0**	0.0
	Hybrid	USA	15	0.342	0.49
	Hybrid	CAN	0	NaN	NaN
Setting 4	Clustering	USA	34	0.748	0.268
	Clustering	CAN	2	1.0	0.0
	Hybrid	USA	6	**1.0**	0.0
	Hybrid	CAN	0	NaN	NaN

$$SC_{tweet}(t, C) = \frac{MinDistToOtherClusters(t, C, C') - AvgDistInsideCluster(t, C)}{max(MinDistToOtherClusters(t, C, C'), AvgDistInsideCluster(t, C))} \tag{5.2}$$

where,

- AvgDistInsideCluster(t, C): average distance between tweet t and other tweets within the same cluster C.
- MinDistToOtherClusters(t, C, C'): the lowest average distance of tweet t to all other tweets in any other cluster C'.

$$AvgDistInsideCluster(t, C) = 1 - \frac{\sum_{t' \in C} CosineSimilarity(t, t')}{|\{t' \in C\}|} \tag{5.3}$$

where,

- CosineSimilarity(t, t'): Cosine similarity of t and t' given in 3.5 where t' is other tweet than t inside same cluster C.
- $|\{t' \in C\}|$: the number of tweets shaping the cluster C.

$$MinDistToOtherClusters(t, C, C') = min([AvgDistInsideCluster(t, C'_1), \\ ..., AvgDistInsideCluster(t, C'_n)]) \quad (5.4)$$

where,

- C: cluster where tweet t belongs to.
- t: tweet inside cluster C.
- [AvgDistInsideCluster(t, C'_1), ..., AverageDistInsideCluster(t, C'_n)]: list of average distances where t is the current tweet inside cluster C evaluated and C'_x is the cluster other than current cluster C.

Threshold set values, definitions and comments below:

- **Set 1** First set includes low threshold values:
  - If the number of tweets in cluster is greater than 30 and weight of key is less than 0.005, key will be removed from cluster. This is very light condition. Most of the words will stay in the cluster and clusters will expand more than others.
  - If cosine similarity of two clusters is greater than 0.3, they will merge. Therefore, there will be less duplication of clusters since most of the similar clusters will merge.
  - If the number of tweets in a cluster is greater than 30, it is saved as global cluster for next rounds. Therefore, there will be more global clusters saved to db than other sets which will decrease the performance.
  - Minimum weight of a word in a new cluster assignment is 0.005 which is also a light condition. Most of the words will stay in the newly created clusters and this set will let clusters to expand more than other sets.
- **Set 2** Second set includes higher threshold values but still low :
  - If the number of tweets in cluster is greater than 40 and weight of key is less than 0.01, key will be removed from cluster. This is also a light

condition. Most of the words will stay in the cluster too but elimination will be better than the first set.

  - If cosine similarity of two clusters is greater than 0.4, they will merge. Therefore, there will be less duplication of clusters when compared to next threshold sets.
  - If the number of tweets in a cluster is greater than 50, it is saved as global cluster for next rounds. Therefore, there will be less global clusters than first set but more global clusters than next sets which will also decrease the performance by increasing the db transactions.
  - Minimum weight of a word in a new cluster assignment is 0.04 which is heavier than first set. Words will eliminate in a more appropriate way and cleaner clusters will shape.

- **Set 3** Third set includes medium weight threshold values:

  - If the number of tweets in cluster is greater than 50 and weight of key is less than 0.01, key will be removed from cluster. This is a medium condition. Most of the event related words and some of irrelevant words will remain in clusters.
  - If cosine similarity of two clusters is greater than 0.5, they will merge. Therefore, there may be cluster duplicates since this threshold may force the create new clusters for same events with mostly different selection of words but the number of duplicates should not be much.
  - If the number of tweets in a cluster is greater than 100, it is saved as global cluster for next rounds. Therefore, there will be fewer clusters than the sets mentioned until now.
  - Minimum weight of a word in a new cluster assignment is 0.05 which is also a medium value. Hence, even newly created global clusters will be clean.

- **Set 4** Fourth set includes higher threshold values:

  - If the number of tweets in cluster is greater than 60 and weight of key is less than 0.025, key will be removed from cluster. This is a heavy

condition when compared to previous sets. Most of the irrelevant words will be removed and very clean clusters will be shaped.

  - If cosine similarity of two clusters is greater than 0.6, they will merge. This condition makes it harder to merge clusters or assign tweets into a related cluster since the similarity score threshold will cause more elimination. Therefore, only very similar tweets will shape clusters and duplication becomes more likely.
  - If the number of tweets in a cluster is greater than 120, it is saved as global cluster for next rounds. This means smaller clusters will be eliminated and more crowded ones will be evaluated for event detection.
  - Minimum weight of a word in a new cluster assignment is 0.06 which is higher than previous sets. This condition may filter many words in a newly created clusters and result in small initial clusters.

- **Set 5** Fifth set is the heaviest set:
  - If the number of tweets in cluster is greater than 70 and weight of key is less than 0.05, key will be removed from cluster. This is the most restrictive condition of all five sets. Only high-weighted words which occur in the most of the tweets will remain in the cluster and the resulting clusters will be very simple and clear.
  - If cosine similarity of two clusters is greater than 0.7, they will merge. This condition makes it harder to merge clusters or assign tweets into a related cluster since the similarity score threshold is very high. Therefore, only very similar tweets will shape the clusters.
  - If the number of tweets in a cluster is greater than 150, it is saved as global cluster for next rounds. This condition forces the system to filter only the large clusters so there is a high possibility to ignore event related clusters.
  - Minimum weight of a word in a new cluster assignment is 0.07 which is higher than previous sets. This set contains the highest value for this threshold value and that results in very small and insensible clusters.

As seen in Table 5.2, Setting 3 and Setting 4 provide the best results in terms of cluster quality as well as the highest event detection accuracy results in our manual analysis.

Observing this, we then conducted a second set of validation experiments on the full dataset, specifically for comparing Setting 3 and Setting 4. The results are given in Table 5.3. Based on these results, we select Setting 3 for the clustering-based method and Setting 4 for the hybrid method in the rest of our experiments. At the end of event detection, keyword-based method labels 220 words as events for USA and 17 words for Canada. Clustering-based approach founds 74 event clusters for USA and 7 for Canada and finally hybrid method produces 6 event clusters for USA and no clusters for Canada. Details of clusters created by clustering-based method is given in Appendix A.1 and for hybrid method further information takes place in Appendix A.2. Appendices include the information when and where the event occurred. Appendix also informs about the number of tweets forming the event clusters, the highest weighted words which are the keywords for the detected events and the ground truth column shows the index of event from ground truth set given in Table 5.4.

5.4 Event Detection Accuracy and Performance

In this section, we analyze the accuracy of event detection methods against the "ground truth". Furthermore, we report our findings on their computational performance.

5.4.1 Ground Truth Construction

In order to evaluate event detection accuracy, we need a ground truth as reference for comparison since we use recall and precision measurements as in typical information retrieval and data mining context. For this purpose it is needed to enumerate all real-life events occurred between May, 31 to June, 6; but it is not feasible to find out all events occurred all over the world for a given time. Therefore, we used events detected by our system to create a ground truth set which is used as the complete set of real-life events happened between the dates of our interest. As a result, 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 events that happened within the same time interval as the dataset collection. Some 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 20 different events for the USA tweets and 4 different events for the Canada tweets, having indices 2, 6, 8 and 15 given in Table 5.4, 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.

5.4.2 Accuracy Comparison

For accuracy evaluation, we used the well-known relevance metrics of *Precision*, *Recall*, and *F-measure*. Since the output of the keyword-based method is a set of bursty keywords denoting events, precision (denoted as $Precision_k$) is calculated accordingly, as given in Equation 5.5.

$$Precision_k = \frac{Number\ of\ keywords\ matching\ some\ event}{Number\ of\ keywords\ found} \tag{5.5}$$

For the clustering-based and the hybrid methods, precision (denoted as $Precision_c$) is calculated based on the number of clusters denoting events, as given in Equation 5.6.

$$Precision_c = \frac{Number\ of\ clusters\ matching\ some\ event}{Number\ of\ clusters\ found} \tag{5.6}$$

Table 5.4: Ground Truth Event Set

	Date	**Events**
1.	31 May 2016	Thompson has written his name to NBA history by recording 11 3s in NBA 2016 Western Conference Finals.
2.	31 May 2016	Draymond Green fouls on Steven Adams in NBA 2016 Western Conference Finals.
3.	31 May 2016	Steven Adams called Stephen Curry ve Klay Thompson as quick little monkeys in NBA Western Conference Finals.
4.	31 May 2016	Penguins win against Sharks in the Stanley Cup Final, NHL.
5.	31 May 2016	Golden State Warriors complete comeback to reach 2016 NBA Finals.
6.	31 May & 2 June 2016	Predictions about 2016 NBA Finals.
7.	31 May 2016	Trailer of the movie "The Guest".
8.	1 & 6 June 2016	Ztro - FDB Ringtone
9.	3 June 2016	John Legend National Anthem in 2016 NBA Finals.
10.	3 June 2016	Stephen Curry fouls on Tristan Thompson in NBA Finals.
11.	3 June 2016	Kevin Love gets an offensive foul in NBA Finals.
12.	3 June 2016	Matthew Dellavedova fouls on Andre Iguodala in NBA Finals.
13.	3 June 2016	2016 NBA Finals, Game 1 - It was the lowest-scoring combined game for Curry and Thompson all season, yet Golden State won with ease.
14.	3 June 2016	Shaun Livingston scored 20 points in NBA Finals.
15.	4 June 2016	Death of Muhammad Ali.
16.	5 June 2016	Brock Lesnar is returning to UFC.
17.	6 June 2016	Carlos Santana's National Anthem in NBA Finals.
18.	6 June 2016	LeBron James Travels But Goes Uncalled in 2016 NBA Finals Game 2.
19.	6 June 2016	Cavaliers vs Hawks playoffs in NBA Finals.
20.	6 June 2016	Kylie Jenner's Twitter account is HACKED.

The definition of recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. In online streaming and event detection studies, the relevant instances correspond to the all event occurred all over the world (or for a given region of the world) for the time of interest. However, it is very hard and infeasible to create a set of relevant instances which provides the requirements of its definition. Therefore, we have created a ground set which is described in detail in Section 5.4.1 and use this ground set as the all relevant instances. In consequence, the definition of recall shows minor differences in this study and it 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, F-measure is calculated conventionally, as the harmonic mean of precision and recall.

$$Recall = \frac{Total\ number\ of\ events\ detected\ by\ method}{Total\ number\ of\ events\ in\ ground\ truth} \tag{5.7}$$

$$F\text{-}measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.8}$$

Table 5.5: Accuracy Results for the Keyword-based Method

Country	**Bursty Keywords Found**	**Keywords Matching Some Event**	**Unde-tected Events**	**Precision**	**Recall**	**F-measure**
USA	220	135 (matching 14 events)	7	61%	67%	64%
CAN	17	7 (matching 2 events)	2	41%	50%	45%
Total	237	142 (matching 16 events)	9	60%	64%	62%

Table 5.6: Accuracy Results for the Clustering-based and the Hybrid Methods

	Country	Clusters Found	Event Clusters	Undetected Events	Precision	Recall	F-measure
Clustering	USA	74	39	0	53%	100%	69%
	CAN	7	5	0	71%	100%	83%
	Total	81	44	0	54%	100%	70%
Hybrid	USA	6	6	15	100%	29%	45%
	CAN	0	0	4	0	0%	0%
	Total	6	6	19	100%	24%	39%

Table 5.7: Details of Silhouette Coefficient (SC) Values for the Clustering-based and the Hybrid Methods

Method	Country	Number 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	6	1.0	1.0	1.0	0.0
Hybrid	CAN	0	NaN	NaN	NaN	NaN

Precision, Recall, and F-measure values for the methods are shown in Table 5.5 and Table 5.6. 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 5.7, as well. The clusters generated by the hybrid method for the USA tweets are all event clusters with high silhouette coefficient 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. On the overall, the clustering-based method gives the highest f-measure score.

5.4.3 Comparison of Events Detected by Keyword-based and Clustering-based Methods

Some statistics about the events found by keyword-based and clustering-based event detection methods are given under this title. While keyword-based method labels single words as events, clustering-based method labels a cluster vector containing many words coming from many tweets as events. Therefore, in this section we have found the intersection ratios for both methods.

For the intersection ratio of keyword-based method, we have checked how many of the words detected as events are included in any of the event clusters detected by clustering-based method.

- For Canada, there are 17 keywords labeled as events with 14 unique keywords[1] and 5 of them occurred in event cluster vectors of Canada. i.e, 0.36 of the event keywords intersect cluster vectors.

- For USA, there are 220 keywords labeled as events with 155 unique keywords[2] and 56 of them occurred in event cluster vectors of USA. i.e, 0.36 of the event keywords intersect cluster vectors.

For the intersection ratio of clustering-based method, we have checked how many of the cluster vectors detected as events contains any of event keywords detected by keyword-based method.

- For Canada, there are 7 cluster vectors labeled as events and 3 of them contain event keywords detected for Canada. i.e, 0.43 of the event clusters intersect event keywords.

- For USA, there are 74 cluster vectors labeled as events and 60 of them contain event keywords detected for USA. i.e, 0.81 of the event clusters intersect event keywords.

[1] Since same event keywords can occur for different rounds, there may be duplicated event entries.
[2] See footnote 1.

Table 5.8: Comparison of Keyword-based and Clustering-based Event Detection

Method	**Total number of Events Found**		**Number of Events Intersected With Other Method**		**Percentage of Intersection**	
	USA	CAN	USA	CAN	USA	CAN
Keyword-based	220 (155 unique)	17 (14 unique)	56	5	0.36	0.36
Clustering-based	74	7	60	3	0.81	0.43

Table 5.9: Ranking of Events Detected By Clustering-based Method using Intersection Ratios CAN (Ground truth set can be found in Table 5.4)

Number of Intersection	**Map Representation of Events Detected**	**Ground Truth**
3	{**adam**, ass, dirty, **draymond**, get, **green**, hurt, pull, steven, try}	2
2	{boxer, great, greatest, legend, **muhammad**, peace, rest, **rip**, time, world}	15
2	{bee, butterfly, float, greatest, legend, muhamm, **muhammad**, **rip**, sting, time}	15
0	{final, nba, take, win }	6
0	{fdb, music, official, ringtone, video, ztro }	8
0	{follow, please, since }	non-event
0	{guitar, lie, many }	non-event

In Table 5.9 and 5.10, we show the events detected by clustering method by map representation of high weighted entries with number of intersected event keywords detected by keyword-based method, e.g for the first row, 10 different event keywords detected by keyword-based method occurred in the event cluster vector detected by clustering-based method. The keywords used in the cluster representation are deter-

mined by using the highest tenth weight in the cluster as threshold. Words having a weight greater than or equal to the threshold or words which occur in both methods regardless of their weights are included in the cluster representation. Additionally, the bursty keywords contained in cluster are shown in bold in tables. These table show that clusters having higher intersections are more meaningful and points out a real-life event for both countries. Especially clusters having intersections more than 3 are showing real-life events for USA. Therefore, we can say that if a large number of events are detected for a country, it could be handful to use the intersection number to find out the real-life events.

Table 5.10: Ranking of Events Detected By Clustering-based Method using Intersection Ratios USA (Ground truth set can be found in Table 5.4)

Number of Intersection	**Map Representation of Events Detected**	**Ground Truth**
10	{**ali**, **alus**, **butterfly**, champ, easy, **greatest**, history, **legend**, **muhammad**, **peace**, **rest**, **rip**, **step**, time}	15
9	{**ali**, **alus**, boxer, **greatest**, **legend**, **muhammad**, **peace**, **rest**, **rip**, **step**, time}	15
9	{**ali**, **alus**, **butterfly**, **greatest**, **legend**, muhamm, **muhammad**, **peace**, **rest**, **rip**, time}	15
9	{**ali**, **alus**, **butterfly**, champ, easy, great, **greatest**, **legend**, **muhammad**, **peace**, **rest**, **rip**}	15
8	{**ali**, bee, **butterfly**, **greatest**, **legend**, **muhammad**, **peace**, **rest**, **rip**, sting, time}	15
7	{**ali**, **alus**, big, champ, easy, fellow, **greatest**, **legend**, **muhammad**, **peace**, piece, **rest**}	15
7	{bee, **butterfly**, eye, float, fly, **greatest**, hit, **legend**, **muhammad**, **peace**, **rest**, **rip**, see, sting}	15
7	{**adam**, **ankle**, cook, **curry**, **dirty**, get, **green**, guard, keep, man, spray, **steven**, **switch**}	3
6	{bee, **butterfly**, champ, float, **greatest**, **muhammad**, **peace**, **rest**, **rip**, sting}	15

Table 5.10 continued

6	{eye, float, **greatest**, hand, hit, **legend**, **muhammad**, **peace**, **rest**, **rip**, see, sting, time}	15
6	{**butterfly**, champion, float, flow, forever, **greatest**, hand, important, inspiring, **muhammad**, **peace**, person, **rest**, **rip**, sting}	15
6	{act, **adam**, **dirty**, **draymond**, **foul**, get, **green**, player, pull, **steven**, try}	2
6	{champ, float, fly, **greatest**, **legend**, **muhammad**, **peace**, **rest**, **rip**, rumble, sting, time}	15
6	{eye, float, fly, **greatest**, hand, hit, **legend**, **muhammad**, **peace**, **rest**, **rip**, see, sting}	15
6	{**ankle**, break, **curry**, **dirty**, fall, get, **jab**, make, **slip**, **step**, thompson}	10
5	{back, bron, call, finally, **foul**, get, **jame**, lebron, **nba**, **ref**, time, travel, **travels**}	18
5	{blues, **bonino**, **dion**, effort, game, get, guess, nigga, **pen**, penguin, shark, take, **waiter**, wednesday, **win**}	4
5	{ball, **delly**, dick, **foul**, get, hit, **iggy**, **iguodala**, man, nut, play, **shumpert**, wrong}	12
5	{apply, **click**, **hospitality**, join, **latest**, **opening**, **retail**, sale, see, team, view}	non-event
4	{arrogance, arrogant, **bench**, cav, **curry**, get, **livingston**, lose, respect, torch, warrior, **win**}	13
4	{**curry**, final, freak, get, **livingston**, miss, **mvp**, nigga, play, **shaun**, well}	14
4	{**anthem**, banner, get, good, **john**, **legend**, **national**, okay, sing, sound, spangled, star}	9
4	{**bench**, cav, **curry**, **double**, game, get, lowest, next, score, season, take, warrior, **win**}	13

Table 5.10 continued

4	{anymore, float, fly, **greatest**, **muhammad**, power, **rest**, **rip**, rumble, sky, sting, young}	15
3	{**cavalier**, final, **nba**, take, **win**}	6
3	{**anthem**, carlo, chile, guitar, **national**, play, **santana**, sing, uruguay, wrong}	17
3	{ass, ball, bang, block, call, contact, feel, **foul**, game, get, **kevin**, love, man, murder, play, refuse, **soft**, trip}	11
3	{beat, **bench**, cav, even, game, get, lose, score, starter, **step**, warrior, **win**}	13
3	{back, beat, cav, come, **curry**, get, gon, **nba**, series, thunder, warrior, **win**}	6
3	{beauty, join, **latest**, open, **opening**, read, **retail**, see, team, ulta, view}	non-event
3	{**cavalier**, final, **nba**, take, **win**}	6
3	{care, join, **latest**, manager, open, **opening**, read, **retail**, see, team, view}	non-event
3	{analyst, compliance, join, **latest**, **opening**, read, register, **retail**, see, team, view}	non-event
3	{corporation, join, **latest**, open, **opening**, read, **retail**, sale, see, team, view}	non-event
3	{join, **latest**, new, open, **opening**, read, see, team, **technical**, view}	non-event
3	{join, **latest**, open, **opening**, read, **retail**, sale, see, shift, supervisor, team, temporary, view}	non-event
3	{half, join, **latest**, open, **opening**, read, **retail**, robert, see, supervisor, team, technology, view}	none-event
3	{apply, **click**, join, **latest**, manager, open, **opening**, sale, see, team}	non-event

Table 5.10 continued

3	{automation, cater, coordinator, engineer, join, **latest**, management, open, **opening**, principal, read, **retail**, see, system, team, view}	non-event
3	{associate, join, **latest**, open, **opening**, read, **retail**, sale, see, team, view}	non-event
2	{cna, georgia, join, **latest**, open, **opening**, read, see, team, view}	non-event
2	{cloud, company, east, energy, harvey, join, **latest**, open, **opening**, peak, read, see, speedway, supply, team, tractor, view}	non-event
2	{cognizant, join, **latest**, open, **opening**, read, resourcemfg, see, team, view}	non-event
2	{engineer, join, **latest**, open, **opening**, read, see, supervisor, team, view}	non-event
2	{advisor, analytic, business, county, dunn, **fit**, great, interest, maintenance, might, near, research, **retail**, system}	non-event
2	{baby, cav, cleveland, fan, final, golden, **ref**, state, wait, **win**, word}	5
2	{big, cable, **curry**, example, fire, game, great, hate, heating, klay, **mvp**, prime, pull, respect, right, shoot, shooter, song, thompson, trouble, warrior}	1
2	{join, **latest**, med, open, **opening**, read, see, team, view, writer}	non-event
2	{join, **latest**, manager, open, **opening**, read, see, staffing, team, view}	non-event
2	{join, **latest**, manager, open, **opening**, read, see, store, team, view}	non-event
2	{aerotek, general, join, **latest**, open, **opening**, read, see, team, view}	non-event

Table 5.10 continued

2	{analyst, join, **latest**, open, **opening**, read, see, senior, team, view}	non-event
2	{california, join, **latest**, open, **opening**, read, see, service, team, view}	non-event
2	{dell, factory, infant, join, keyholder, **latest**, open, **opening**, part, read, register, see, team, time, view, wisconsin, woman}	non-event
1	{corporate, driver, **fit**, great, interest, manager, might, near, sale, service}	non-event
1	{**lyft**, code, free, lyftontwitter, promo, ride, try, use}	non-event
1	{clinical, driver, **fit**, great, interest, manager, might, near, register, service}	non-event
1	{con, **vamo**, nalgita}	non-event
1	{clearly, feel, get, **hack**, idol, kylie, laugh, morning, talk, twitter}	20
1	{application, associate, **fit**, great, interest, manager, might, near, sale, travel}	non-event
0	{boy, cav, cleveland, get, gon, heat, lebron, series, sweep, together }	19
0	{chain, cross, jesus, necklace, pendant, religious, steel }	non-event
0	{comment, jdameanor, love, post, see, share, show, sorry, support, youtube }	non-event
0	{music, official, video }	8
0	{fdb, music, official, ringtone, video, ztro }	8
0	{guest, movie, please,see, trailer}	7
0	{awesome, back, brock, call, fight, happy, imagine, kid, lesnar, lesner, month, next, tho, ufc }	16

Table 5.10 continued

0	{check, please, song }	non-event
0	{birthday, day, friend, happy, hope, love, miss, much, pretty, wait }	non-event
0	{bundle, buy, free, get, huge, low, sale }	non-event
0	{fdb, music, official, ringtone, video, ztro }	8
0	{fdb, music, official, ringtone, video, ztro }	8
0	{comment, jdameanor, love, post, see, share, show, sorry, support, youtube }	non-event
0	{comment, jdameanor, love, post, see, share, show, sorry, support, youtube }	non-event

5.4.4 Comparison of Events Detected by Keyword-based and Hybrid Methods

Some statistics about the events found by keyword-based and hybrid event detection methods are given under this title. While keyword-based method labels single words as events, hybrid method labels a cluster vector containing many words coming from many tweets as events. Therefore, in this section we have found the intersection ratios for both methods.

For the intersection ratio of keyword-based method, we have checked how many of the words detected as events are included in any of the event clusters detected by hybrid method.

- For Canada, there is no event cluster found by hybrid methodology. Therefore, the intersection ratio is 0.
- For USA, there are 220 keywords labeled as events with 155 unique keywords[3] and 24 of them occurred in event cluster vectors of USA. i.e, 0.16 of the event keywords intersect cluster vectors.

[3] Since same event keywords can occur for different rounds, there may be duplicated event entries.

For the intersection ratio of hybrid method, we have checked how many of the event cluster vectors contains any of event keywords detected by keyword-based method.

- For Canada, there is no event cluster found by hybrid methodology. Therefore, the intersection ratio is 0.
- For USA, there are 6 cluster vectors labeled as events and all of them contain event keywords detected for USA. i.e, 1.0 of the event clusters intersect event keywords.

In Table 5.12, the number of intersections are shown in the same way applied in Table 5.10. For the hybrid methodology, the intersection number of hybrid and keyword-based methods are not enlightening since the number of event clusters are small and all of them points out a real-life event. But from the table, we see that the minimum intersection number is 2 and this information can also be used if there is a large number of events detected by hybrid methodology for further researches.

Table 5.11: Comparison of Keyword-based and Hybrid Method of Event Detection

Method	**Total number of Events Found**		**Number of Events Intersected With**		**Percentage of Intersection**	
	USA	CAN	USA	CAN	USA	CAN
Keyword-based	220 (155 unique)	17 (14 unique)	24	0	0.16	-
Hybrid	6	0	6	0	1.0	-

Table 5.12: Ranking of Events Detected By Hybrid Method using Intersection Ratios for USA (Ground truth set can be found in Table 5.4)

Number of Intersection	**Map Representation of Events Detected**	**Ground Truth**
7	{**adam**, **dirty**, doin, **draymond**, **flop**, **foul**, **green**, pick, pull, series, stay, **steven**, ugly, wait}	2
5	{**ankle**, break, **curry**, fall, get, **jab**, make, **slip**, **step**, thompson, tristan}	10
5	{**ali**, dammit, goat, god, **greatest**, **legend**, man, muhamm, **muhammad**, **rip**}	15
3	{**cavalier**, final, **nba**, take, **win**}	6
3	{**anthem**, get, **john**, **legend**, light, nigga, sing, skin, take, tryna, voice}	9
2	{back, deck, exactly, get, help, hurt, **kevin**, love, rush, **soft**}	11

Table 5.13: Detected Words by Keyword-based and Clustering-based Methods

	Words detected by both methods	**Words detected only by keyword-based method**
CAN	draymond, adam, green, rip, muhammad	livingston, mtvpopcd, barbosa, agp, curry, bucciovertimechallenge, justshowupshow, 20:20, ali
USA	technical, opening, latest, fit, hospitality, click, retail, rest, butterfly, peace, rip, greatest, muhammad, bench, double, curry, win, step, legend, alus, ali, shaun, livingston, mvp, foul, iggy, iguodala, delly, shumpert, kevin, soft, lyft, dirty, slip, jab, ankle, anthem, national, john, cavalier, nba, hack, travels, ref, jame, vamo, santana, bonino, pen, dion, waiter, adam, green, steven, switch, draymond	3-1, movement, 11:11, cub, marquez, rafa, allin, nbafinalsvote, barbosa, denzel, bucciovertimechallenge, shump, clock, flop, kobe, drake, ddt, flagrant, marleau, rko, careerarc, jobs, sjsharks, elbow, gameofthrones, hound, 2-0, andy, varejao, diaz, mcgregor, karlie, speight, nbafinals, barne, harrison, diabetes, diabetic, type, 1diabete, crosby, sheary, lovejoy, ward, navybestwhip, braun, tie, mookie, greta, seager, spoon, guardado, jefferson, roberson, tommie, hip, hop, dellavedova, replay, backstreet, hbk, kessel, phil, anderson, ibaka, knee, layup, steph, kerr, sprint, verizon, rust, rusty, chad, retweet, lovebts, (word written in other than latin alphabet), westvirginia, hiring, job, recommend, nursing, sales, toronto, -lsb-, -rsb-, 06/05, ave, div, dagger, andnew, bisp, bisping, michael, murray, muhammadali, 2k17, goldberg, answer

Table 5.14: Detected Words by Keyword-based and Hybrid Methods

	Words detected by both methods	**Words detected only by keyword-based method**
CAN	-	-
USA	cavalier, nba, win, dirty, draymond, foul, adam, green, steven, flop, slip, jab, step, ankle, curry, anthem, legend, john, rip, ali, greatest, muhammad, kevin, soft	3-1, movement, 11:11, cub, marquez, rafa, travels, lyft, allin, nbafinalsvote, barbosa, denzel, bucciovertimechallenge, shump, shumpert, clock, kobe, drake, national, santana, livingston, shaun, ddt, double, flagrant, marleau, rko, careerarc, jobs, sjsharks, elbow, ref, gameofthrones, hound, 2-0, jame, andy, varejao, diaz, mcgregor, karlie, speight, nbafinals, barne, harrison, hack, diabetes, diabetic, type, 1diabete, bench, bonino, pen, dion, waiter, crosby, sheary, lovejoy, ward, delly, navybestwhip, braun, tie, mookie, greta, seager, spoon, guardado, jefferson, vamo, mvp, roberson, tommie, hip, hop, dellavedova, iguodala, replay, technical, backstreet, hbk, kessel, phil, anderson, ibaka, knee, layup, steph, kerr, sprint, verizon, switch, rust, rusty, chad, retweet, lovebts, (word written in other than latin alphabet), westvirginia, hiring, job, retail, click, fit, latest, opening, recommend, hospitality, nursing, sales, toronto, -lsb-, -rsb-, 06/05, ave, div, dagger, andnew, bisp, bisping, michael, murray, muhammadali, alus, butterfly, peace, rest, 2k17, goldberg, iggy, answer

Discussion. There are two reasons why some of the words does not appear in any of the clusters:

- Tweets, containing same event keywords detected by keyword-based method, are splitted into multiple clusters and this may result in clusters with small volumes and these clusters may be eliminated by thresholds. For example, the word '3-1' has the count 74 whereas 'curry' has the count 654 when they become an event in keyword-based event detection method, therefore if tweets containing word '3-1' are splitted into multiple clusters and the number of tweets forming the clusters are less than the threshold, these clusters may be removed. On the other hand the count of 'curry' is much more than the count of '3-1' and even if tweets containing this word is splitted into multiple clusters there is a high possibility for some of the clusters to become an event.
- Cluster containing an event keyword, detected by keyword-based method, become an event cluster but word may be eliminated since its weight is below threshold. For example, for USA people mostly use 'muhammad' (having count 478) and 'ali' (having count 534) separately in their posts and this causes keyword 'muhammadali' (having count 107) to have low weight and removed from clusters.

This is an expected behavior of clustering-based method since it has a stronger elimination conditions and does not allow weak keywords to pass. Therefore, we can say that clustering-based method is more accurate than keyword-based method. On the other hand, it can be seen from tables that hybrid has the strongest filtering algorithm and eliminates a big majority of words, but it also is an expected behavior because there is two elimination steps; one by keyword-based algorithm elimination and then clustering-based elimination. However this causes some events to be missed and this is an disadvantage.

5.4.5 Comparison of Events Detected by Clustering and Hybrid Method

For the comparison of these two methods, we calculate the intersection ratio of clusters. For example, cluster A has 10 words in map and B has 5 words and 2 of them

are common, then the maximum intersection rate is 0.4 for these two clusters. Additionally, a threshold value is needed to say if those two clusters point to the same event or not. Therefore, we use multiple threshold values for this purpose and results are given in Table 5.15.

Table 5.15: Comparison of clustering and hybrid techniques

Threshold	**C_1**	**C_2**	**C_3**	**C_4**	**C_5**	**C_6**	**Percentage**
0.1	✓	✓	✓	✓	✓		0.83
0.2	✓	✓	✓	✓	✓		0.83
0.3	✓	✓	✓	✓	✓		0.83
0.4	✓	✓	✓	✓	✓		0.83
0.5		✓	✓	✓	✓		0.66
0.6		✓		✓			0.33
0.7		✓		✓			0.33

Common words of clusters are given below:

- C_1 : [adam, dirty, doin, draymond, green, series, stay, ugly]
- C_2 : [cavalier, final, lock, nba, take, warrior, win]
- C_3 : [john, legend, skin, take, tryna]
- C_4 : [ankle, break, fall, jab, step, thompson]
- C_5 : [ali, dammit, god, muhammad, rip]
- C_6 : [back, deck, exactly, help, kevin, love, rush]

Table 5.15 contains the information below:

- Threshold: Cluster A detected by clustering-based method and Cluster B detected by hybrid method are similar if intersection percentage of words is above the threshold value. For example Cluster A has 10 words and 5 of them are also contained in Cluster B, then intersection percentage is 0.5 and if threshold is less or equal to 0.5 then these clusters are assumed to point the same event.

- Cluster 1 - 6: Event clusters detected by hybrid methodology and Yes/No shows even if there is an intersected event cluster detected by clustering-based method or not.

- ✓: Shows even the similar event cluster occurs in the event clusters obtained using clustering methodology.

- Percentage: Similarity rate between clusters of hybrid and clustering approaches. i.e, 5/6 = 0.83 means that 5 of the event clusters created by hybrid methodology are similar to some of the event clusters formed by clustering approach

The above table shows that Cluster 2 and 4 have similar clusters in clustering method for each threshold values. Those two clusters are related to NBA final game and both approaches label event clusters related to this event.

Cluster 1, 2, 3, 4 and 6 are related to NBA final game also. Specifically 1 is related to Draymond Green at NBA finals, 2 is about the game in general Warriors vs Cavaliers, 3 is related to John Legend National Anthem at NBA finals, 4 is related to Stephen curry with the ankle breaker jab step on Tristan Thompson, 5 is related to the death of Muhammad Ali (RIP) and 6 is related to Kevin Love's struggles in NBA Finals. Clustering-based method also marks some clusters as events for each of the above event clusters detected by hybrid method.

Table 5.16: Performance Results for All Methods

Method	**Total Execution Time of 7 days (hours)**	**Number of Tweets Processed per Second**	**Number of Rounds Processed per Minute**	**Round Processing Time (seconds)**
Keyword	~ 3	1200	9.3	6.5
Clustering	~ 11	300	2.5	24
Hybrid	~ 3.5	950	8	7.5

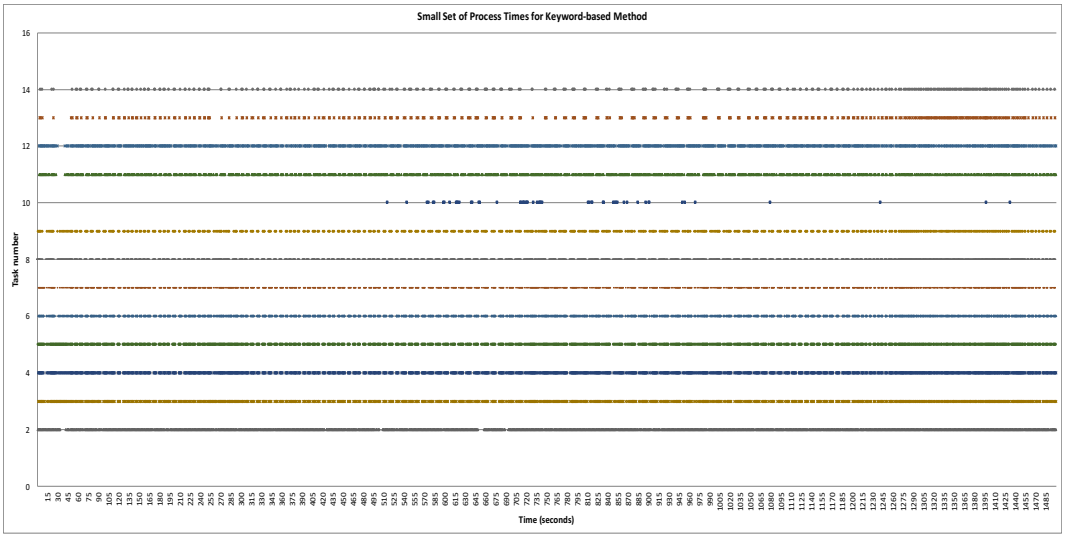


Figure 5.1: Processing time of keyword-based method

5.4.6 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 total processing time of the complete input dataset through the corresponding Storm topology for each method. Table 5.16 summarizes our results. As expected, the most efficient method is the keyword-based event detection method with the total execution time of about 3 hours, since this method does not involve as many iterations over the data as the other methods and it does not work with large blocks of data. 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 highest total execution time of about 11 hours. The hybrid method shows a major improvement over the clustering-based method, bringing the total execution time down to about 3.5 hours. This proves that filtering tweets based on bursty keywords can be effective in reducing the cost of cluster computation. In this section, small subsets of processing times are added to discuss the execution times of tasks.

As seen in Figure 5.1, the task numbers and the corresponding tasks are as follows:

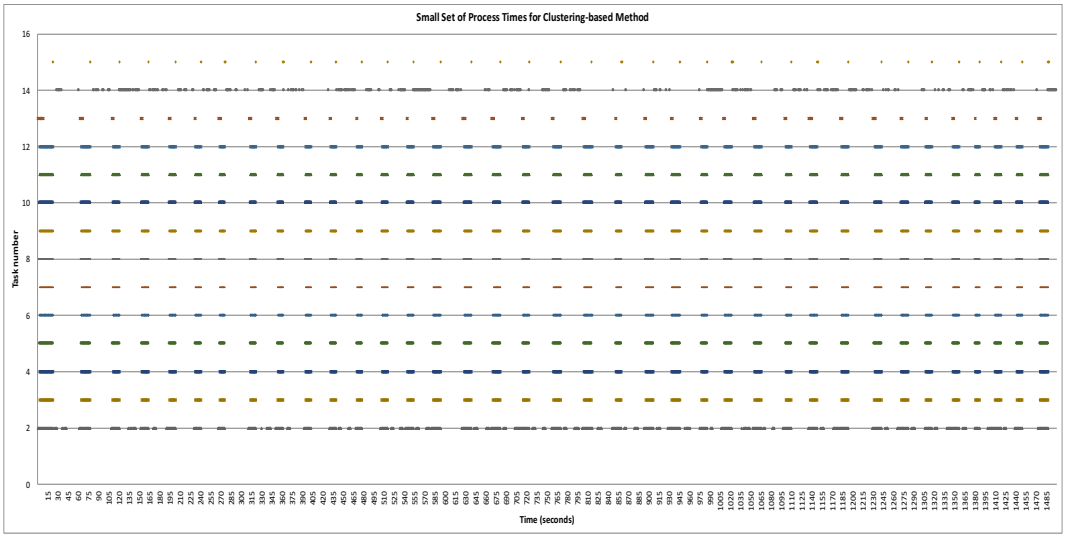


Figure 5.2: Processing time of clustering-based method

Task 2: input stream

Task 3, 4, 5, 6, 7: word count bolts for USA

Task 8, 9: word count bolts for CAN

Task 10: compare bolt

Task 11, 12: event detector bolts for USA

Task 13, 14: event detector bolts for CAN

As seen from Figure 5.1, input stream is nearly always active and emitting data to word count bolts. Word count bolts are also very active with some stop sections where event detector bolts takes action and detect keywords as events. While event detector bolts of USA have a little idle time, event detector bolts of CAN have more idle time since the data volume of CAN is much smaller than USA. The compare bolt has the longest idle time since it gets active only when there is a new event detection.

As seen in Figure 5.2, the task numbers and the corresponding tasks are as follows: tasks are as follows:

Task 2: input stream

Task 3, 4, 5, 6, 7, 8, 9, 10, 11, 12: clustering bolts for USA

Task 13: clustering bolt for CAN

Task 14: event detector bolt for USA

Task 15: event detector bolt for CAN

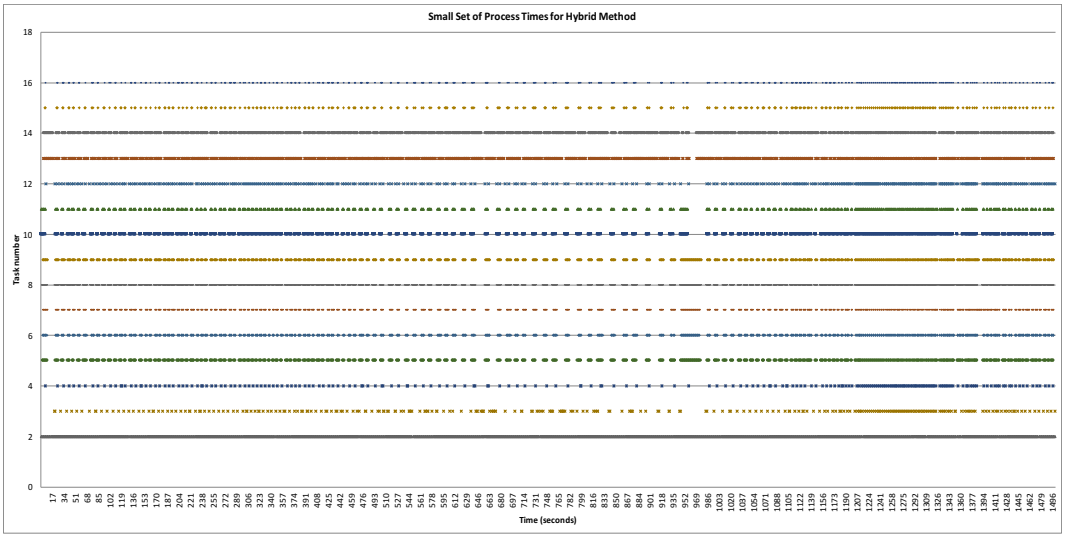


Figure 5.3: Processing time of hybrid method

As seen from Figure 5.2, the input stream is slower and waits more than keyword-based methods, because the complexity of this method is higher than keyword-based method and the process times are longer for clustering tasks. Therefore, input stream waits for all tasks to finish their work before streaming the next round. The active and idle times of event detector tasks and clustering tasks complete each other in this method, because event detector tasks wait for clustering task to complete its local clustering job to start the global clustering. And also bolts of CAN is less active than bolts of USA as expected.

As seen in Figure 5.3, the task numbers and the corresponding tasks are as follows:

Task 2: input stream
Task 3: clustering bolt for USA
Task 4: clustering bolt for CAN
Task 5, 6, 7, 8, 9: word count bolts for USA
Task 10, 11: word count bolts for CAN
Task 12: event detector bolt for both countries
Task 13, 14: keybased event detector bolts for USA
Task 15, 16: keybased event detector bolts for CAN

As seen from Figure 5.3, input stream is also nearly always active in this method since splits tweets into words and emit words one by one. Keybased event detector

bolt takes action when a common word is reached to itself and evaluates words to detect keybased events. Clustering bolt is activated when keybased event detector bolt emits the list of event keywords so that it can filter the tweet list and create local clusters. Event detector bolt waits for clustering bolts as expected since this bolt is responsible of global clustering and should wait for local clusters. Similar to other methods, tasks of CAN is less active than tasks of USA.

5.5 Discussion

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

- The clustering-based method provides the highest recall and f-measure values per country as well as 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. 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.
- 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. The fragmentation problem in the clustering-based method could also be improved through using semantic similarity measurement methods, however, this likely would incur additional processing cost.
- On the other hand, there might be an advantage to generating several event clusters for the same event in that, the event itself or its effect would then last for several rounds. This could be useful in detecting events that may have different durations.

- While tweet filtering applied in the hybrid method brings efficiency, it also causes a drop in recall and f-measure for both the USA and the Canada events. This drop is in fact quite drastic for Canada, such that it was not possible to generate any clusters (hence, the 0's in Table 5.6 and the NaN's in Table 5.7). It may be possible to fix this problem with a more detailed analysis of our parameter settings. Despite this, the clusters that were successfully generated strongly indicate the occurrence of relevant events from the ground truth set.

- 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.

- 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 transactional processing to efficiently coordinate concurrent data accesses, which we plan to investigate in more depth as part of our future work.

CHAPTER 6

CONCLUSION

Online event detection aims to discover the events in real time or near real time through analysis of streaming web content such as blog posts or social media messages. 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 *stream processing framework* Apache Storm.

The methods are evaluated on a set of tweets collected in one week. The experiments are conducted on stream simulation, so that the methods can be compared on the same data. The tweet streams are processed in time windows, called *rounds*. In each round, the tweets are processed through the defined topologies and the change in the output of each round determines whether an event is detected or not.

The experimental results show the applicability of the stream processing frameworks for online event detection. Among the compared methods, clustering-based one provides higher f-measure and recall scores. On the other hand, keyword-based method is a more lightweight solution in terms of topology structure and processing time. However, it provides lower recall and captures less information about the event, relying on a single keyword. Hybrid method, as expected, provides a better balance between accuracy and processing time cost. However, the recall values are much lower than that of the other two methods.

There are several directions for improvement over this work. As one of the improvements, parameter settings for clustering-based method can be further analyzed. By this way, the recall value for the hybrid method can be increased to a more satisfac-

tory level. In clustering operations such as cluster update and merge, we used cosine similarity on term vectors. Semantic similarity based measurements can be utilized to prevent fragmentation of clusters related to the same event. As another research direction, utilization of transactional support within stream processing can be investigated and its effect on clustering accuracy can be analyzed.

REFERENCES

[1] J. L. Dario Simonassi, Gabriel Eisbruch, *Getting Started with Storm*. O'Reilly Media, Inc., 2012.

[2] S. Aslam, "Twitter by the numbers: Stats, demographics & fun facts." `https://www.omnicoreagency.com/twitter-statistics/`. Accessed: 2018-03-20.

[3] M. Cordeiro and J. Gama, "Online Social Networks Event Detection: A Survey," in *Solving Large Scale Learning Tasks. Challenges and Algorithms* (S. Michaelis, N. Piatkowski, and M. Stolpe, eds.), vol. 9580 of *Lecture Notes in Computer Science*, pp. 1–41, Springer, Cham, 2016.

[4] M. F. Mokbel and A. Magdy, "Microblogs Data Management Systems: Querying, Analysis, and Visualization (Tutorial)," in *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 2219–2222, 2016.

[5] F. Atefeh and W. Khreich, "A Survey of Techniques for Event Detection in Twitter," *Computational Intelligence*, vol. 31, no. 1, pp. 132–164, 2015.

[6] V. Gulisano, Z. Jerzak, S. Voulgaris, and H. Ziekow, "The DEBS 2016 Grand Challenge," in *ACM International Conference on Distributed and Event-based Systems (DEBS)*, pp. 289–292, 2016.

[7] J. Allan, *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers, 2002.

[8] E. Wu, Y. Diao, and S. Rizvi, "High-performance Complex Event Processing over Streams," in *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 407–418, 2006.

[9] M. Akdere, U. Cetintemel, and N. Tatbul, "Plan-based Complex Event Detection Across Distributed Sources," in *International Conference on Very Large Data Bases (VLDB)*, pp. 66–77, 2008.

[10] M. Li, M. Mani, E. A. Rundensteiner, and T. Lin, "Complex Event Pattern Detection over Streams with Interval-based Temporal Semantics," in *ACM International Conference on Distributed Event-based Systems (DEBS)*, pp. 291–302, 2011.

[11] T. Sakaki, M. Okazaki, and Y. Matsuo, "Tweet Analysis for Real-Time Event Detection and Earthquake Reporting System Development," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 25, no. 4, pp. 919–931, 2013.

[12] K. Watanabe, M. Ochi, M. Okabe, and R. Onai, "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.

[13] H. Abdelhaq, C. Sengstock, and M. Gertz, "EvenTweet: Online Localized Event Detection from Twitter," *PVLDB*, vol. 6, no. 12, 2013.

[14] M. Hasan, M. A. Orgun, and R. Schwitter, "TwitterNews+: A Framework for Real Time Event Detection from the Twitter Data Stream," in *Social Informatics* (E. Spiro and Y.-Y. Ahn, eds.), vol. 10046 of *Lecture Notes in Computer Science*, pp. 224–239, Springer, Cham, 2016.

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

[16] Y. Wang, R. Xu, B. Liu, L. Gui, and B. Tang, "A Storm-Based Real-Time Micro-Blogging Burst Event Detection System," in *Machine Learning and Cybernetics* (X. Wang, W. Pedrycz, P. Chan, and Q. He, eds.), vol. 481 of *Communications in Computer and Information Science*, pp. 186–195, Springer, 2014.

[17] "IEEE Data Engineering Bulletin, Special Issue on Next-Generation Stream Processing," 2015.

[18] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pp. 56–65, ACM, 2007.

[19] S. Milstein, A. Chowdhury, G. Hochmuth, B. Lorica, and R. Magoulas, "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.

[20] H. Sayyadi, M. Hurst, and A. Maykov, "Event Detection and Tracking in Social Streams," in *International Conference on Web and Social Media (ICWSM)*, pp. 311–314, 2009.

[21] M. Hasan, M. A. Orgun, and R. Schwitter, "A survey on real-time event detection from the twitter data stream," *Journal of Information Science*, vol. 0, no. 0, p. 0165551517698564, 0.

[22] C. Li, A. Sun, and D. A. Twevent:, "segment-based event detection from tweets," in *Proceedings of the ACM international conference on information and knowledge management (CIKM '12), Maui, HI, 29 October–2 . : ACM*, pp. 155–164, November 2012.

[23] A. Marcus, M. S. Bernstein, O. Badar, *et al.*, "Twitinfo: aggregating and visualizing microblogs for event exploration," in *Proceedings of the SIGCHI conference on human factors in computing systems (CHI'11)*, (New York), pp. 227–236, ACM, 2011.

[24] M. Mathioudakis and K. N. TwitterMonitor:, "trend detection over the twitter stream," in *10) . New York: ACM*, pp. 1155–1158, Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD, 2010.

[25] F. Alvanaki, M. Sebastian, K. Ramamritham, *et al.*, "Enblogue: emergent topic detection in web 2.0 streams," in *Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD '11), Athens, pp. 1271–1274. : ACM*, pp. 12–16, June 2011.

[26] S. Gaglio, G. L. Re, and M. A. Morana, "framework for real-time twitter data analysis," *Comput Commun*, vol. 73, pp. 236–242, 2016.

[27] M. Cataldi, "Di caro l and schifanella c. emerging topic detection on twitter based on temporal and social terms evaluation," in *Proceedings of the tenth in-*

ternational workshop on multimedia data mining (MDMKDD '10), Washington, DC, 25 –4-10. : ACM, pp. 4–1, 2010.

[28] G. Stilo and P. Velardi, "Efficient temporal mining of micro-blog texts and its application to event discovery," *Data Min Knowl Disc*, vol. 30, pp. 372–402, 2016.

[29] R. Parikh and K. K. E. events from tweets, "In: Proceedings of the 22nd international conference on world wideweb (www'13 companion)," pp. pp. 613–620, 2013.

[30] J. Weng and L. Bs., "Event detection in twitter," in *Proceedings of the international AAAI conference on web and social media (ICWSM)*, pp. 401–408, vol. 11, 2011.

[31] X. Zhang, X. Chen, Y. Chen, *et al.*, "Event detection and popularity prediction in microblogging," *Neurocomput*, vol. 149, pp. 1469–1480, 2015.

[32] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang, "TopicSketch: Real-Time Bursty Topic Detection from Twitter," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 28, no. 8, pp. 2216–2229, 2016.

[33] S. Petrovic, M. Osborne, and V. Lavrenko, "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.

[34] M. Osborne, S. Moran, R. McCreadie, *et al.*, "Real-time detection tracking. monitoring of automatically discovered events in social media," in *Proceedings of the 52nd annual meeting of the association for computational linguistics*, Association for Computational Linguistics, 2014.

[35] M. Osborne, S. Petrovic, R. McCreadie, C. Macdonald, and I. Ounis, "Bieber no more: First Story Detection using Twitter and Wikipedia," in *SIGIR Workshop on Time-aware Information Access (TAIA)*, 2012.

[36] H. Becker, M. Naaman, and L. Gravano, "Beyond Trending Topics: Real-World Event Identification on Twitter," in *International AAAI Conference on Weblogs and Social Media (ICWSM)*, pp. 438–441, 2011.

[37] M. Mathioudakis and N. Koudas, "TwitterMonitor: Trend Detection over the Twitter Stream," in *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 1155–1158, 2010.

[38] C. De Boom, "Van canneyt s and dhoedt b," in *Semantics-driven event clustering in Twitter feeds*, pp. 2–9, In: Proceedings of the 5th workshop on making sense of microposts, vol. 1395 , CEUR, 2015.

[39] S. Phuvipadawat and T. Murata, "Breaking news detection and tracking in twitter," in *Proceedings of the IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology. Toronto, Canada, vol. 3 of WI-IAT '10, 31 August–3 September, , DC: IEEE Computer Society*, pp. 120–123, 2010.

[40] A. J. McMinn and J. Jm., "Real-time entity-based event detection for twitter," in *Experimental IR meets multilinguality, multimodality, and interaction: 6th international conference of the CLEF association (CLEF'15)* (J. Mothe, J. Savoy, J. Kamps, and others Kamps J. al, eds.), pp. 65–77, Berlin: Springer, 2015.

[41] S. Unankard, X. Li, and S. Ma., "Emerging event detection in social networks with location sensitivity," *World Wide Web*, vol. 18, no. 5, pp. 1393–1417, 2015.

[42] S. B. Kaleel and A. Abhari, "Cluster-discovery of twitter messages for event detection and trending," *J Comput Sci*, vol. 6, pp. 47–57, 2015.

[43] C. H. Lee and C. Tf., "Leveraging microblogging big data with a modified density-based clustering approach for event awareness and topic ranking," *J Inf Sci*, vol. 39, pp. 523–543, 2013.

[44] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, "Twitterstand: news in tweets," in *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems*, pp. 42–51, ACM, 2009.

[45] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors," in *International Conference on World Wide Web (WWW)*, pp. 851–860, 2010.

[46] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang, "TEDAS: A Twitter-based Event Detection and Analysis System," in *IEEE International Conference on Data Engineering (ICDE)*, pp. 1273–1276, 2012.

[47] H. Park, S.-B. Youn, G. Y. Lee, and H. Ko, "Trendy episode detection at a very short time granularity for intelligent vod service: a case study of live baseball game," in *Proceedings of EuroITV*, 2011.

[48] S. Petrović, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 181–189, Association for Computational Linguistics, 2010.

[49] A. Das Sarma, A. Jain, and C. Yu, "Dynamic relationship and event discovery," in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 207–216, ACM, 2011.

[50] H. Sayyadi, M. Hurst, and A. Maykov, "Event detection and tracking in social streams.," in *ICWSM*, 2009.

[51] S. Huang, X. Wu, and A. Bolivar, "The effect of title term suggestion on e-commerce sites," in *Proceedings of the 10th ACM workshop on Web information and data management*, pp. 31–38, ACM, 2008.

[52] O. Ozdikis, P. Senkul, and H. Oguztuzun, "Semantic expansion of hashtags for enhanced event detection in twitter," in *Proceedings of the 1st International Workshop on Online Social Systems*, Citeseer, 2012.

[53] X. Zhou and L. Chen, "Event Detection over Twitter Social Media Streams," *The VLDB Journal*, vol. 23, no. 3, pp. 381–400, 2014.

[54] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang, "STREAMCUBE: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration Over the Twitter Stream," in *IEEE International Conference on Data Engineering (ICDE)*, pp. 1561–1572, 2015.

[55] M. G. V. K. Singh and R. Jain, "Situation Detection and Control using Spatio-temporal Analysis of Microblogs," in *International Conference on World Wide Web (WWW)*, 2010.

[56] H. Abdelhaq, C. Sengstock, and M. Gertz, "Eventweet: Online localized event detection from twitter," *PVLDB*, vol. 6, no. 12, p. 2013, 2013.

[57] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, "TwitterStand: News in Tweets," in *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, pp. 42–51, 2009.

[58] "Wordnet website." `https://wordnet.princeton.edu/`. Accessed: 2016-04-03.

[59] A. D. Sarma, A. Jain, and C. Yu, "Dynamic Relationship and Event Discovery," in *ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 207–216, 2011.

[60] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003.

[61] "Twitter for java website." `http://twitter4j.org/en/index.html`. Accessed: 2016-04-03.

[62] N. Marz, "Nathan marz website." `http://nathanmarz.com/about/`. Accessed: 2016-04-03.

[63] "Backtype website." `http://www.backtype.com/`. Accessed: 2016-04-03.

[64] N. Marz, "A storm is coming." `https://blog.twitter.com/2011/storm-coming-more-details-and-plans-release`. Accessed: 2016-04-03.

[65] A. Cassandra, "Apache cassandra." `http://cassandra.apache.org/`. Accessed: 2017-07-30.

[66] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.

APPENDIX A

EVENTS DETECTED BY CLUSTERING-BASED AND HYBRID METHODS

A.1 Events detected by clustering-based methods

Tables in this appendix include the following information:

- Date: The date when the event occurred.
- Country: The country where the event occurred.
- Number of Tweets: The number of tweets forming the cluster.
- Common Words: The most common words contained in cluster with their weights.
- Ground Truth: The index of the event given in Table 5.4.

Table A.1: Events found by clustering-based method

Date	**Country**	**Number of Tweets**	**Common Words**	**Ground Truth**
31 May 2016, Tuesday, 05:30 AM	USA	106	[apply:0.82, click:0.82, join:0.76, latest:0.94, open:0.41, opening:0.59, see:0.87, team:0.69]	Non-event

Table A.1 continued

31 May 2016, Tuesday, 09:42 AM	USA	82	[comment:1.0, jdameanor:1.0, love:1.0, post:1.0, see:1.0, share:1.0, show:1.0, sorry:1.0, support:1.0, youtube:1.0]	Non-event
31 May 2016, Tuesday, 10:48 AM	USA	151	[game:0.42, klay:1.0, thompson:0.93]	1
31 May 2016, Tuesday, 10:54 AM	USA	750	[adam:0.62, draymond:0.77, green:0.8, steven:0.27]	2
31 May 2016, Tuesday, 10:54 AM	CAN	108	[adam:0.98, draymond:0.92, foul:0.6, green:0.76, steven:0.48]	2
31 May 2016, Tuesday, 11:30 AM	USA	169	[adam:0.98, curry:0.87, get:0.45, guard:0.34, keep:0.21, steven:0.57]	3
31 May 2016, Tuesday, 11:54 AM	USA	111	[blues:0.22, game:0.95, get:0.57, nigga:0.25, pen:0.71, penguin:0.28, shark:0.26, take:0.58, win:0.6]	4
31 May 2016, Tuesday, 12:24 PM	USA	309	[back:0.35, beat:0.22, cav:0.81, come:0.37, get:0.22, gon:0.39, series:0.3, warrior:0.76, win:0.72]	5
31 May 2016, Tuesday, 12:24 PM	USA	175	[golden:1.0, state:1.0]	6

Table A.1 continued

31 May 2016, Tuesday, 09:48 PM	USA	74	[guest:0.98, movie:0.98, please:1.0, see:1.0, trailer:0.98]	7
31 May 2016, Tuesday, 10:00 PM	USA	138	[join:0.61, latest:1.0, open:0.42, opening:0.8, read:0.6, see:0.61, team:0.56, view:0.53]	Non-event
01 June 2016, Wednesday, 01:00 AM	USA	110	[join:0.22, latest:1.0, opening:0.9, read:0.64, see:0.22, team:0.24, view:0.46]	Non-event
01 June 2016, Wednesday, 02:30 AM	USA	108	[join:0.65, latest:1.0, open:0.4, opening:0.8, read:0.56, see:0.65, team:0.58, view:0.72]	Non-event
01 June 2016, Wednesday, 04:00 AM	USA	113	[analyst:0.33, join:0.43, latest:1.0, open:0.38, opening:0.71, read:0.5, see:0.43, team:0.45, view:0.71]	Non-event
01 June 2016, Wednesday, 04:42 AM	USA	106	[join:0.52, latest:1.0, open:0.26, opening:0.8, read:0.49, see:0.52, team:0.51, view:0.59]	Non-event (job advertisement)
01 June 2016, Wednesday, 06:00 AM	USA	105	[join:0.68, latest:1.0, open:0.46, opening:0.83, read:0.47, see:0.68, team:0.68, view:0.39]	Non-event (job advertisement)

Table A.1 continued

01 June 2016, Wednesday, 03:54 PM	USA	97	[fdb:1.0, music:1.0, official:1.0, ringtone:1.0, video:1.0, ztro:1.0]	8
01 June 2016, Wednesday, 11:24 PM	USA	145	[fit:0.99, great:1.0, interest:0.62, might:0.55, near:0.62]	Non-event (job advertisement)
01 June 2016, Wednesday, 11:24 PM	USA	249	[apply:1.0, click:1.0, join:0.29, latest:0.89, opening:0.44, read:0.23, retail:0.26, sale:0.27, see:0.63, team:0.36, view:0.25]	Non-event
02 June 2016, Thursday, 12:00 AM	USA	108	[cognizant:0.5, join:0.69, latest:1.0, open:0.32, opening:0.83, read:0.47, resourcemfg:0.5, see:0.69, team:0.7, view:0.41]	Non-event (job advertisement)
02 June 2016, Thursday, 03:06 AM	USA	112	[join:0.52, latest:1.0, open:0.43, opening:0.76, read:0.56, sale:0.27, see:0.52, shift:0.25, supervisor:0.25, support:0.21, team:0.53, temporary:0.25, view:0.61]	Non-event
02 June 2016, Thursday, 05:24 AM	USA	103	[join:0.51, latest:1.0, open:0.46, opening:0.8, read:0.44, see:0.51, team:0.54, view:0.56]	Non-event

Table A.1 continued

02 June 2016, Thursday, 05:30 AM	USA	126	[join:0.64, latest:1.0, open:0.4, opening:0.79, read:0.57, see:0.74, team:0.69, view:0.37]	Non-event
02 June 2016, Thursday, 10:06 AM	USA	67	[check:1.0, please:1.0, song:1.0]	Non-event
02 June 2016, Thursday, 11:24 AM	USA	76	[chain:1.0, cross:1.0, jesus:1.0, necklace:1.0, pendant:1.0, religious:1.0, steel:1.0]	Non-event (Necklace advertisement)
02 June 2016, Thursday, 12:12 PM	USA	64	[bundle:1.0, buy:1.0, free:1.0, get:1.0, huge:1.0, low:1.0, sale:1.0]	Non-event
02 June 2016, Thursday, 12:12 PM	USA	102	[fdb:1.0, music:1.0, official:1.0, ringtone:1.0, video:1.0, ztro:1.0]	8
02 June 2016, Thursday, 08:30 PM	USA	127	[cavalier:0.63, final:1.0, lock:0.37, nba:1.0, take:1.0, warrior:0.37, win:1.0]	6
02 June 2016, Thursday, 08:36 PM	USA	266	[cavalier:0.62, final:1.0, lock:0.37, nba:1.0, take:1.0, warrior:0.38, win:1.0]	6
02 June 2016, Thursday, 10:00 PM	USA	111	[join:0.52, latest:1.0, open:0.32, opening:0.78, read:0.61, see:0.53, team:0.58, view:0.59]	Non-event

Table A.1 continued

02 June 2016, Thursday, 10:54 PM	CAN	71	[cavalier:0.49, final:1.0, lock:0.46, nba:1.0, take:1.0, warrior:0.49, win:1.0]	6
03 June 2016, Friday, 12:00 AM	USA	104	[county:0.33, fit:1.0, great:1.0, interest:0.79, might:0.71, near:0.79]	Non-event
03 June 2016, Friday, 08:24 AM	CAN	99	[guitar:1.0, lie:1.0, many:1.0]	Non-event
03 June 2016, Friday, 08:42 AM	USA	101	[fdb:0.97, music:1.0, official:1.0, ringtone:0.97, video:1.0, ztro:0.97]	8
03 June 2016, Friday, 10:00 AM	USA	252	[anthem:0.21, good:0.29, john:0.99, legend:1.0, okay:0.21, sound:0.22]	9
03 June 2016, Friday, 10:18 AM	USA	130	[ankle:0.29, break:0.34, curry:0.31, fall:0.31, get:0.21, jab:0.98, make:0.23, slip:0.21, step:0.98, thompson:0.3]	10
03 June 2016, Friday, 10:48 AM	USA	111	[foul:0.37, get:0.51, kevin:0.9, love:1.0, soft:0.23]	11
03 June 2016, Friday, 11:54 AM	USA	242	[ball:0.54, delly:0.31, dick:0.41, get:0.43, hit:0.81, iggy:0.25, man:0.29, nut:0.6, play:0.32, wrong:0.25]	12

Table A.1 continued

03 June 2016, Friday, 12:06 PM	USA	200	[arrogance:0.33, arrogant:0.33, bench:0.89, cav:0.59, curry:0.36, get:0.33, lose:0.42, respect:0.33, take:0.32, torch:0.33, warrior:0.9]	13
03 June 2016, Friday, 12:06 PM	USA	204	[livingston:1.0, miss:0.24, shaun:0.95]	14
03 June 2016, Friday, 12:12 PM	USA	145	[beat:0.35, bench:0.9, cav:0.87, get:0.38, starter:0.26, warrior:0.8]	13
03 June 2016, Friday, 12:30 PM	USA	109	[cav:0.65, game:0.93, get:0.33, klay:0.24, lowest:0.26, next:0.4, score:0.27, season:0.27, take:0.25, warrior:0.58, win:0.95]	13
03 June 2016, Friday, 01:00 PM	USA	105	[birthday:0.99, day:0.21, happy:1.0, hope:0.23, love:0.38, much:0.22, pretty:0.24, wait:0.23]	Non-event
03 June 2016, Friday, 02:00 PM	CAN	83	[chapter:0.94, follow:1.0, please:1.0, since:1.0]	Non-event
04 June 2016, Saturday, 12:00 AM	USA	119	[join:0.36, latest:1.0, open:0.33, opening:0.77, read:0.46, see:0.36, team:0.36, view:0.74]	Non-event

Table A.1 continued

04 June 2016, Saturday, 12:00 PM	USA	93	[code:1.0, free:1.0, lyft:1.0, lyftontwitter:1.0, promo:1.0, ride:1.0, try:1.0, use:1.0]	Non-event (Lyft advertisement)
04 June 2016, Saturday, 01:18 PM	USA	375	[ali:0.96, muhammad:0.87, rest:0.25, rip:0.73]	15
04 June 2016, Saturday, 01:24 PM	USA	1283	[ali:0.84, greatest:0.27, muhammad:0.77, peace:0.23, rest:0.27, rip:0.69]	15
04 June 2016, Saturday, 01:24 PM	USA	174	[float:0.88, rip:0.24, sting:0.96]	15
04 June 2016, Saturday, 01:30 PM	USA	2655	[ali:0.82, greatest:0.31, muhammad:0.73, rip:0.71]	15
04 June 2016, Saturday, 01:30 PM	USA	304	[float:0.9, rip:0.29, sting:0.96]	15
04 June 2016, Saturday, 01:30 PM	USA	161	[ali:0.24, big:0.48, champ:0.4, easy:0.7, fellow:0.51, greatest:0.25, peace:0.88, piece:0.43, rest:1.0]	15
04 June 2016, Saturday, 01:30 PM	CAN	143	[greatest:0.24, muhammad:0.68, rip:0.8]	15

Table A.1 continued

04 June 2016, Saturday, 01:36 PM	USA	151	[ali:0.34, champ:0.47, easy:0.41, greatest:0.37, legend:0.29, muhammad:0.28, peace:0.85, rest:1.0]	15
04 June 2016, Saturday, 01:36 PM	USA	276	[float:0.94, muhammad:0.21, rip:0.28, sting:0.94]	15
04 June 2016, Saturday, 01:36 PM	CAN	121	[greatest:0.39, muhammad:0.79, rip:0.61]	15
04 June 2016, Saturday, 01:42 PM	USA	178	[float:0.91, rip:0.21, sting:0.97]	15
04 June 2016, Saturday, 01:42 PM	USA	128	[ali:0.28, alus:0.22, champ:0.36, easy:0.46, greatest:0.31, peace:0.91, rest:1.0]	15
04 June 2016, Saturday, 01:48 PM	USA	159	[bee:0.95, butterfly:0.97, float:0.9, sting:0.97]	15
04 June 2016, Saturday, 01:54 PM	USA	123	[butterfly:0.97, float:0.93, sting:0.99]	15
04 June 2016, Saturday, 02:00 PM	USA	108	[bee:0.95, butterfly:0.98, float:0.95, muhammad:0.26, rip:0.24, sting:0.98]	15

Table A.1 continued

04 June 2016, Saturday, 10:00 PM	USA	108	[fit:1.0, great:1.0, interest:0.57, might:0.69, near:0.57]	Non-event (Job advertisement)
05 June 2016, Sunday, 12:00 AM	USA	132	[join:0.5, latest:1.0, open:0.27, opening:0.78, read:0.55, see:0.5, team:0.5, view:0.58]	Non-event
05 June 2016, Sunday, 12:00 AM	USA	120	[fit:1.0, great:1.0, interest:0.53, might:0.7, near:0.53]	Non-event (Job advertisement)
05 June 2016, Sunday, 02:12 AM	USA	106	[join:0.69, latest:1.0, open:0.38, opening:0.8, read:0.59, see:0.69, team:0.73, view:0.64]	Non-event
05 June 2016, Sunday, 12:36 PM	USA	108	[back:0.5, brock:0.96, fight:0.46, lesnar:0.86, lesner:0.32, ufc:0.71]	16
05 June 2016, Sunday, 08:18 PM	USA	114	[comment:1.0, jdameanor:1.0, love:1.0, post:1.0, see:1.0, share:1.0, show:1.0, sorry:1.0, support:1.0, youtube:1.0]	Non-event

Table A.1 continued

06 June 2016, Monday, 12:00 AM	USA	106	[join:0.38, latest:1.0, open:0.66, opening:0.74, read:0.45, see:0.38, team:0.38, view:0.68]	Non-event
06 June 2016, Monday, 01:00 AM	USA	102	[engineer:0.25, join:0.52, latest:1.0, open:0.29, opening:0.84, read:0.59, see:0.52, team:0.59, view:0.77]	Non-event
06 June 2016, Monday, 03:36 AM	USA	110	[join:0.61, latest:1.0, open:0.26, opening:0.86, read:0.59, see:0.61, team:0.59, view:0.45]	Non-event
06 June 2016, Monday, 04:00 AM	USA	109	[join:0.44, latest:1.0, open:0.28, opening:0.87, read:0.55, see:0.44, team:0.54, view:0.59]	Non-event
06 June 2016, Monday, 05:30 AM	USA	117	[join:0.74, latest:1.0, manager:0.27, open:0.39, opening:0.77, read:0.69, see:0.74, store:0.25, team:0.74, view:0.49]	Non-event
06 June 2016, Monday, 09:00 AM	USA	109	[anthem:0.99, carlo:0.21, chile:0.26, guitar:0.33, national:0.95, play:0.43, santana:0.37, uruguay:0.36]	17
06 June 2016, Monday, 10:42 AM	USA	119	[con:1.0, nalgita:0.98, vamo:1.0]	Non-event

Table A.1 continued

06 June 2016, Monday, 10:48 AM	USA	230	[call:0.9, finally:0.24, lebron:0.68, travel:0.87, travels:0.5]	18
06 June 2016, Monday, 10:54 AM	USA	105	[cav:0.82, get:0.98, gon:0.31, lebron:0.53, sweep:0.63, together:0.21]	19
06 June 2016, Monday, 12:12 PM	USA	71	[comment:1.0, jdameanor:1.0, love:1.0, post:1.0, see:1.0, share:1.0, show:1.0, sorry:1.0, support:1.0, youtube:1.0]	Non-event
06 June 2016, Monday, 02:06 PM	CAN	104	[fdb:1.0, music:1.0, official:1.0, ringtone:1.0, video:1.0, ztro:1.0]	8
06 June 2016, Monday, 02:12 PM	USA	148	[get:0.66, hack:0.99, kylie:0.82, twitter:0.44]	20
06 June 2016, Monday, 02:36 PM	USA	81	[fdb:1.0, music:1.0, official:1.0, ringtone:1.0, video:1.0, ztro:1.0]	8
07 June 2016, Tuesday, 02:00 AM	USA	101	[join:0.83, latest:1.0, open:0.57, opening:0.74, read:0.5, see:0.83, team:0.75, view:0.42]	Non-event

A.2 Events detected by hybrid method

Table A.2: Events found by hybrid method

Date	**Country**	**Number of Tweets**	**Common Words**	**Ground Truth**
31 May 2016, Tuesday, 10:54 AM	USA	465	[adam:0.63, dirty:0.82, doin:0.33, draymond:0.69, green:0.73, pick:0.23, pull:0.28, series:0.32, stay:1.0, ugly:0.37, wait:0.23]	2
02 June 2016, Thursday, 08:30 PM	USA	126	[cavalier:0.63, final:1.0, lock:0.37, nba:1.0, take:1.0, warrior:0.37, win:1.0]	10
03 June 2016, Friday, 10:00 AM	USA	215	[john:1.0, legend:1.01, light:0.25, nigga:0.24, skin:0.5, take:0.5, tryna:0.5]	15
03 June 2016, Friday, 10:18 AM	USA	119	[ankle:0.4, break:0.34, curry:0.27, fall:0.38, jab:0.98, make:0.26, step:0.93, thompson:0.34]	6
04 June 2016, Saturday, 01:18 PM	USA	277	[ali:0.98, dammit:1.0, god:0.92, muhammad:0.87, rip:0.81]	9
06 June 2016, Monday, 09:54 AM	USA	116	[back:0.38, deck:0.5, exactly:0.5, help:1.0, kevin:0.98, love:0.99, rush:0.5, soft:0.25]	11

APPENDIX B

APACHE CASSANDRA TABLES

B.1 Common Tables

Each method gets Twitter data and parse it before using. Therefore, "tweets" table is used with all methods. The schema of the table is described below:

```
CREATE TABLE tweets (
    round bigint,
    country text,
    tweettime timestamp,
    id bigint,
    retweetcount bigint,
    tweet text,
    userid bigint,
    PRIMARY KEY (round, country, tweettime, id)
)
```

B.2 Tables of Key-based Event Detection Method

There are 5 tables required for the uncommonly common algorithm. The first four of them are required for both approaches, using direct and shuffle grouping(suspension). The last table is only needed for the approach using direct grouping since the information of block execution done for each bolt is needed.

- Tweets table: As mentioned above tweets are replicated for each method using

this table.

- Counts table: Counts table is used to hold the count of words for the each block. Due to the tf-idf calculation, the count of the words in blocks are calculated again and again during the project, it slows the computation to count the same word many times. Therefore, we stored the counts of words with the round information into COUNTS table when it is calculated for the first time.

```
CREATE TABLE counts (
    round bigint,
    word text,
    country text,
    count bigint,
    totalnumofwords bigint,
    PRIMARY KEY (round, word, country)
)
```

- Events table: At the end of the event detection flow, the events found by our project are stored in the events table of Cassandra with event keyword, round and country information. The increment rate between tf-idf values of the last two rounds are also stored in the table.

```
CREATE TABLE events (
    round bigint,
    country text,
    word text,
    incrementpercent double,
    PRIMARY KEY (round, country, word)
)
```

- Process times: This table is used for the analytics of methodologies. This bolt stores the execution times of bolts for each tuple during blocks and process times are created according to this table.

```
CREATE TABLE processtimeskeybased (
  row int,
```

```
    column int,
    id int,
    PRIMARY KEY (row, column)
);
```

- Processed table: Processed table holds the information of bolt tasks even they finish executing the current block or not. Since the streaming of next block starts after all the tasks finish their executions of the current block, this table is needed.

```
CREATE TABLE.processedtaskskeybased (
    round bigint,
    boltid int,
    finished boolean,
    PRIMARY KEY (round, boltid)
);
```

B.3 Tables of Clustering-based Event Detection Method

There are 5 tables required for the clustering algorithm:

- Tweets table: As mentioned above tweets are replicated for each method using this table.

- Clusters table: This table holds all the active clusters during whole execution time. Clustering algorithm can add new clusters, update them or remove if they are inactive. Cluster map, number of tweets assigned to this cluster, last round of update and some other fields occur in this table.

```
CREATE TABLE clusters (
    country text,
    id timeuuid,
    cosinevector map text, double ,
    currentnumtweets int,
```

```
    lastround bigint,
    prevnumtweets int,
    PRIMARY KEY (country, id)
);
```

- Events table: At the end of each block clusters are reviewed and some of them are marked as events. This table is responsible to store the events.

```
CREATE TABLE eventclusters (
    round bigint,
    clusterid timeuuid,
    cosinevector map text, double ,
    country text,
    incrementrate double,
    numtweet int,
    PRIMARY KEY (round, clusterid)
);
```

- Process times: This table is used for the analytics of methodologies. This bolt stores the execution times of bolts for each tuple during blocks and process times are created according to this table.

```
CREATE TABLE processtimesclustering (
    row int,
    column int,
    id int,
    PRIMARY KEY (row, column)
);
```

- Processed table: Processed table holds the information of bolt tasks even they finish executing the current block or not. Since the streaming of next block starts after all the tasks finish their executions of the current block, this table is needed.

```
CREATE TABLE processedtasksclustering (
    round bigint,
```

```
    boltid int,
    boltprocessed bigint,
    country text,
    finished boolean,
    spoutsent bigint,
    PRIMARY KEY (round, boltid)
);
```

B.4 Tables of Hybrid Method

There are 7 tables required for the clustering algorithm:

- Tweets table: As mentioned above tweets are replicated for each method using this table.
- Clusters table: This table holds all the active clusters during whole execution time. Clustering algorithm can add new clusters, update them or remove if they are inactive. Cluster map, number of tweets assigned to this cluster, last round of update and some other fields occur in this table.

```
CREATE TABLE clustershybrid (
    country text,
    id timeuuid,
    cosinevector map text, double ,
    currentnumtweets int,
    lastround bigint,
    prevnumtweets int,
    PRIMARY KEY (country, id)
)
```

- Counts table: Counts table is used to hold the count of words for the each block. Due to the tf-idf calculation, the count of the words in blocks are calculated again and again during the project, it slows the computation to count the same

word many times. Therefore, we stored the counts of words with the round information into COUTNS table when it is calculated for the first time.

```
CREATE TABLE countshybrid (
  round bigint,
  word text,
  country text,
  count bigint,
  totalnumofwords bigint,
  PRIMARY KEY (round, word, country)
)
```

- Events table: At the end of each block clusters are reviewed and some of them are marked as events. This table is responsible to store the events.

```
CREATE TABLE eventshybrid (
  round bigint,
  clusterid timeuuid,
  cosinevector map text, double ,
  country text,
  incrementrate double,
  numtweet int,
  PRIMARY KEY (round, clusterid)
);
```

- Events keybased table:This table holds the event candidate keywords found by tf-idf calculation. This table is used only for development.

```
CREATE TABLE eventskeybasedhybrid (
  round bigint,
  country text,
  word text,
  incrementpercent double,
  PRIMARY KEY (round, country, word)
);
```

- Process times: This table is used for the analytics of methodologies. This bolt stores the execution times of bolts for each tuple during blocks and process times are created according to this table.

```
CREATE TABLE processtimeshybrid (
  row int,
  column int,
  id int,
  PRIMARY KEY (row, column)
);
```

- Processed table: Processed table holds the information of bolt tasks even they finish executing the current block or not. Since the streaming of next block starts after all the tasks finish their executions of the current block, this table is needed.

```
CREATE TABLE processedtaskshybrid (
  round bigint,
  boltid int,
  finished boolean,
  PRIMARY KEY (round, boltid)
);
```