EVENT DETECTION ON SOCIAL MEDIA USING TRANSACTION BASED STREAM PROCESSING ENGINE

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

BY

HÜSEYIN ALPER ÇINAR

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

JUNE 2019

Approval of the thesis:

EVENT DETECTION ON SOCIAL MEDIA USING TRANSACTION BASED STREAM PROCESSING ENGINE

submitted by **HÜSEYIN ALPER ÇINAR** 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, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. İsmail Sengor 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, Surname: Hüseyin Alper Çınar

Signature :

ABSTRACT

EVENT DETECTION ON SOCIAL MEDIA USING TRANSACTION BASED STREAM PROCESSING ENGINE

Çınar, Hüseyin Alper
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Pınar Karagöz

June 2019, 87 pages

The aim of this study is detecting events on social media by improving current solutions in terms of accuracy and time performance. An event is something that occurs in a short duration of time in a certain place. In this thesis, the problem is modelled as a streaming transaction process. Three different event detection method is adapted to our solution. First one is the keyword-based event detection method that looks for bursty keywords in a period. The second one is the clustering-based event detection method which is a version of the hierarchical clustering algorithm. And the last one is the hybrid event detection method of keyword-based and clustering-based algorithms. To specify the problem as streaming transaction process, all algorithms are implemented on top of S-Store. S-Store is a streaming OLTP engine having distributed, scalable and guaranteed ordered delivery features. All of the event detection methods are run and evaluated their performance with a real data set obtained from Twitter.

Keywords: Online event detection, Streaming online transaction processing, Distributed systems, Keyword-based event detection, Clustering-based event detection, Twitter, S-Store

ÖZ

İŞLEM TABANLI AKIŞ İŞLEME MOTORU İLE SOSYAL MEDYA ÜZERİNDEN OLAY ALGILAMA

Çınar, Hüseyin Alper
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Pınar Karagöz

Haziran 2019 , 87 sayfa

Bu çalışmanın amacı sosyal media üzerindeki olayları mevcut çözümlerin doğruluk ve süre performanslarını arttırarak algılamaktır. Bir olay, belirli bir yerde belirli bir zaman aralığında gerçekleşen şeyler olarak tanımlanabilir. Bu tezde, bu sorun işlem tabanlı akış işleme problemi olarak modellenmiştir. Üç değişik olay algılama yöntemi bizim problemimize uyarlanmıştır. Bunlardan ilki anahtar kelimeye dayalı olay algılamadır. Kısa süre içerisinde sayısal olarak fazla artışa sahip olan anahtar kelimeleri bulur. İkincisi kümelemeye dayalı olay algılamadır. Bunu hiyerarşik kümelemenin bir versiyonu olarak da tanımlayabiliriz. Son olarak, hibrid olay algılama yöntemi ile ilk iki yöntem birleştirilmiştir. Problemimizi işlem tabablı akış işleme problemi olarak tanımladığımız için tüm yöntemler S-Store üzerinde uygulanmıştır. S-Store, dağıtık, ölçeklenebilir ve garantili sıralı iletim özelliklerine sahip bir işlem tabanlı akış işleme moturudur. Tüm olay algılama yöntemleri Twitter'dan alınan gerçek veri seti ile çalıştırılmış ve performansları değerlendirilmiştir.

Anahtar Kelimeler: Çevirimiçi olay algılama, İşlem tabanlı akış işleme, Dağıtık sistemler, Anahtar kelimeye dayalı olay belirleme, Kümelemeye dayalı olay belirleme, Twitter, S-Store

to my beloved adigepsase...

ACKNOWLEDGMENTS

First of all, I'd like to thank Prof. Dr. Pınar Karagöz, my thesis advisor. She was always patient and supportive during the whole study. She involved every part of this thesis. She encouraged me and show me the right direction to get better results in this thesis.

I thank Nesime Tatbul for her suggestions and comments when re-modelling event detection methods to comply with S-Store. Her feedbacks on this thesis helped to enhance this work.

Of course, I thank Özlem Ceren Şahin for helping me to understand the internals of her event detection methods. She spent so many hours to support me in this work. I also thank John Meehan for supporting me technically when using S-Store.

I am grateful to my thesis defence jury members Assoc. Prof. Dr. İmail Sengör Altıngövde and Assist. Prof. Dr. Orkunt Sabuncu for evaluating this thesis with their valuable feedback.

I'd like to thank Güneş Sucu and my brother Arda Çınar for helping me extend ground truth events used on evaluation.

The special thanks go to my wife, Feyza Yılmaz Çınar. She gave all her support and help during this work. I owe her countless events and activities that she postponed because I was working on this thesis.

Lastly, I thank my mother, Ayşin Gökşin, who devoted her life to raise me and my brother and to support our education.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AWS	Amazon Web Services
CSV	Comma Seperated Values
EC2	Elastic Compute Cloud
NLP	Natural Language Processing
OLTP	Online Transaction Processing
tf-idf	term frequency-inverse document frequency

CHAPTER 1

INTRODUCTION

1.1 Overview

As the number of people using social media is increased through the years, it became one of the most reliable sources to gain insight about people's expressions, thoughts and events occurring around them. Since the content on social media is widely public, it gives data scientists an organic, objective data set to perform analysis.

Twitter is one of the most popular microblogging services. In the latest report [1], it is stated that Twitter has 330 million monthly active users. These users post 500 million daily tweets [2]. One of each contains a maximum 280 character length. These tweets also contain some valuable meta-data like geo-location, which obtained from the device that is used for sending these tweets.

Social media used in many of academic researches [3, 4, 5, 6, 7, 8]. In this work we will focus on enhancing Sahin's previous work [7, 8, 9] which proposes solutions for online event detecting on social media. We use the same terminology for an event, which is an activity occurring at a specific time and place, attracting attention in a short time [7].

The previous solution [7] achieved performance in manners of both accuracy and speed. This work aims to overcome the previous results by these manners. To make a decent comparison, the same data-set and similar configurations are used in experiments.

The problem considered for event detection in this study is a distributed streaming online transactional processing problem. It aims to enhance the proposed event detec-

tion methods which are keyword based, clustering based and hybrid event detection methods [7]. Keyword-based event detection method analyzes the burst of tweets, while clustering-based event detection method analyzes the tweets by their similarity. In addition to these 2 methods, there is a hybrid event detection which combines keyword based and clustering based event detection methods.

1.2 Contributions

The main aim of this thesis is enhancing a previous solution for event detection methods applied on social media using a distributed streaming OLTP engine. For this purpose, the implementation of three different event detection algorithms hypothesized in previous solution [7] is modelled as a streaming OLTP problem. These methods are namely keyword-based, clustering-based and hybrid event detection methods. Keyword-based event detection method looks for bursty keywords in a short time. Is the fastest method among these three, however it cannot find a relation between resulted keywords. On the other hand, clustering-based event detection method groups tweets contextually which makes it easier to deduct an event. However, its accuracy is not high as keyword-based event detection method although it runs in a longer time. To improve the accuracy of clustering-based event detection, hybrid event detection is implemented which is clustering tweets only containing bursty keywords.

To increase the time performance, in addition to performance enhancement methods used in the previous solution like implementing a distributed system, using S-Store, we were able to use an in-memory relational database which makes a huge impact on runtime duration. The runtimes of all of the event detection methods are significantly decreased.

To adapt the previous solution to S-Store, some other features are implemented on top of it. For example, reading streamed data and processing these data is split into different worker threads to eliminate data loss. The data limits of S-Store is also a challenge to fit the implementations. To overcome this, the corresponding portion of the data is extracted when data is received instead of splitting before sending.

In the experiments, pre-fetched twitter data used in the previous solution is also used

in this thesis. It makes the results of event detection methods to be comparable with both among each other and with the previous solution. When compared with the previous solution, we may now express results using minutes instead of long hours.

1.3 Organization of Thesis

This thesis consists of 7 chapters. This is the first chapter making an introduction. S-Store and the reasons to be used in this thesis is explained in the 2nd Chapter. Previous relevant studies about event detection methods are given on the 3rd Chapter. In Chapter 4, three different event detection methods are described with illustrative figures. The implementation details of these event detection methods with S-Store are given in detail in Chapter 5. Distributed and non-distributed configurations of event detection methods are also given in Chapter 5. In Chapter 6, the experiments with implemented event detection methods, comparison among those methods and comparison with the previous solution. Both accuracy and time performance are compared. Lastly, this thesis is concluded in Chapter 7.

CHAPTER 2

S-STORE

The event detection methods applied in this study are based on transactional online streaming processing system called S-store. The purpose of S-store is meeting the requirements of high-velocity streaming inputs while providing transactional robustness [10]. For that purpose, S-Store was built as an extension of H-Store which is a main-memory OLTP platform with strong support for state and transaction management [11]. With its streaming enhancements, S-store performs better than H-Store on a variety of streaming workloads [12].

One of the reasons for using S-store in this study is that it guarantees ordered execution which is necessary to ensure correct results [13]. With this feature, we always get the same result when a benchmark is executed with the same parameters and data set. This helps us evaluate the results more accurate.

Another reason is the speed of S-store. Since S-store uses main memory for streaming and transactions, it provides low latency. S-Store has built-in support for Volt DB which is an in-memory relational database. With this support, S-Store optimizes the scheduling of the DB transactions and streams itself, which helps to speed up the execution. Although it requires more memory when executing an S-Store benchmark, proper cleanup methods and optimization help us not to reach the memory limits in this study.

Lastly, its distributed structure is another factor contributing to execution speed. Running procedures in different hosts or partitions help to decrease the runtime duration. Also, designing the configuration of event detection methods for a distributed system, helped us set our mindset to encapsulate the data within a procedure and flatting the

output of a procedure when sending to next procedure. The details of the implementation and S-Store adaptations can be found in the following chapters.

CHAPTER 3

RELATED WORK

Today, social media is the most important communication channel that people use to make their voices heard. Lots of people share their thoughts, moods, activities and the *events* around them. By the help of the Twitter API researchers start to mine the tweets to detect the events of any size from worldwide sports activities to local fairs. Some of these researchers focused on a specific type of events while others focus on more generic perspective. Moreover, some studies investigate and classify these studies. Atefeh and Khreich [5] classified event detection techniques by their types, detection technique and detection task and proposes application areas for these techniques. Becker et al. [14] helped to reveal important information about real-world events by proposing an approach to distinguish Tweets about the real-world events from other events. According to Atefeh and Khreich [5] this approach is suitable for general event detection. Another study about distinguishing real-world events is conducted by Walther and Kaisser [15]. Their aim was identifying real-world events and presenting them to the users on a map which makes the information more actionable.

The geotagging feature of Twitter is a great feature to identify and segment Tweets by location. However, only 0.7% of documents are geotagged [16]. To resolve this lack, Jasmine [16] proposed an automatic geotagging method and the number of geographic groups increased dramatically with this method.

Aiello et al. [17] compared the performance of different topic detection algorithms on Twitter streams and found that standard NLP techniques performed better on focused topics while novel techniques perform better on heterogeneous streams.

Li et al. [18] compared their segment-based event detection technique “Twevent" with the state-of-the-art method and found out Twevent outperforms in terms of both precision and recall. Petrovic et al. [19] adapted locality sensitive hashing to the first story detection and experimented with a truly large scale data (160 Million Tweets).

Mathioudakis and Koudas [20] created a trend detection system “TweetMonitor” which is based on detecting and grouping bursty keywords. Similarly the work of Sankaranarayanan et al. [21] called “TwitterStand”, focuses on cleaning noisy data from the Tweets and detecting late-breaking news from them.

About emerging topics on Twitter, Cataldi et al. [22] proposed an approach using ageing theory to mine terms that frequently used in specified time interval while they are not popular at other times.

Ritter et al. [23] describes the first open domain event-extraction and categorization system for Twitter called “TwiCal” which is based on natural language processing.

Another system developed about event detection is TEDAS [24] which has offline and online processing mechanisms. Its offline processing mechanism determines and stores Crime and Disaster-related Events from Twitter API while online processing mechanism answers user’s queries by generating visual results. EvenTweet [25] uses an online detection algorithm which works in a similar manner of TEDAS’s and shows detected localized events according to the user’s search queries. The study of Becker et al. [14] is also an example of an online event detection mechanisms since they use an online clustering technique.

Cordeiro and Frias [26] studied on an event detection method using wavelet signal analysis of hashtags in tweets and combined them with Latent Dirichlet Allocation topic inference model for a better description of the events. Weng and Lee [27] conducted another event detection study based on wavelet-based signals called EDCoW (Event Detection with Clustering of Wavelet-based Signals).

Lee and Sumiya [28] proposed Geo-social Event Detection Method to detect location based expected or unexpected events by using 21 million geo-tweets found around Japan between a specific time interval.

While the above studies focus on general event detection using Twitter API, there are some studies focused on special event detection like traffic event detection from Twitter streams [29]. Sakaki et al. [30] also focused on a specific event and proposed an algorithm to monitor tweets and to detect a target event such as an earthquake. The system detects earthquakes and notifies users much faster than broadcasts.

In this work, we searched for the approaches that would fit into our model. We used geotagging to separate our dataset into countries to focus on regional events. NLP techniques are highly used in the pre-processing phase to extract only root words to help our similarity calculations. Similar to previous works, this work is also focused on keyword bursts. Additionally, this work also handles a group of tweets by building tweet clusters and tries to detect events among those clusters. However, the main difference of this study is that we use a transactional streaming model to detect events. This work is the first event detection system that uses S-Store. With S-Store, we process a group of tweets as an atomic unit in a stream easily.

CHAPTER 4

EVENT DETECTION METHODS

In this chapter, two different event detection methods and the hybridization of these two methods are explained in detail. First keyword-based and clustering-based event detection methods are discussed. Lastly, the hybrid event detection method is explained in detail.

Tweets are grouped by the publishing times with a predefined time interval. These time intervals will be called *rounds* on the remaining chapters. All of the event detection methods depend on the difference of the state of the processed data between two consecutive rounds.

4.1 Keyword-Based Event Detection Method

The main interest of this event detection method is to find uncommonly common words within each round. These words are treated as events. By looking for uncommonly common words, we eliminate the words that are common and words that are slightly used in every round.

Marking a word to be an uncommonly common word, term frequency-inverse document frequency, or tf-idf, algorithm is used. Since tf-idf algorithm is mainly used for documents, we may treat all tweets in a round as one whole document to use this algorithm.

Tf-idf can be calculated with the following equation:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{41}$$

Where,

- t: the corresponding term
- d: the document containing t
- D: the set of documents of the corpus
- $tf(t, d)$: term frequency of t in d (See Equation 42)
- $idf(t, D)$: inverse document frequency of t in D (See Equation 43)

Term frequency can be found by the following equation:

$$tf(t, d) = \frac{|\{t \in d\}|}{|d|} \tag{42}$$

Where,

- $|\{t \in d\}|$: the number of times that t occurs in d
- $|d|$: the total number of terms in d

And inverse document frequency can be calculated by the following equation:

$$idf(t, D) = \log \frac{|D|}{|d \in D : t \in d|} \tag{43}$$

Where,

- $|D|$: total number of documents
- $|d \in D : t \in d|$: the number of documents containing t

As proposed in [7], to increase performance we use tweets of last 2 rounds as documents in equation 43. However, using such number of documents causes the $idf(t, D)$ formula to result either $\log(1)$ or $\log(2)$. To have more precise idf value, we used the following equation when calculating idf value instead of equation 43:

$$idf(t, D) = \log \frac{\sum_{d \in D} |d|}{\sum_{d \in D} |t_d|} \tag{44}$$

Where,

- $|d|$: total number terms in document d
- $|t_d|$: total number of term t in document d

After finding tf-idf value for each word, we compare each value with the tf-idf calculated on the previous round. Words having an increased rate of tf-idf value for a certain threshold are considered as events.

Figure 4.1 shows the steps of keyword based event detection method using tweets.

Figure 4.1: Keyword-Based Event Detection Steps

4.2 Clustering-Based Event Detection Method

In this method, an event is defined by the cluster of tweets having the highest growth rate. Clusters are constructed by words of similar tweets where cosine similarity is used for the measurement for similarity. Both tweets and clusters are represented by vectors to calculate cosine similarity with the following formula:

$$\cos(\theta) = \frac{\vec{T_1} \cdot \vec{T_2}}{|\vec{T_1}||\vec{T_2}|} \tag{45}$$

Where,

- $\vec{T_1}$ first tweet as vector
- $\vec{T_2}$ second tweet as vector
- $\cos(\theta)$ cosine similarity between $\vec{T_1}$ and $\vec{T_2}$

In the beginning, this method starts with zero clusters and increases to an undefined number. There are two types of clusters created in this method. First one is the local cluster. A local cluster is a cluster containing tweets from only one specific round. The other one is the global cluster. A global cluster consists of one or multiple local clusters that are similar. After a round is completed, eligible local clusters may become a global cluster or merge into an existing one. Eligibility of a local cluster is defined by the number of tweets contained.

After global clusters are extended at the end of each round, ones that grow with a certain threshold rate are defined as events.

Figure 4.2 shows the steps of clustering based event detection method using tweets.

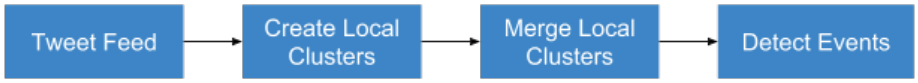

Figure 4.2: Clustering-Based Event Detection Steps

4.3 Hybrid Event Detection Method

In the last method, the first two methods are applied consecutively. First keyword based event detection method is applied to find bursty keywords as explained in 4.1. After finding bursty keywords, tweets containing these keywords in the current round are supplied to the clustering-based event detection method instead of all tweets. Then, the clustering-based event detection method is run to find events as explained in 4.2.

By filtering tweets by having only burst keywords, it is aimed to increase the performance by reducing the number of operations when creating local and global clusters.

Since cluster operations are costly, the reduction of the number of tweets impacts efficiency.

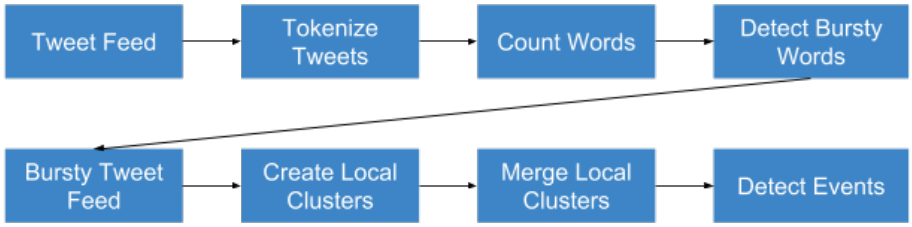

Figure 4.3: Hybrid Event Detection Steps

4.4 Illustrative Example

In this section, an illustrative example is given to have a better understanding of event detection methods. We'll go through the steps of keyword-based and clustering-based event detection methods with the sample dataset given in Table 4.1. There are 9 tweets and 3 rounds on this dataset. For the sake of simplicity, we keep tweets as simple as possible.

First, we describe keyword-based event detection. For this example, we make the following assumptions:

- the increase rate of tf-idf value of keyword should be at least 1.05 to be detected as a bursty keyword.
- If a given keyword does not exist on the previous round, its tf-idf should at least be 0.375 to considered as a bursty keyword.

Based on those assumptions, there is no bursty keyword on the first round since none of the keywords has required tf-idf value. When it comes to the second round, we see that the increase rate of tf-idf value of "kobe" is 1.05, which fits our requirements. We may say that "kobe" is a bursty keyword. In addition, "concert" is a keyword having 0.389 tf-idf value and has no previous value. Therefore "concert" is another bursty

Table 4.1: Sample Dataset for Illustrative Example

Round No	Tweet Id	Tweet
1	t_1	kobe shot
1	t_2	caramel tasty
2	t_3	kobe shot
2	t_4	kobe layup
2	t_5	concert great
2	t_6	kobe night
2	t_7	concert rock
3	t_8	kobe best
3	t_9	concert end
3	t_{10}	kobe video
3	t_{11}	dance good

keyword. On the third round, there is no keyword matching our criteria. Table 4.2 shows the calculations of each round.

For clustering based event detection method, we make the following assumptions:

- A cluster should have at least 2 tweets to considered as a global cluster
- A global cluster should have at least 3 tweets to represent an event
- A global cluster should grow at least 2 times or new structured to represent an event

On first round, 2 local clusters L_1 and L_2 are constructed with 1 tweets. Those local clusters do not match the criteria to be a global cluster. Therefore no event is detected on the 1st round. On the 2nd round, 2 local clusters L_3 and L_4 are generated and both of them are eligible to be a global cluster. Since L_3 is new structured and has 3 tweets, we may mark L_3 as an event. On 3rd round, 3 new clusters are structured but only L_5 is eligible to be a global cluster. Since L_5 and L_3 are similar clusters, L_5 merges into L_3, which makes L_3 1.66 times larger. However, although L_3 has enough number of

Table 4.2: Illustrative Example: Keyword-Based Event Detection Flow

Round No	Word	Count	tf-idf	tf-idf rate
1	kobe	1	0.347	-
	shot	1	0.347	-
	caramel	1	0.347	-
	tasty	1	0.347	-
2	kobe	3	0.376	**1.05**
	shot	1	0.195	0.56
	layup	1	0.264	-
	concert	2	**0.389**	-
	rock	1	0.264	-
	great	1	0.264	-
	night	1	0.264	-
3	kobe	2	0.320	0.85
	best	1	0.361	-
	concert	1	0.224	0.58
	end	1	0.361	-
	dance	1	0.361	-
	good	1	0.361	-
	video	1	0.361	-

tweets to be an event, since its growth rate is less than specified, it does not represent an event on this round.

These steps for keyword-based and clustering-based event detection methods are also applied in hybrid event detection method. The only difference for the hybrid is, less number of tweets passes to clustering-based event detection phase since most of the tweets are eliminated on the clustering-based event detection phase.

Table 4.3: Illustrative Example: Clustering-Based Event Detection Flow

Round No	Local Clusters	Global Clusters	Cluster Growth Rate
1	$L_1 : \{t_1\}$	-	-
	$L_2 : \{t_2\}$	-	-
2	$L_3 : \{t_3, t_4, t_6\}$	$L_3 : \{t_3, t_4, t_6\}$	-
	$L_4 : \{t_5, t_7\}$	$L_4 : \{t_5, t_7\}$	-
3	$L_5 : \{t_8, t_{10}\}$	$L_3 : \{t_3, t_4, t_6, t_8, t_{10}\}$	1.66
	$L_6 : \{t_9\}$	$L_4 : \{t_5, t_7\}$	1.0
	$L_7 : \{t_{11}\}$		-

CHAPTER 5

EVENT DETECTION ON SOCIAL MEDIA USING S-STORE

In this chapter, we will show our efforts when implementing event detection methods with S-Store. First, we will cover the elements of S-Store and then details about non-distributed and distributed implementations of event detection methods. Lastly, we will talk about our challenges and workarounds when adapting event detection methods to S-Store.

5.1 S-Store Client

In S-Store, the client is the source of the input. It prepares input data and triggers the initial procedure of the benchmark. In all of the event detection methods, the client reads the tweets in CSV format from an input source and sends them to the initial procedure.

5.2 Common S-Store Procedures

In this section, procedures used in keyword-based, clustering-based and hybrid event detection methods are explained in detail. 9 different procedures in total are re-used in different event detection methods.

5.2.1 Feed Procedure

Feed Procedure is the initial procedure of every event detection method. It is responsible for reading raw data obtained from the client and structuring them to be used

in the next procedures. Feed Procedure streams the id, country, round and text of the tweets with tweets_s stream defined in A.1.1.

5.2.2 Tokenizer Procedure

Tokenizer Procedure is a procedure to tokenize tweets into words within a round. This procedure can be distributed and each of distributed instance sends its part to next procedure with tokenized_word_counts_s stream defined in A.2.1.

5.2.3 Word Count Procedure

Word Count Procedure is responsible for merging tokenized tweets within a round. It sends the merged results to next procedure with word_counts_s stream defined in A.2.2.

5.2.4 Burst Detector Procedure

Burst Detector Procedure computes TF-IDF of words on the current round by comparing them with the previous round. It sends bursty words with burst_words_s defined on A.2.3

5.2.5 Burst Report Procedure

Burst Report Procedure is a procedure that reports the number of bursty words. This procedure retrieves word counts of a given keyword for last 10 rounds and plots a graph for the result. This is the last procedure of keyword-based event detection method.

5.2.6 Tweet Filter Procedure

Tweet Filter Procedure is a procedure responsible for gathering tweets having bursty keywords and streaming them to the next procedure. The output stream of this proce-

dure is the same as Feed Procedure and defined in A.4.2.

5.2.7 Local Clustering Procedure

Local clustering procedure is a procedure that can be distributed among different partitions. It is responsible for clustering a subset of tweets in a round and then streaming clustered tweets to the next procedure. This cluster reads structured tweet data to generate clusters. It streams local clusters to the next round with local_cluster_results_s defined on A.3.1.

5.2.8 Cluster Merge Procedure

Cluster Merge Procedure is a procedure that obtains cluster information from multiple instances of Local Clustering Procedure. This procedure is responsible for merging similar local clusters constructed in different partitions. It streams output clusters to the next round with merged_cluster_results_s defined on A.3.2.

5.2.9 Clustering Event Detector Procedure

This is the final procedure of both clustering-based and hybrid event detection. Clustering Event Detector Procedure reads latest clusters, compares them with the clusters from previous rounds and marks events accordingly.

5.3 Procedure Input Outputs

The input and output of all procedures are given in Table 5.1. In addition to given input and output, all of the procedure I/O contains country and round information.

Table 5.1: Inputs and Outputs of Procedures

Procedure Name	Input	Output
Feed Procedure	Raw tweet	Structured tweet
Tokenizer Procedure	Structured tweet	Words of tweets
Word Count Procedure	Words of tweets	Total counts for each word
Burst Detector Procedure	Total counts for each word	Bursty words
Burst Report Procedure	Bursty words	Historical word counts for previous rounds
Local Clustering Procedure	Structured tweet	Local clusters of tweets
Cluster Merge Procedure	Local clusters of tweets	Merged clusters of local clusters
Clustering Event Detector Procedure	Merged cluster of tweets	Clusters representing an event
Tweet Filter Procedure	Bursty words	Structured tweets containing bursty words

5.4 Keyword-Based Event Detection

In this section, we will discuss implementing the keyword-based event detection method. 3 different configurations are implemented to perform this method. One of them is non-distributed while the other two is distributed among different S-Store hosts. In all of the implementations, Feed Procedure, Tokenizer Procedure, Word Count Procedure, Burst Detector Procedure and Report Procedure is used. The outputs and performance of these implementations will be discussed in the next chapters.

The general flow of all implementations are as the following:

- S-Store client reads unstructured tweet data and forwards them to Feed Procedure,
- Feed Procedure structures raw tweets and sends their country, round number and text of the tweet to Tokenizer Procedure
- Tokenizer Procedure tokenizes incoming tweet text into words and groups them by their country, round number and word counts
- Word Count Procedure merges all tokenized word counts and sends them to Burst Detector Procedure when the corresponding round is completed
- Burst Detector Procedure performs bursty keyword detection and sends them to Report Procedure. This procedure also writes word counts into the database

to be used when preparing the report.

- Report Procedure reads counts of bursty keywords from the database and prepares a report.

The only constraint in these implementations is that Burst Report Procedure should be in the same partition with Burst Detector Procedure to increase the performance as they are using the same database table.

5.4.1 Non-Distributed Implementation

In the non-distributed implementation, all of the procedures are configured to run sequentially in the same S-Store partition. This is our worst implementation since we prevent procedures from running in parallel. The configuration is illustrated in Figure 5.1.

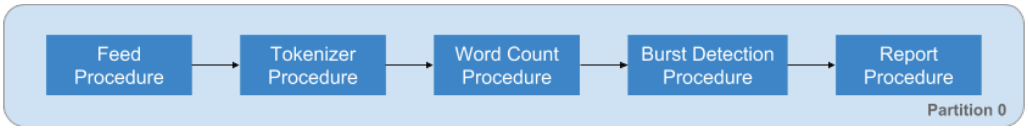

Figure 5.1: Keyword-Based Event Detection: Non Distributed Configuration

5.4.2 Distributed Implementation

For distributed implementation, our first option placing each procedure into different nodes. There are some constraints we need to follow. Feed procedure cannot be distributed since it is the initial procedure. We may distribute the Tokenizer Procedure and Word Count procedure with multiple instances. The only way to distribute Burst Detection Procedure is distributing it by country. Because we need the whole round of data of a country to be able to run the Burst Detector Procedure. Therefore, we may have at most 2 Burst Detector Procedure for USA and Canada. Since Report Procedure is dependent on Burst Detector Procedure, they should be in the same hosts. The configuration is represented in Figure 5.2.

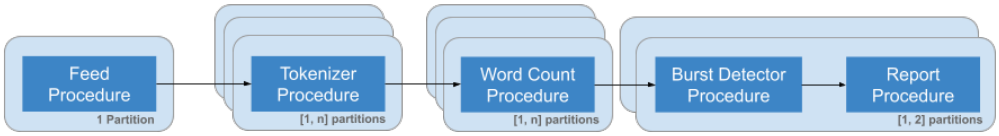

Figure 5.2: Keyword-Based Event Detection: Country-Wise Distributed Configuration

5.5 Clustering-Based Event Detection

The general flow of all implementations are as the following:

- S-Store client reads unstructured tweet data and forwards them to Feed Procedure,
- Feed Procedure structures raw tweets and sends their country, round number and text of the tweet to Local Clustering Procedure
- Local Clustering Procedure generates clusters for a round, labels every tweet with a local cluster id and sends the to Cluster Merge Procedure
- Cluster Merge Procedure merges the clusters retrieved from Local Clustering Procedure having same round and updates the cluster labels of every tweet accordingly and sends them to Cluster Event Detection Procedure
- Cluster Event Detection Procedure checks clusters to find an event

5.5.1 Non-distributed Implementation

Like in non-distributed implementation of keyword-based event detection, this implementation also runs all of the procedures sequentially in the same S-Store partition. This configuration is shown in Figure 5.3.

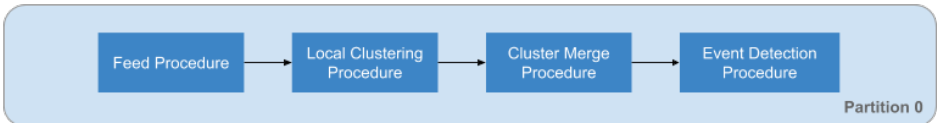

Figure 5.3: Clustering Based Event Detection: Non-Distributed Configuration

5.5.2 Distributed Implementation

Again, we may start by distributing each procedure to different hosts. While Local Procedure can have multiple instances in multiple hosts, Cluster Merge Procedure and Event Detector Procedure can only have at most 2 instances since they can only be distributed by country. This configuration can be observed on 5.4.

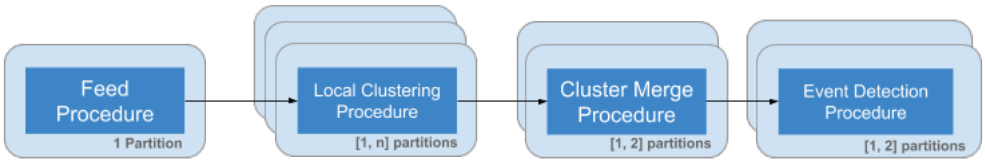

Figure 5.4: Clustering Based Event Detection: Distributed Configuration

5.6 Hybrid Event Detection

The general flow of all implementations are as the following:

- All of the procedures in Keyword-based event detection listed in 5.4 runs except Report Procedure
- Instead of Report Procedure, Tweet Filter Procedure receives bursty words and sends tweets having bursty words to next procedure
- All of the procedures in Clustering-based event detection listed in 5.5 runs except Feed Procedure since the data is supplied from Tweet Filter Procedure

5.6.1 Non Distributed Implementation

For non-distributed configuration, we first call procedures of keyword-based event detection method except Report Procedure. After keyword-based event detection procedures, we call Tweet Filter Procedure to find candidate tweets. Lastly, we call the procedures of the clustering-based event detection method. This configuration is illustrated in 5.5.

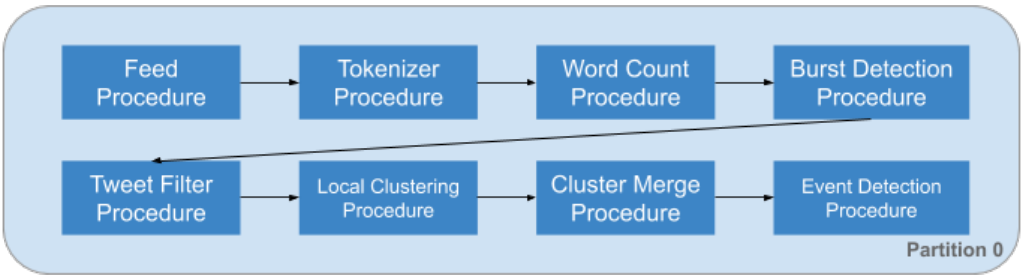

Figure 5.5: Hybrid Event Detection Configuration: Non-Distributed Configuration

5.6.2 Distributed Implementation

We follow a similar way we do for non-distributed configuration. The only constraint we have is that Feed Procedure and Tweet Filter Procedure should share the same node since Tweet Filter Procedure needs to access those tweets. Therefore we may implement this configuration as seen in Figure 5.6.

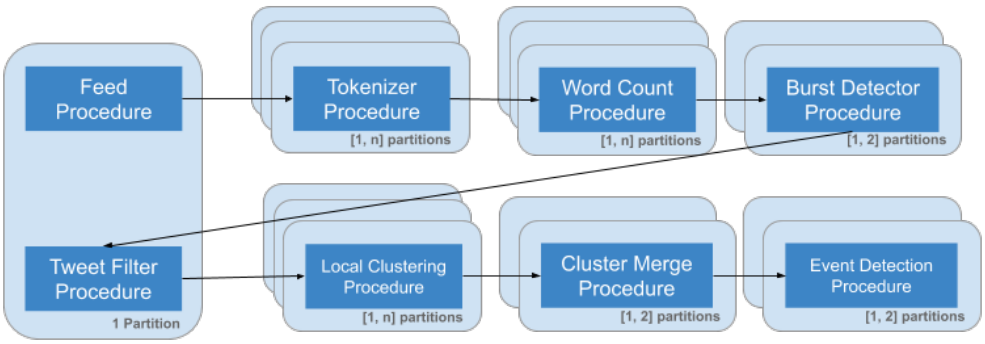

Figure 5.6: Hybrid Event Detection Configuration: Distributed Configuration

5.7 Adapting to S-Store

Implementing event detection methods, S-Store provided us with the basic requirements including streaming, transaction scheduling, fast data flow and guaranteed ordering. However, we required more features to complete the implementation. In this section, we'll give details about the problems and challenges we have encountered and our solutions.

First of all, since S-Store is an experimental tool, we have encountered some unexpected issues that we could not found a solution from the documentation. In these situations, we needed to contact the S-Store developers directly to get help and feedback for our implementation. This was the most time-consuming part since we needed to schedule a meeting.

One of the challenges is that fitting the implementation to the limitations of S-Store. In S-Store, the number of rows to downstream the next procedure is limited by 1000. This is fine when the data is processed row by row. However, we process the data round by round and we cannot pass the whole round to the next procedure. Therefore we split the rounds into rows of 1000s and track when a round is completely passed to the next procedure. And when a round is completely passed, we process that round and put the output of procedure in a queue so that we send them 1000 by 1000 on next transaction.

Since we need to wait for a round to be completed to run a procedure's main work, most of the times the procedure produces nothing as output when a new data arrives. However, S-Store needs at least one tow to be sent to the next procedure to continue the execution. In this scenario, we pass a *null* row to the next procedure.

Another problem we have encountered is the rejection of procedures. If a procedure is currently busy when new data arrives, that procedure rejects new data and causes data loss. It generally occurs on long-running procedures like Cluster Merge Procedure or Event Detection Procedure. The first solution decreasing the rate of the input stream, however, it increases the runtime duration. To minimally effect the procedure duration, we just receive data and write them to a temporary place. When a round of data is completely received from the previous procedure, we create an additional

thread to process the data so that it does not block the streaming and thus decreases the rejection. To eliminate the rejection, we both tuned the input rate and run the procedures main work in a different thread.

CHAPTER 6

EXPERIMENTS

6.1 Setup

All of the experiments in this chapter have run the on AWS t3.2xlarge EC2 instance.

- CPU: 8 vCPU, Intel Xeon Platinum 8000 3.1 GHz
- Memory: 32 GB + 16 GB Swap
- Storage: 250MB/s Read/Write throughput

6.2 Data Set

To make a decent comparison between event detection methods and configurations, all of the experiments listed in this chapter have used the same data set. This data set contains about 12M tweets gathered from the USA and Canada between May 31 and June 6, 2016. The previous solution also uses exactly the same data set. Therefore we can make a fair accuracy comparison with the previous solution.

To simulate a real tweet stream, all of these tweets are written into a CSV file ordered by publishing time. This CSV file is served through a socket to S-Store client using an intermediate program, namely, stream ingestor.

6.3 Preprocessing

Preprocessing is applied before all of the experiments to merge words with the same root and eliminate unnecessary words, URLs, mentions and punctuation marks as proposed in [7]. Table 6.1 shows some examples of preprocessed tweets.

Table 6.1: Sample Input and Output for Prepocessing

Input	Output
I love shopping online for things I don't need it makes me feel fulfilled	love shopping online thing need make feel fulfil
Success - located fried chicken on a stick at Sasquatch to feed my addiction!! #Sasquatch2016 #chickencrawl	success located fried chicken stick sasquatch feed addiction #sasquatch #chickencrawl
STEPH!!!!!!!!!!!!!!	steph
4 3's and it's tied..	tie
@Kellinquinn I thought you would enjoy this video of you talking to Copeland at the @SWStheband concert in Oregon :)	think enjoy video talk copeland concert oregon
I'm at @BigChefsCafe in Ankara	-

6.4 Ground Truth Construction

To evaluate the accuracy of event detection methods explained in Chapter 4, first we need to have a set of events that occurred between May 31 and June 6, 2016. Since we cannot possibly know all of the events in that interval, we used the events detected by our event detection methods. At the initial point, we used events already proposed on [7]. These events are determined and cross-checked by different judges. There are 20 events and these events are listed on Table B.1.

In addition to existing events, there are more events detected in our experiments. Like on [7] the outputs of experiments are reviewed by 3 different judges and the individual results are discussed in a session which all of the judges are attended. These events are constituted by searching among tweets, news and videos between the date interval

of data set using the words obtained from event detection methods. As a result of this session, 23 events are accepted and these events are listed on Table B.2.

As a result, a total number of 43 events are used as ground truth when evaluating event detection methods. While 41 of these events except 24th and 28th are detected in the USA, only 19 events are detected in Canada. The event indices detected in Canada are 1, 2, 3, 5, 6, 8, 9, 11, 14, 15, 17, 18, 20, 23, 24, 28, 29, 39 and 42.

6.5 Evaluation Metrics

In this section, metrics used for evaluating accuracy and time performance are discussed.

6.5.1 Accuracy Evaluation Metrics

To evaluate the accuracy of benchmarks, we used three metrics, namely Precision, Recall and F-measure.

To find a precision of the results we use different units for different event detection methods. Since the output of the keyword-based event detection is bursty words, we consider the number of keywords corresponding an event while having clusters as the output of clustering based and hybrid methods, we consider the number of clusters. Precision for keyword-based event detection method is given in the equation 61.

$$\text{precision}_k = \frac{\text{\# of keywords matching an event}}{\text{Total \# of keywords}} \tag{61}$$

Precision for clustering-based and hybrid event detection method is given in the equation 62.

$$\text{precision}_c = \frac{\text{\# of clusters matching an event}}{\text{Total \# of clusters}} \tag{62}$$

Another metric used when evaluating the accuracy of event detection method is recall, which is essentially the ratio of outputs corresponding to an event and the total number

of events. Since we cannot know all of the events in the time interval of our dataset, we consider only the events we defined in section 6.4. Therefore, the formula for recall becomes the following:

$$\text{recall} = \frac{\text{\# of detected events}}{\text{Total \# of events in ground truth}} \tag{63}$$

The last metric, F-measure, is calculated as the harmonic mean of precision and recall. It gives us a more comparable value since it combines both of two other metrics. F-measure is calculated with the following formula:

$$\text{f-measure} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{64}$$

6.5.2 Time Performance Evaluation Metrics

To evaluate the time performance of benchmarks, we used three metrics namely Tweets per Second, Rounds for Minute and Total Execution Time.

6.6 Parameter Tuning

In this section, efforts to find the best parameters for clustering and hybrid methods are shown. As explained in Chapter 5, there are 3 parameters defined for clustering-based and hybrid event detection methods. These are the following:

- p_1: Minimum cosine similarity between tweets and clusters used when assigning a tweet into a cluster, and between two clusters used when merging them.
- p_2: Minimum number of tweets that a local cluster needs to have to be marked as a global cluster.
- p_3: Minimum number of tweets that a global cluster needs to have to be accepted as an event.

We defined a set of these parameters and run them one by one to find out the best result. The set of parameters are defined is the following, where a combination of (P_1, P_2, P_3) are defined differently on clustering-based and hybrid event detection methods for USA and Canada. The details are in following subsections.

$$(p_1, p_2, p_3) = \{p_1 \in P_1\} \times \{p_2 \in P_2\} \times \{p_3 \in P_3\}$$

There will be a high number of combinations for $\{p_1, p_2, p_3\}$ tuple. However, for parameter tuning, we may keep the parameter p_3 minimum and run the benchmark only for $\{p_1, p_2\}$. We may then post-process the output and filter out clusters that do not fit the other values for $p3$.

6.6.1 Parameter Tuning for Clustering-Based Event Detection of USA

(P_1, P_2, P_3) is defined by the following

- P_1: $\{0.45, 0.5, 0.55, 0.6, 0.65, 0.7\}$
- P_2: $\{40, 60, 80, 100, 120\}$
- P_3: $\{75, 100, 125, 150\}$

After running all of the configurations combinations the followings are observed:

- When $p_1 = 0.7$, the number of clusters are between 8 and 20, which are far less than the total number of events. We may safely eliminate these configurations.
- When $p_1 = 0.45$, the number of clusters are between 109 and 1285, which are far more than the total number of events. This parameter can also be safely eliminated.

All of the (p_1, p_2, p_3) where $p_1 \in \{0.5, 0.55, 0.6, 0.65\}$ is run and configurations having the best performance are listed in Table 6.2. Although $(0.6, 80, 75)$ has the highest precision and f-measure it has very low recall. Then may choose $(0.5, 120, 125)$ since its f-measure value is close to $(0.6, 80, 75)$ and its precision and recall are more uniform. The complete table with all configurations can be found on C.1.

Table 6.2: Parameter Tuning Results for USA Clustering Based Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.6	80	75	12	0.780	0.293	0.426
0.5	120	125	15	0.470	0.366	0.411
0.5	120	100	15	0.447	0.366	0.402
0.5	120	75	15	0.447	0.366	0.402
0.5	120	150	12	0.633	0.293	0.400

6.6.2 Parameter Tuning for Clustering-Based Event Detection of Canada

Since the data set of Canada is smaller than the USA's, smaller values for P_2 and P_3 is chosen as follows:

- P_1: $\{0.5, 0.55, 0.6, 0.65\}$
- P_2: $\{10, 15, 20, 25, 30\}$
- P_3: $\{18, 25, 31, 37\}$

The results of the configurations are listed in Table 6.3. As we can see, $(0.5, 30, 31)$ is accepted since it has the highest f-measure. The complete result list of configurations can be found in C.2

Table 6.3: Parameter Tuning Results for Canada Clustering Based Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.5	30	31	10	0.351	0.526	0.421
0.5	25	31	11	0.318	0.579	0.411
0.5	30	25	10	0.333	0.526	0.408
0.5	30	18	10	0.333	0.526	0.408
0.5	25	25	13	0.266	0.684	0.383

6.6.3 Parameter Tuning for Hybrid Event Detection of USA

For hybrid event detection of USA, (P_1, P_2, P_3) is defined by the following

- P_1: $\{0.5, 0.55, 0.6, 0.65\}$
- P_2: $\{20, 30, 40, 50, 60\}$
- P_3: $\{30, 50, 60, 75\}$

Since the filtered tweet number is far less than the original tweet number, low values are used for P_2 and P_3. As seen on 6.4, the winner in this case undoubtedly $(0.5, 30, 30)$ since it has the maximum of both precision and recall. The complete table with all configurations can be found on C.3.

Table 6.4: Parameter Tuning Results for USA Hybrid Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.5	30	30	25	0.618	0.610	0.614
0.5	20	30	23	0.611	0.561	0.585
0.5	40	30	20	0.591	0.488	0.534
0.55	30	30	18	0.578	0.439	0.499
0.55	20	30	18	0.565	0.439	0.494

6.6.4 Parameter Tuning for Hybrid Event Detection of Canada

Lastly, for Canada we used the following options for the configuration of hybrid event detection:

- P_1: $\{0.5, 0.55, 0.6, 0.65\}$
- P_2: $\{5, 7, 10, 12, 15\}$
- P_3: $\{7, 12, 15, 18\}$

As you may be noted, since tweets having bursty keywords in Canada is the smallest data set so we used the smallest options for the parameters. As seen in Table 6.5 all top 5 configurations have similar f-measure values so we need another criterion to pick the best configuration. By checking silhouette coefficients of resulting clusters on Table 6.6, we may choose one of $(0.5, 7, 7)$ and $(0.5, 5, 7)$ since they have the maximum average silhouette coefficient and has the same result. The full table of configurations can be found in Table C.4

Table 6.5: Parameter Tuning Results for Canada Hybrid Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.5	7	7	3	0.833	0.158	0.265
0.5	5	7	3	0.833	0.158	0.265
0.55	7	7	3	0.800	0.158	0.264
0.55	5	7	3	0.800	0.158	0.264
0.5	7	18	3	0.750	0.158	0.261

Table 6.6: Silhouette Coefficients of Parameter Tuning Results for Canada Hybrid Event Detection

p_1	p_2	p_3	Max	Min	Average	Standard Deviation
0.5	7	7	1.0	0.05	0.46	0.34
0.5	5	7	1.0	0.05	0.46	0.34
0.55	7	7	1.0	0.11	0.57	0.33
0.55	5	7	1.0	0.11	0.57	0.33
0.5	7	18	1.0	0.39	0.60	0.28

6.7 Event detection accuracy

In this section, we analyze and compare the accuracy of event detection methods. First, we show the accuracy results of event detection methods of our implementation. Then, we compare those accuracy results. Lastly, we compare our results with the

previous solution.

6.7.1 Accuracy Comparison Among Event Detection Methods

On the result of keyword-based event detection, 212 bursty keywords are found in total. 196 of these keywords are from the USA while 16 of them are from Canada. After matching the keywords with the events listed as ground truth, we see that 96 of these keywords points to 36 different events. The list of bursty keywords and corresponding events are listed on Table D.1 for USA, and Table D.2 for Canada. As a result Table 6.7 is constructed by the summary of keyword-based event detection results.

Table 6.7: Results of Keyword-Based Event Detection Method

Country	Bursty Keyword #	Keyword # Matching an Event	Detected Event #	Undetected Event #
USA	196	88	30	11
Canada	16	10	6	13
All	212	98	36	24

We have run various configurations when tuning parameters for clustering-based and hybrid event detection on Section 6.6. Considering the best performing configuration, we have found 140 event clusters for clustering-based event detection. 83 of them are for the USA while 57 of them are for Canada. After assigning ground truth events for each cluster, we see that 59 different events mapped to resulting clusters. The cluster and event mapping can be found on Table E.1 for USA and E.2 for Canada. Similarly, 60 clusters are found containing 38 different events for hybrid event detection, which can be seen in Table F.1 for USA and Table F.2 for Canada. The summary of the clustering based and hybrid event detection results are shown in Table 6.8.

With the numeric results listed in Table 6.7 and Table 6.8 we may calculate their precision, recall and f-measure values with equations given in section 6.5. The results are shown in 6.9, we see that in overall hybrid event detection method has the highest f-measure. Hybrid event detection method has the highest f-measure for the USA,

Table 6.8: Results of Clustering-Based and Hybrid Event Detection Method

Method	Country	Total Cluster #	Event Cluster #	Detected Event #	Undetected Event #
Clustering	USA	83	39	15	26
Clustering	Canada	57	20	10	9
Clustering	All	140	59	25	35
Hybrid	USA	55	34	25	16
Hybrid	Canada	5	4	3	16
Hybrid	All	60	38	28	32

while clustering-based and keyword-based methods have the highest f-measure for Canada.

Table 6.9: Accuracy of Event Detection Methods

Method	Country	Precision	Recall	F-measure
Keyword	USA	45%	73%	**56%**
Keyword	Canada	63%	32%	**42%**
Keyword	All	46%	60%	**52%**
Clustering	USA	47%	37%	41%
Clustering	Canada	35%	53%	**42%**
Clustering	All	42%	42%	42%
Hybrid	USA	62%	61%	**61%**
Hybrid	Canada	80%	16%	26%
Hybrid	All	63%	47%	**54%**

6.7.2 Accuracy Comparison with the Previous Solution

We have compared accuracy results on the previous sub-section. Now we can compare our results with the previous solution [7] and [8]. To make a decent comparison, first, we needed to re-calculate the result of the previous solution with the extended

ground truth events listed in Table B.2.

After finding the accuracy results of the previous solution for keyword-based event detection method, found on Table 6.10, we see that the number of detected events are the same as our solution for both USA and Canada. Therefore, we have the same recall values for both solutions which are 73% for the USA, 32% for Canada and 60% for overall results. The difference between the result of the two solutions is the number of total keywords detected. Hence, the determining factor is precision and f-measure. By looking at these factors, we may say that both the previous solution and this one have very similar accuracy performance. F-measure for USA results is increased from 54% to 56% and Canada results are increased from 41% to 42%. In overall, f-measure is slightly increased from 51% to 52%.

Table 6.10: Re-calculated Results of the Previous Keyword-Based Event Detection Method

Country	Bursty Keyword #	Keyword # Matching an Event	Detected Event #	Undetected Event #
USA	220	95	30	11
Canada	17	10	6	13
All	237	105	36	24

When we mapped the result of the previous solution with extended ground truth events, we see that none of the clusters can be mapped to new ground truth events for clustering-based and hybrid event detection methods. Therefore we may summarize the result of the previous solution as stated in Table 6.11.

After calculating the accuracy metrics, Table 6.12 is constructed. As seen on this table, f-measure of clustering-based event detection has decreased for the USA and overall results, while f-measure for Canada has increased from 33% to 42%.

A significant improvement can be seen in hybrid event detection results. While previous solution and our solution has very similar accuracy for Canada, f-measure of USA has increased from 48% to 61%. Similarly, overall f-measure result is increased from 42% to 54%.

Table 6.11: Re-calculated Results of Previous Clustering-Based and Hybrid Event Detection Method

Method	Country	Total Cluster #	Event Cluster #	Detected Event #	Undetected Event #
Clustering	USA	74	39	20	21
Clustering	Canada	7	5	4	15
Clustering	All	81	44	24	36
Hybrid	USA	87	53	16	25
Hybrid	Canada	3	3	3	16
Hybrid	All	90	56	19	41

6.8 Event Detection Time Performance

In this section, we'll show and compare time performance of event detection methods explained in 5.4, 5.5 and 5.6.

While comparing non-distributed and distributed implementations, we calculated *speedup* by their duration with the following equation:

$$speedup = \frac{t_{old}}{t_{new}} \tag{65}$$

Where,

- t is the runtime duration

6.8.1 Keyword-Based Event Detection Time Performance

To measure and evaluate the time performance of keyword based event detection we have run non-distributed configuration and different distributed configurations with USA, Canada and mixed dataset.

Table 6.12: Accuracy of the Previous Event Detection Methods

Method	Country	Precision	Recall	F-measure
Keyword	USA	43%	73%	54%
Keyword	Canada	59%	32%	41%
Keyword	All	44%	60%	51%
Clustering	USA	53%	49%	51%
Clustering	Canada	71%	21%	33%
Clustering	All	54%	40%	46%
Hybrid	USA	61%	39%	48%
Hybrid	Canada	100%	16%	27%
Hybrid	All	62%	32%	42%

6.8.1.1 Non-Distributed Configuration

For this experiment, we have run the benchmark with the configuration explained in Section 5.4.1.

To validate our non-distributed implementation, we first run our experiment with data set of each country separately. When we run experiment for only Canada, the benchmark processed whole data within **01:47** minutes, while data set of USA is processed in **07:25** minutes. This result is expected since the size of dataset is different for each country.

Later, we have run the experiment with whole data set by expecting it to be last about the sum of the first two experiment, and it did. The experiment with whole data set processed in **09:33** minutes, which is nearly the sum of the duration of the experiments with individual countries.

Comparing the output of these 3 experiments, experiment with USA data set has **196** keywords as output while experiment with Canada data set has **16** keywords. Like the duration, the output of experiment with whole data set gave the union of the outputs of country data set experiments.

6.8.1.2 Distributed Configuration

For distributed configuration, we have tried different numbers of procedure instances explained in Section 5.4.2.

First, we run different experiments for country datasets separately. We got exactly the same accuracy results with non-distributed configuration. Table 6.13 shows the number of different procedures and their time performance. We experimented using 1, 2 and 4 instances of Tokenizer Procedure and 1 and 2 instances of Word Count Procedure for Canada, USA and mixed dataset. Since Burst Detector procedure can only distributed by country we run 2 instances only on mixed dataset. As a result we got maximum speedup by **206%** when we used 1 instance for each procedure. Splitting and merging the data introduced more overhead than processing it together.

Table 6.13: Keyword-Based Event Detection Speedup from Non-Distributed to Distributed Configuration

Data Set	Tokenizer	Word Count	Burst Detector	Duration	Speedup
Canada	1	1	1	0:57	**187%**
Canada	2	1	1	1:03	169%
Canada	4	1	1	1:20	133%
Canada	1	2	1	1:09	155%
USA	1	1	1	3:57	187%
USA	2	1	1	3:55	**189%**
USA	4	1	1	4:34	162%
USA	1	2	1	4:22	169%
All	1	1	1	4:38	**206%**
All	2	1	1	4:44	201%
All	4	1	1	5:52	162%
All	1	2	1	5:08	186%
All	1	1	2	4:53	195%

6.8.2 Clustering-Based Event Detection Time Performance

We applied the same methodology that we used on Section 6.8.1 to evaluate time performance of clustering based event detection implementation. First we have run USA and Canada data set separately and after run the whole data set with different configurations. Later we examine the results and the performance improvement between distributed and non-distributed implementations.

6.8.2.1 Non-Distributed Configuration

First we run the non distributed implementation defined on Section 5.5.1 to create a base results to evaluate the speedup value of distributed implementations.

When we run this configuration with only Canada data set, the benchmark have completed in **3:00** minutes, while it lasted **1:34:28** hours with USA data set. Unlike the keyword based event detection method, the duration of the benchmarks are not proportional with the size of the data set since the time complexity of this implementation grows exponentially.

When we run the whole data set, we see that the duration is **1:49:46** hours which is just a bit longer that the sum of the duration of USA and Canada data sets. This is expected since there is only one partition to compute the data sets of both countries.

6.8.2.2 Distributed Configuration

For distributed configuration, we experimented different number of procedure instance explained in Section 5.5.2.

Similar to keyword-based event detection experiments, we first run different experiments for country datasets separately. This time the only time we got the same result with non-distributed configuration is when we used one instance for each procedure. For other number of procedures, we needed to change the clustering parameters to get similar results. We experimented 1, 2, 4 and 8 instances for Local Clustering Procedure. Since Cluster Merge and Event Detection Procedures can be distributed

by country, we only used at most 2 instances. Table 6.14 shows the performance and speedup results for different number of procedure instances. Canada configuration starts to slow down if it has more than 2 Local Clustering Procedures while USA gets the best speedup with 8. As a result when we select 10 Local Clustering Procedure, 8 for USA and 2 for Canada, and one Cluster Merge and Event Detector Procedure for each country, we got **222%** speedup.

Table 6.14: Clustering-Based Event Detection Speedup from Non-Distributed to Distributed Configuration

Data Set	Local Clustering	Cluster Merge	Event Detection	Duration	Speedup
Canada	1	1	1	2:39	113%
Canada	2	1	1	1:32	**195%**
Canada	4	1	1	1:57	153%
Canada	8	1	1	2:41	111%
USA	1	1	1	1:46:36	90%
USA	2	1	1	1:24:17	114%
USA	4	1	1	52:46	182%
USA	8	1	1	40:14	**239%**
All	1	1	1	1:51:49	86%
All	10	2	2	43:24	**222%**

6.8.3 Hybrid Event Detection Time Performance

To measure time performance of hybrid event detection performance, we first run non distributed configuration. When picking distributed configuration, for keyword-based event detection phase, we used 1 instance for each procedure. Since we are using tweets having only bursty keywords for clustering based event detection, we also used 1 instance for clustering-based event detection phase. Because there are only 32602 tweets having bursty keywords. As the dataset gets smaller, increasing number of procedure instances decreases the time performance. We got exactly the same results on both non-distributed and distributed configuration.

When we run non-distributed configuration, the experiment lasted 30:27 minutes while distributed configuration lasted 28:12 minutes. With this results, we had 108% speedup on hybrid event detection method. The reason we have a lower speedup with hybrid event detection method when compared to keyword based and clustering based is that our bottleneck is Tweet Filter Procedure and we could not manage distribute it since it requires to be in same partition with Feed Procedure.

6.8.4 Comparison Among Event Detection Methods

Running all of the experiments, we may now compare the time performance of our event detection implementations. In this section, we will only compare the result of running the whole data set, not the countries separately as we did in previous sections. Table 6.15 is constructed by these results.

Table 6.15: Time Performance Comparison of Event Detection Methods

Event Detection Method	Duration	Tweets per Second	Rounds per Minute
Keyword Based	4:38	46700	362
Clustering Based	43:24	5000	39
Hybrid	28:12	7650	59

As we may see on Table 6.15, the fastest event detection method is keyword based event detection method. It processes **46700** tweets per second which means it consumes **362** rounds or **36:12:00** hours of tweets in a minute.

Clustering-based event detection method is the slowest method. It processes **5000** tweets per seconds, consuming **39** rounds or **3:54:00** hours of tweets in a minute.

The hybrid event detection method is the second fast yet the most accurate event detection method. It processes **7650** tweets per seconds, consuming **59** rounds or **6:54:00** hours of tweets in a minute.

6.8.5 Comparison with the Previous Solution

We can now proceed to compare our results with the previous solution [7]. The previous solution is implemented with Apache Storm using the Cassandra database. The timing results are given using approximate hours. Therefore we will give speedup values approximately. Table 6.16 consists of timing results of our solution, previous solution and the speedup values. Please note that the experiments of the previous solution have run on a different computer, which has 3.2 GHz Intel Core i5 CPU with 4 threads and 16GB of RAM. It has slightly more CPU clock speed and half of our CPU threads and RAM. Since the previous solution relies on disk database, the amount of RAM would not affect the run-time. Therefore the speedup values will be slightly lower when all experiments are run on the same setup.

Table 6.16: Time Performance Comparison with the Previous Solution

Event Detection Method	Previous Solution	Our Solution	Speedup
Keyword Based	$\sim$ 3:00:00	4:38	$\sim$ **3884%**
Clustering Based	$\sim$ 11:00:00	43:24	$\sim$ **1520%**
Hybrid	$\sim$ 3:30:00	28:12	$\sim$ **744%**

Keyword-based event detection is the fastest event detection in previous solution like our solution. It lasted approximately **3:00:00** hours. Running our keyword-based event detection in **4:38** minutes, makes our implementation having **3884%** speedup, or almost **39** times faster.

Clustering-based event detection is the slowest event detection in previous solution. It lasts about **11:00:00** hours. In our solution it lasts **43:24** minutes. This results makes our solution to have **1520%** speedup, or **15** times faster.

When it comes to hybrid event detection method, we see that its duration is less than the clustering-based event detection method and much more than the keyword-based event detection method. Lasting **3:30:00** hours on previous solution makes our solution to have **744%** speedup, or **7** times faster.

CHAPTER 7

CONCLUSION

In this work, we aimed to enhance the time and accuracy performance of online event detection methods proposed by Sahin [7] by using a transactional online processing system, S-Store. We re-modelled the proposed methods to fit S-Store and analyzed the results in terms of accuracy and time.

To enabling a fair comparison, the same data set is used as the data set used by the previous solution. This data set is collected from twitter for a week and divided into windows, called rounds. The events are detected at the end of each round. We have streamed the same data set to observe accuracy and time performance for each method proposed.

We have compared the results of experiments through time and accuracy. For accuracy, although we got similar performance with the previous solution for keyword based and clustering based, we got better results for hybrid event detection method. When we compare time performance, we observed a dramatic decrease in run-time for all event detection methods.

For the future, there are 3 main works may be addressed. First, to enhance accuracy, new parameters can be added and tuned for all event detection methods. Second, time performance of hybrid event detection can be enhanced by experimenting new distributed setup. And lastly, the whole system may be distributed among different physical devices and use twitter API to work in a real-world environment.

REFERENCES

[1] @TwitterIR, "Q1 2019 letter to shareholders." https://s22.q4cdn.com/826641620/files/doc_financials/2019/q1/Q1-2019-Shareholder-Letter.pdf, April 2019.

[2] S. A. Mughal, "Twitter by the numbers (2019): Stats, demographics & fun facts." https://www.omnicoreagency.com/twitter-statistics/, Jan 2019.

[3] M. Cordeiro and J. Gama, "Online social networks event detection: A survey," in *Solving Large Scale Learning Tasks. Challenges and Algorithms of Lecture Notes in Computer Science* (S. Michaelis, N. Piatkowski, , and M. Stolpe, eds.), vol. 9580, p. 1–41, Cham: Springer.

[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)*, p. 2219–2222.

[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 *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, DEBS '16, (New York, NY, USA), pp. 289–292, ACM, 2016.

[7] O. C. Sahin, "Online event detection from streaming data," Master's thesis, Middle East Technical University, 5 2018.

[8] O. C. Sahin, P. Karagoz, and N. Tatbul, "Streaming event detection in microblogs: Balancing accuracy and performance," in *Web Engineering - 19th International Conference, ICWE 2019, Daejeon, South Korea, June 11-14, 2019,*

Proceedings (M. Bakaev, F. Frasincar, and I. Ko, eds.), vol. 11496 of *Lecture Notes in Computer Science*, pp. 123–138, Springer, 2019.

[9] O. C. Sahin, N. Tatbul, and P. Karagoz, "Using a stream processing platform for event detection: Advantages and limitations," *Turkish National Software Engineering Symposium*, 2018.

[10] U. Cetintemel, J. Du, T. Kraska, S. Madden, D. Maier, J. Meehan, A. Pavlo, M. Stonebraker, E. Sutherland, N. Tatbul, K. Tufte, H. Wang, and S. B. Zdonik, "S-store: a streaming newsql system for big velocity applications," *very large data bases*, vol. 7, no. 13, pp. 1633–1636, 2014.

[11] N. Tatbul, "S-store: Real-time analytics meets transaction processing." `http://istc-bigdata.org/index.php/s-store-real-time-analytics-meets-transaction-processing/`, 2014.

[12] J. Meehan, "S-store: A big-velocity database system." `http://istc-bigdata.org/index.php/s-store-a-big-velocity-database-system/`, 2014.

[13] J. Meehan, N. Tatbul, S. B. Zdonik, C. Aslantas, U. Cetintemel, J. Du, T. Kraska, S. Madden, D. Maier, A. Pavlo, M. Stonebraker, K. Tufte, and H. Wang, "S-store: streaming meets transaction processing," *Very Large Data Bases*, vol. 8, no. 13, pp. 2134–2145, 2015.

[14] H. Becker, M. Naaman, and L. Gravano, "Beyond trending topics: Real-world event identification on twitter," in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[15] M. Walther and M. Kaisser, "Geo-spatial event detection in the twitter stream," in *ECIR'13 Proceedings of the 35th European conference on Advances in Information Retrieval*, pp. 356–367, 2013.

[16] 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 *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 2541–2544, 2011.

[17] L. M. Aiello, G. Petkos, C. J. Martín, D. Corney, S. Papadopoulos, R. Skraba, A. Göker, I. Kompatsiaris, and A. Jaimes, "Sensing trending topics in twitter," *IEEE Transactions on Multimedia*, vol. 15, no. 6, pp. 1268–1282, 2013.

[18] C. Li, A. Sun, and A. Datta, "Twevent: segment-based event detection from tweets," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 155–164, 2012.

[19] S. Petrovic, 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, 2010.

[20] M. Mathioudakis and N. Koudas, "Twittermonitor: trend detection over the twitter stream," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 1155–1158, 2010.

[21] 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, 2009.

[22] M. Cataldi, L. D. Caro, and C. Schifanella, "Emerging topic detection on twitter based on temporal and social terms evaluation," in *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, p. 4, 2010.

[23] A. Ritter, O. Etzioni, and S. Clark, "Open domain event extraction from twitter," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1104–1112, 2012.

[24] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang, "Tedas: A twitter-based event detection and analysis system," in *2012 IEEE 28th International Conference on Data Engineering*, pp. 1273–1276, 2012.

[25] H. Abdelhaq, C. Sengstock, and M. Gertz, "Eventweet: online localized event detection from twitter," *Very Large Data Bases*, vol. 6, no. 12, pp. 1326–1329, 2013.

[26] M. Cordeiro and R. Frias, "Twitter event detection: combining wavelet analysis and topic inference summarization," 2011.

[27] J. Weng and B.-S. Lee, "Event detection in twitter," in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[28] R. Lee and K. Sumiya, "Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection," in *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, pp. 1–10, 2010.

[29] E. D'Andrea, P. Ducange, B. Lazzerini, and F. Marcelloni, "Real-time detection of traffic from twitter stream analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2269–2283, 2015.

[30] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*, pp. 851–860, 2010.

Appendix A

STREAM AND TABLE DEFINITIONS

A.1 Common Streams for All Event Detection Methods

A.1.1 Stream: tweets_s

tweets_s stream is used for streaming structured tweet data to next procedure. All of the event detection methods uses this stream.

```
CREATE STREAM tweets_s (
  tweet_id  BIGINT          NOT NULL,
  country   VARCHAR(3)      NOT NULL,
  round     INT             NOT NULL,
  tweet     VARCHAR (2048)  NOT NULL,
  batch_id  BIGINT          NOT NULL,
  part_id   INT             NOT NULL
);
```

A.2 Streams for Keyword-Based Event Detection

A.2.1 Stream: tokenized_word_counts_s

tokenized_word_counts_s stream is used for streaming word counts gathered from a partition for each round of each country. These word counts are intermediate results since it should be merged with the word counts of other partitions.

```
CREATE STREAM tokenized_word_counts_s (
  country     VARCHAR(3)      NOT NULL,
  round       INT             NOT NULL,
  word        VARCHAR(1024)   NOT NULL,
  word_count  INT             NOT NULL,
  batch_id    BIGINT          NOT NULL,
  part_id     INT             NOT NULL
);
```

A.2.2 Stream: word_counts_s

word_counts_s stream is used for streaming merged word counts of tokenized_word_conts_s gathered from different partitions.

```
CREATE STREAM word_counts_s (
  country     VARCHAR(3)      NOT NULL,
  round       INT             NOT NULL,
  word        VARCHAR(1024)   NOT NULL,
  word_count  INT             NOT NULL,
  batch_id    BIGINT          NOT NULL,
  part_id     INT             NOT NULL
);
```

A.2.3 Stream: burst_words_s

burst_words_s stream is used for streaming detected bursty words to next procedure.

```
CREATE STREAM burst_words_s (
  country   VARCHAR(3)      NOT NULL,
  round     INT             NOT NULL,
  word      VARCHAR(1024)   NOT NULL,
  batch_id  BIGINT          NOT NULL,
  part_id   INT             NOT NULL
```

```
);
```

A.3 Streams of Clustering-Based Event Detection

A.3.1 Stream: local_cluster_results_s

local_cluster_results_s is used for streaming clusters has been constructed in a single partitions to next procedure. This is intermediate clusters that needs to be merged with local clusters of other partitions.

```
CREATE STREAM local_cluster_results_s (
  country           VARCHAR(3)      NOT NULL,
  round             INT             NOT NULL,
  local_cluster_id  INT             NOT NULL,
  tweet_id          BIGINT          NOT NULL,
  tweet_text        VARCHAR(2048)   NOT NULL,
  batch_id          BIGINT          NOT NULL,
  part_id           INT             NOT NULL
);
```

A.3.2 Stream: merged_cluster_results_s

merged_cluster_results_s is used for streaming merged local clusters to next procedure.

```
CREATE STREAM merged_cluster_results_s (
  country           VARCHAR(3)      NOT NULL,
  round             INT             NOT NULL,
  cluster_id        INT             NOT NULL,
  tweet_id          BIGINT          NOT NULL,
  tweet_text        VARCHAR(2048)   NOT NULL,
  batch_id          BIGINT          NOT NULL,
```

```
  part_id           INT             NOT NULL
);
```

A.4 Streams and Tables of Hybrid Event Detection

A.4.1 Table: tweet_words

tweet_words table is used as a look-up table holding tweet id and word mapping. It is used when filtering tweets having bursty keywords. It is indexed by country, round, word and part_id since it is called frequently.

```
CREATE TABLE tweet_words (
  tweet_id  BIGINT          NOT NULL,
  country   VARCHAR(3)      NOT NULL,
  round     INT             NOT NULL,
  word      VARCHAR(1024)   NOT NULL,
  tweet     VARCHAR (2048)  NOT NULL,
  part_id   INT             NOT NULL
);

CREATE INDEX tweet_words_idx
  ON tweet_words(country, round, word, part_id);
```

A.4.2 Stream: filtered_tweets_s

filtered_tweets_s stream is used for streaming tweets having bursty keywords to next procedure. It has the same structure as tweets_s stream.

```
CREATE STREAM filtered_tweets_s (
  tweet_id  BIGINT          NOT NULL,
  country   VARCHAR(3)      NOT NULL,
  round     INT             NOT NULL,
```

```
  tweet     VARCHAR (2048)  NOT NULL,
  batch_id  BIGINT          NOT NULL,
  part_id   INT             NOT NULL
);
```

Appendix B

GROUND TRUTH EVENTS

Table B.1: Existing Ground Truth Events

#	Date (2016)	Event
1.	May 31	Klay Thompson breaks NBA record by 11 3-pointers in NBA Western Conference Finals
2.	May 31	Draymond Green kicks Steven Adams in groin in NBA Western Conference Finals
3.	May 31	Steven Adams called Warrior's guards as quick little monkeys
4.	May 31	Penguins won NHL Stanley Cup by defeating Sharks on final game
5.	May 31	Golden State Warriors complete comeback to reach NBA Finals
6.	May 31 - June 2	2016 NBA champion predictions
7.	May 31	Trailer of "The Guest" movie is released
8.	June 1 - June 6	FDB Ringtone by Ztro is released
9.	June 3	National Anthem is sung by John Legend in NBA Finals
10.	June 3	Stephen Curry fouls Tristian Thompson in NBA Finals
11.	June 3	Kevin Love gets an offensive foul in NBA Finals
12.	June 3	Matthew Dellavedova fouls on Andre Iguodala in NBA Finals
13.	June 3	Game I of NBA Finals was the lowest scoring game for Curry and Thompson. Warriors won the game easily.
14.	June 3	Shaun Livingstone scored 20 points in NBA Finals.
15.	June 4	Muhammad Ali died
16.	June 5	Brock Lesnar is returns to NFC.
17.	June 6	Carlos Santana sung National Anthem in NBA Finals.
18.	June 6	Lebron James Travels travels but uncalled on Game II of NBA Finals.
19.	June 6	Cavaliers matched with Hawks on playoffs.
20.	June 6	Kyle Jenner's Twitter acount is hacked.

Table B.2: New Events Added to Ground Truth

#	Date (2016)	Event
21.	May 31	Bryan Rust scores a goal and had a great performance on NHL playoffs.
22.	May 31	Steph Curry crosses Andre Robertson with ankle breaker
23.	May 31	Curry holds his knee after a layup
24.	May 31	National Donut Day
25.	May 31	Shaun Livingstone dunks over 3 players in NBA Western Conference Finals
26.	May 31	Speights had a bad performance on NBA Western Conference Finals
27.	June 2	Mookie Betts became the first leadoff hitter in franchise history to hit three home runs in a single game
28.	June 3	Katy Parry's Twitter account is hacked
29.	June 3	Leandro Barbosa scored 11 points coming from bench in the first game of the NBA Finals.
30.	June 3	Golden State Warriors' coach Steve Kerr lost his cool and shattered his whiteboard with his marker.
31.	June 3	Gang member Denzel Barbosa sentenced 45 years in prison for shooting his rival.
32.	June 3	Draymont Green flops in NBA Finals.
33.	June 3	Anderson Varejao flops in NBA Finals.
34.	June 3	Kobe Bryant plays on Ghostbusters Commercial
35.	June 4	USA lost to Colombia 2-0 where James Rodriguez scores the 2nd goal.
36.	June 5	Braun scores a inevitable goal against Murry in NHL game.
37.	June 5	Conor McGregor vs Nate Diaz match announced officially.
38.	June 6	Iman Shumpert gets new hairstyle.
39.	June 6	Kevin love suffered a concussion after elbow to the head by Harrison Barnes in the second match of the NBA finals.
40.	June 6	Game of Thrones S6E7 The Hound is released.
41.	June 6	Verizon released new ad.
42.	June 6	Mexico beat Uruguay 3-1 in Copa America Group C
43.	June 6	Miss USA event and performance of Backstreet Boys

Appendix C

PARAMETER TUNING RESULTS

Table C.1: Parameter Tuning Results of All Configurations for USA Clustering Based Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.6	80	75	12	0.780	0.293	0.426
0.5	120	125	15	0.470	0.366	0.411
0.5	120	100	15	0.447	0.366	0.402
0.5	120	75	15	0.447	0.366	0.402
0.5	120	150	12	0.633	0.293	0.400
0.6	60	75	14	0.448	0.341	0.388
0.6	80	100	10	0.926	0.244	0.386
0.55	120	125	11	0.633	0.268	0.377
0.55	120	100	11	0.627	0.268	0.376
0.55	120	75	11	0.627	0.268	0.376
0.6	60	100	13	0.456	0.317	0.374
0.55	100	100	12	0.500	0.293	0.369
0.55	100	75	12	0.500	0.293	0.369
0.5	100	125	16	0.348	0.390	0.368
0.55	100	125	11	0.577	0.268	0.366
0.65	80	75	9	0.960	0.220	0.357
0.55	80	100	13	0.407	0.317	0.357
0.5	100	150	13	0.398	0.317	0.353
0.55	80	125	12	0.438	0.293	0.351

Continued C.1: Parameter Tuning Results of All Configurations for USA Clustering Based Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.65	60	75	9	0.857	0.220	0.350
0.55	80	75	16	0.283	0.390	0.328
0.6	60	150	8	0.913	0.195	0.322
0.5	100	100	19	0.240	0.463	0.316
0.5	100	75	19	0.240	0.463	0.316
0.6	60	125	10	0.435	0.244	0.313
0.6	120	100	7	1.000	0.171	0.292
0.6	120	75	7	1.000	0.171	0.292
0.6	100	100	7	1.000	0.171	0.292
0.6	100	75	7	1.000	0.171	0.292
0.6	80	125	7	0.909	0.171	0.287
0.65	60	100	7	0.885	0.171	0.286
0.65	60	125	7	0.875	0.171	0.286
0.55	120	150	7	0.710	0.171	0.275
0.55	100	150	7	0.583	0.171	0.264
0.55	80	150	8	0.404	0.195	0.263
0.6	120	125	6	1.000	0.146	0.255
0.6	80	150	6	0.889	0.146	0.251
0.55	60	150	10	0.256	0.244	0.250
0.55	60	125	17	0.172	0.415	0.243
0.55	60	75	19	0.152	0.463	0.229
0.55	60	100	17	0.158	0.415	0.229
0.65	100	125	5	1.000	0.122	0.217
0.65	100	100	5	1.000	0.122	0.217
0.65	100	75	5	1.000	0.122	0.217
0.6	120	150	5	1.000	0.122	0.217
0.65	60	150	5	1.000	0.122	0.217
0.65	80	125	5	1.000	0.122	0.217

Continued C.1: Parameter Tuning Results of All Configurations for USA Clustering Based Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.65	80	100	5	1.000	0.122	0.217
0.6	100	150	5	1.000	0.122	0.217
0.6	100	125	5	1.000	0.122	0.217
0.5	80	125	16	0.148	0.390	0.215
0.5	80	100	19	0.137	0.463	0.212
0.5	80	150	13	0.138	0.317	0.192
0.65	100	150	4	1.000	0.098	0.178
0.65	80	150	4	1.000	0.098	0.178
0.5	80	75	22	0.102	0.537	0.171
0.65	120	150	3	1.000	0.073	0.136
0.65	120	125	3	1.000	0.073	0.136
0.65	120	100	3	1.000	0.073	0.136
0.65	120	75	3	1.000	0.073	0.136

Table C.2: Parameter Tuning Results of All Configurations for Canada Clustering Based Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.5	30	31	10	0.351	0.526	0.421
0.5	25	31	11	0.318	0.579	0.411
0.5	30	25	10	0.333	0.526	0.408
0.5	30	18	10	0.333	0.526	0.408
0.5	25	25	13	0.266	0.684	0.383
0.5	25	18	13	0.266	0.684	0.383
0.55	20	18	12	0.255	0.632	0.363
0.6	20	18	8	0.304	0.421	0.353
0.6	30	31	5	0.500	0.263	0.345
0.6	30	25	5	0.500	0.263	0.345
0.6	30	18	5	0.500	0.263	0.345
0.55	25	25	7	0.321	0.368	0.343
0.55	25	18	7	0.321	0.368	0.343
0.6	15	18	9	0.260	0.474	0.336
0.55	30	25	5	0.462	0.263	0.335
0.55	30	18	5	0.462	0.263	0.335
0.55	30	31	5	0.458	0.263	0.334
0.5	25	37	7	0.300	0.368	0.331
0.55	20	25	7	0.290	0.368	0.325
0.5	30	37	6	0.324	0.316	0.320
0.55	15	25	8	0.252	0.421	0.316
0.55	10	31	9	0.232	0.474	0.311
0.55	15	18	11	0.212	0.579	0.310
0.6	30	37	4	0.583	0.211	0.309
0.55	15	31	7	0.263	0.368	0.307
0.6	25	31	5	0.364	0.263	0.305
0.55	10	25	11	0.204	0.579	0.302
0.55	30	37	4	0.533	0.211	0.302

Continued C.2: Parameter Tuning Results of All Configurations for Canada Clustering Based Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.6	15	31	6	0.286	0.316	0.300
0.55	25	31	5	0.333	0.263	0.294
0.6	20	25	5	0.333	0.263	0.294
0.55	10	37	6	0.271	0.316	0.292
0.6	25	25	5	0.324	0.263	0.291
0.6	25	18	5	0.324	0.263	0.291
0.6	25	37	4	0.467	0.211	0.290
0.6	20	31	5	0.321	0.263	0.289
0.6	15	25	6	0.267	0.316	0.289
0.55	25	37	4	0.391	0.211	0.274
0.55	20	31	5	0.282	0.263	0.272
0.55	15	37	5	0.271	0.263	0.267
0.55	10	18	15	0.159	0.789	0.265
0.65	20	37	4	0.348	0.211	0.262
0.6	20	37	4	0.348	0.211	0.262
0.65	20	18	5	0.258	0.263	0.261
0.5	20	25	11	0.164	0.579	0.255
0.65	15	37	4	0.321	0.211	0.254
0.65	15	31	4	0.310	0.211	0.251
0.65	20	31	4	0.308	0.211	0.250
0.5	20	31	8	0.177	0.421	0.249
0.65	15	18	5	0.235	0.263	0.248
0.6	15	37	4	0.300	0.211	0.247
0.55	20	37	4	0.294	0.211	0.245
0.65	30	37	3	0.545	0.158	0.245
0.65	20	25	4	0.282	0.211	0.241
0.65	15	25	4	0.269	0.211	0.236
0.65	30	31	3	0.412	0.158	0.228

Continued C.2: Parameter Tuning Results of All Configurations for Canada Clustering Based Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.65	30	25	3	0.412	0.158	0.228
0.65	30	18	3	0.412	0.158	0.228
0.65	25	37	3	0.400	0.158	0.226
0.5	20	18	14	0.133	0.737	0.226
0.65	25	31	3	0.300	0.158	0.207
0.65	25	25	3	0.257	0.158	0.196
0.65	25	18	3	0.257	0.158	0.196
0.5	10	37	9	0.117	0.474	0.188
0.5	20	37	5	0.145	0.263	0.187
0.5	10	31	13	0.105	0.684	0.182
0.5	10	25	14	0.085	0.737	0.152
0.5	10	18	15	0.072	0.789	0.132

Table C.3: Parameter Tuning Results of All Configurations for USA Hybrid Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.5	30	30	25	0.618	0.610	0.614
0.5	20	30	23	0.611	0.561	0.585
0.5	40	30	20	0.591	0.488	0.534
0.55	30	30	18	0.578	0.439	0.499
0.55	20	30	18	0.565	0.439	0.494
0.5	40	50	16	0.643	0.390	0.486
0.5	30	50	16	0.621	0.390	0.479
0.5	50	50	15	0.593	0.366	0.452
0.5	50	30	15	0.593	0.366	0.452
0.5	40	60	13	0.667	0.317	0.430
0.5	20	50	14	0.571	0.341	0.427
0.5	30	60	13	0.609	0.317	0.417
0.5	50	60	12	0.632	0.293	0.400
0.6	20	30	13	0.541	0.317	0.400
0.55	40	30	13	0.533	0.317	0.398
0.5	20	60	12	0.591	0.293	0.391
0.6	30	30	12	0.571	0.293	0.387
0.5	30	75	11	0.688	0.268	0.386
0.5	40	75	11	0.688	0.268	0.386
0.55	30	50	12	0.542	0.293	0.380
0.65	30	30	11	0.613	0.268	0.373
0.5	60	60	11	0.611	0.268	0.373
0.5	60	50	11	0.611	0.268	0.373
0.5	60	30	11	0.611	0.268	0.373
0.6	40	30	11	0.583	0.268	0.368
0.65	20	30	11	0.581	0.268	0.367
0.55	30	60	11	0.579	0.268	0.367
0.55	40	50	11	0.571	0.268	0.365

Continued C.3: Parameter Tuning Results of All Configurations for USA Hybrid Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.55	50	50	11	0.571	0.268	0.365
0.55	50	30	11	0.571	0.268	0.365
0.65	40	30	10	0.700	0.244	0.362
0.6	30	50	11	0.550	0.268	0.361
0.5	50	75	10	0.667	0.244	0.357
0.55	60	60	10	0.667	0.244	0.357
0.55	60	50	10	0.667	0.244	0.357
0.55	60	30	10	0.667	0.244	0.357
0.6	50	50	10	0.647	0.244	0.354
0.6	50	30	10	0.647	0.244	0.354
0.55	20	50	11	0.520	0.268	0.354
0.5	20	75	10	0.625	0.244	0.351
0.55	50	60	10	0.625	0.244	0.351
0.6	40	50	10	0.611	0.244	0.349
0.6	60	60	9	0.833	0.220	0.347
0.6	60	50	9	0.833	0.220	0.347
0.6	60	30	9	0.833	0.220	0.347
0.55	40	60	10	0.588	0.244	0.345
0.6	20	50	11	0.478	0.268	0.344
0.6	50	60	9	0.769	0.220	0.342
0.6	40	60	9	0.769	0.220	0.342
0.55	20	60	10	0.550	0.244	0.338
0.5	60	75	9	0.643	0.220	0.327
0.6	30	60	9	0.600	0.220	0.321
0.65	50	50	8	0.833	0.195	0.316
0.65	50	30	8	0.833	0.195	0.316
0.65	40	50	8	0.833	0.195	0.316
0.55	60	75	8	0.800	0.195	0.314

Continued C.3: Parameter Tuning Results of All Configurations for USA Hybrid Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.55	50	75	8	0.727	0.195	0.308
0.65	30	50	8	0.714	0.195	0.307
0.6	20	60	9	0.500	0.220	0.305
0.55	40	75	8	0.667	0.195	0.302
0.55	20	75	8	0.615	0.195	0.296
0.55	30	75	8	0.615	0.195	0.296
0.65	20	50	8	0.579	0.195	0.292
0.65	50	75	7	1.000	0.171	0.292
0.6	60	75	7	1.000	0.171	0.292
0.65	40	75	7	1.000	0.171	0.292
0.65	50	60	7	0.900	0.171	0.287
0.65	40	60	7	0.900	0.171	0.287
0.6	50	75	7	0.889	0.171	0.286
0.6	40	75	7	0.889	0.171	0.286
0.65	20	75	7	0.800	0.171	0.281
0.65	30	75	7	0.800	0.171	0.281
0.6	30	75	7	0.778	0.171	0.280
0.65	30	60	7	0.750	0.171	0.278
0.65	20	60	7	0.667	0.171	0.272
0.6	20	75	7	0.636	0.171	0.269
0.65	60	75	6	1.000	0.146	0.255
0.65	60	60	6	0.889	0.146	0.251
0.65	60	50	6	0.889	0.146	0.251
0.65	60	30	6	0.889	0.146	0.251

Table C.4: Parameter Tuning Results of All Configurations for Canada Hybrid Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.5	7	7	3	0.833	0.158	0.265
0.5	5	7	3	0.833	0.158	0.265
0.55	7	7	3	0.800	0.158	0.264
0.55	5	7	3	0.800	0.158	0.264
0.5	7	18	3	0.750	0.158	0.261
0.5	7	15	3	0.750	0.158	0.261
0.5	7	12	3	0.750	0.158	0.261
0.5	5	18	3	0.750	0.158	0.261
0.5	5	15	3	0.750	0.158	0.261
0.5	5	12	3	0.750	0.158	0.261
0.55	10	15	3	0.750	0.158	0.261
0.55	10	12	3	0.750	0.158	0.261
0.55	10	7	3	0.750	0.158	0.261
0.55	15	15	3	0.750	0.158	0.261
0.55	15	12	3	0.750	0.158	0.261
0.55	15	7	3	0.750	0.158	0.261
0.55	12	15	3	0.750	0.158	0.261
0.55	12	12	3	0.750	0.158	0.261
0.55	12	7	3	0.750	0.158	0.261
0.55	7	15	3	0.750	0.158	0.261
0.55	7	12	3	0.750	0.158	0.261
0.55	5	15	3	0.750	0.158	0.261
0.55	5	12	3	0.750	0.158	0.261
0.5	12	18	3	0.750	0.158	0.261
0.5	12	15	3	0.750	0.158	0.261
0.5	12	12	3	0.750	0.158	0.261
0.5	12	7	3	0.750	0.158	0.261
0.5	15	18	3	0.750	0.158	0.261

Continued C.4: Parameter Tuning Results of All Configurations for Canada Hybrid Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.5	15	15	3	0.750	0.158	0.261
0.5	15	12	3	0.750	0.158	0.261
0.5	15	7	3	0.750	0.158	0.261
0.5	10	18	3	0.750	0.158	0.261
0.5	10	15	3	0.750	0.158	0.261
0.5	10	12	3	0.750	0.158	0.261
0.5	10	7	3	0.750	0.158	0.261
0.65	12	12	2	0.800	0.105	0.186
0.65	12	7	2	0.800	0.105	0.186
0.65	10	12	2	0.800	0.105	0.186
0.65	10	7	2	0.800	0.105	0.186
0.6	10	7	2	0.800	0.105	0.186
0.6	7	7	2	0.800	0.105	0.186
0.65	7	12	2	0.800	0.105	0.186
0.65	7	7	2	0.800	0.105	0.186
0.65	5	12	2	0.800	0.105	0.186
0.65	5	7	2	0.800	0.105	0.186
0.6	5	7	2	0.800	0.105	0.186
0.6	12	15	2	0.750	0.105	0.185
0.6	12	12	2	0.750	0.105	0.185
0.6	12	7	2	0.750	0.105	0.185
0.6	10	15	2	0.750	0.105	0.185
0.6	10	12	2	0.750	0.105	0.185
0.6	7	15	2	0.750	0.105	0.185
0.6	7	12	2	0.750	0.105	0.185
0.6	15	15	2	0.750	0.105	0.185
0.6	15	12	2	0.750	0.105	0.185
0.6	15	7	2	0.750	0.105	0.185

Continued C.4: Parameter Tuning Results of All Configurations for Canada Hybrid Event Detection

p_1	p_2	p_3	Event #	Precision	Recall	F-Measure
0.6	5	15	2	0.750	0.105	0.185
0.6	5	12	2	0.750	0.105	0.185
0.55	10	18	2	0.667	0.105	0.182
0.55	15	18	2	0.667	0.105	0.182
0.55	12	18	2	0.667	0.105	0.182
0.55	7	18	2	0.667	0.105	0.182
0.55	5	18	2	0.667	0.105	0.182
0.6	12	18	1	0.667	0.053	0.098
0.65	15	15	1	0.667	0.053	0.098
0.65	15	12	1	0.667	0.053	0.098
0.65	15	7	1	0.667	0.053	0.098
0.65	12	15	1	0.667	0.053	0.098
0.65	10	15	1	0.667	0.053	0.098
0.6	10	18	1	0.667	0.053	0.098
0.6	7	18	1	0.667	0.053	0.098
0.65	7	15	1	0.667	0.053	0.098
0.65	5	15	1	0.667	0.053	0.098
0.6	15	18	1	0.667	0.053	0.098
0.6	5	18	1	0.667	0.053	0.098
0.65	15	18	1	0.500	0.053	0.095
0.65	12	18	1	0.500	0.053	0.095
0.65	10	18	1	0.500	0.053	0.095
0.65	7	18	1	0.500	0.053	0.095
0.65	5	18	1	0.500	0.053	0.095

Appendix D

OUTPUTS OF KEYWORD-BASED EVENT DETECTION METHOD

Keywords indicating events are highlighted with **bold**.

Table D.1: Keyword Based Event Detection Outputs of USA

Date (2016)	Detected Keywords	Ground Truth
May 31, 03:06	karlie	-
May 31, 03:24	tommie	-
May 31, 03:42	**rust**, **rusty**	21
May 31, 03:48	**2-0**, **crosby**, **sheary**	4
May 31, 03:54	anthem, national	-
May 31, 04:06	**adam**, 2k17, **steven**, goldberg	3
May 31, 04:18	dion, waiter	-
May 31, 04:24	**ankle**, **roberson**	22
May 31, 04:36	**speight**	26
May 31, 04:54	marleau, ddt, **green**, **adam**, rko, **steven**, **dirty**, **draymond**	2
May 31, 05:00	ibaka, **steph**, **knee**, **layup**	23
May 31, 05:30	switch, tie	-
May 31, 05:36	**livingston**, **shaun**	25
May 31, 05:42	**andy**, **anderson**, barbosa, **varejao**	-
May 31, 05:48	**bonino**	4
May 31, 05:54	**speight**	26
May 31, 06:00	**roberson**	22

Continued D.1: Keyword Based Event Detection Outputs of USA

Date (2016)	Detected Keywords	Ground Truth
May 31, 06:18	ibaka, dagger	-
May 31, 06:24	**#nbafinals**	6
May 31, 07:06	11:11	-
May 31, 09:06	11:11	-
May 31, 14:48	#careerarc	-
Jun 01, 06:06	11:11	-
Jun 01, 06:18	seager	-
Jun 01, 07:06	11:11	-
Jun 01, 09:06	11:11	-
Jun 01, 14:48	#careerarc	-
Jun 01, 17:18	#careerarc, opening, #hiring, latest, recommend, click, #retail, #jobs, #job	-
Jun 01, 22:24	#diabetes, #diabetic, 1diabete, #type	-
Jun 01, 22:36	hip, hop	-
Jun 01, 23:24	#sales, #careerarc, opening, #hiring, latest, recommend, fit, #retail, #nursing, #hospitality, #jobs, #job	-
Jun 02, 02:06	#greta	-
Jun 02, 02:48	**mookie**	28
Jun 02, 03:48	lyft	-
Jun 02, 04:30	**hbk**, **bonino**, **kessel**	4
Jun 02, 05:42	braun	-
Jun 02, 05:48	#bucciovertimechallenge	-
Jun 02, 06:06	sheary	-
Jun 02, 07:06	11:11	-
Jun 02, 09:06	11:11	-
Jun 02, 14:30	**#nbafinalsvote**, #allin, **nba**, **cavalier**	6
Jun 02, 14:48	#careerarc	-
Jun 03, 04:00	**anthem**, **john**, **legend**	9
Jun 03, 04:12	**barne**, **harrison**	13

Continued D.1: Keyword Based Event Detection Outputs of USA

Date (2016)	Detected Keywords	Ground Truth
Jun 03, 04:18	jab, step, slip	15
Jun 03, 04:30	**kobe**, clock, **flop**	32, 36
Jun 03, 04:42	**barbosa**, **bench**	29
Jun 03, 04:54	**iggy**	12
Jun 03, 05:06	**andy**, **flop**, **varejao**	33
Jun 03, 05:36	**kerr**	30
Jun 03, 05:48	**delly**	12
Jun 03, 05:54	replay, technical	-
Jun 03, 06:00	**denzel**, **barbosa**	31
Jun 03, 06:06	**livingston**, **shaun**	14
Jun 03, 09:06	11:11	-
Jun 03, 14:48	#careerarc	-
Jun 04, 05:18	**2-0**, **jame**	35
Jun 04, 05:54	lyft	-
Jun 04, 06:06	11:11	-
Jun 04, 07:18	**#muhammadali**, **#rip**, **legend**, **peace**, **rip**, **r.i.p**, **muhammad**, **rest**, **ali**, **greatest**, **butterfly**, **alus**	15
Jun 04, 09:06	11:11	-
Jun 04, 14:48	#careerarc	-
Jun 04, 20:30	retweet	-
Jun 05, 00:24	**hack**	20
Jun 05, 03:30	lovejoy	-
Jun 05, 03:42	**braun**, **murray**	36
Jun 05, 05:30	ward	-
Jun 05, 05:54	#bucciovertimechallenge	-
Jun 05, 06:00	**diaz**, **mcgregor**	37
Jun 05, 06:06	11:11	-
Jun 05, 07:48	#andnew, bisping, bisp, michael	-
Jun 05, 08:30	#방탄소년단, #lovebts	-

Continued D.1: Keyword Based Event Detection Outputs of USA

Date (2016)	Detected Keywords	Ground Truth
Jun 05, 09:06	11:11	-
Jun 05, 14:48	#careerarc	-
Jun 06, 00:48	div, -rsb-, 06/05, #toronto, ave	-
Jun 06, 01:54	seager	-
Jun 06, 03:00	**anthem**, **santana**, **national**	17
Jun 06, 03:18	#westvirginia	-
Jun 06, 03:42	**shump**, **shumpert**	38
Jun 06, 03:54	**elbow**, **ref**, **kevin**	39
Jun 06, 04:00	**hound**	32
Jun 06, 04:18	answer	-
Jun 06, 04:30	**sprint**, **verizon**	41
Jun 06, 04:36	**vamo**, **guardado**	42
Jun 06, 04:42	**backstreet**	43
Jun 06, 04:48	**marquez**, travels, **rafa**	42
Jun 06, 04:54	**3-1**, **movement**	42
Jun 06, 05:00	barbosa	-
Jun 06, 06:06	11:11	-
Jun 06, 07:06	11:11	-
Jun 06, 09:06	11:11	-
Jun 06, 14:48	#careerarc	-
Jun 06, 18:54	**hack**	20
Jun 06, 23:36	-lsb-, div, -rsb-, 06/06, #toronto	-

Table D.2: Keyword Based Event Detection Outputs of Canada

Date (2016)	Detected Keywords	Ground Truth
May 31, 04:54	**green**, **draymond**, **adam**	2
May 31, 05:00	**curry**	23
May 31, 06:00	**curry**	22
May 31, 19:42	#agp	-
Jun 01, 08:00	20:20	-
Jun 02, 05:48	#bucciovertimechallenge	-
Jun 03, 04:42	**barbosa**	29
Jun 03, 06:06	**livingston**	14
Jun 04, 07:18	**rip**, **muhammad**, **ali**	15
Jun 04, 17:48	#agp	-
Jun 04, 20:24	#justshowupshow	-
Jun 05, 04:18	#mtvpopcd	-

Appendix E

OUTPUTS OF CLUSTERING-BASED EVENT DETECTION METHOD

Table E.1: Clustering-Based Detection Outputs of USA

Date (2016)	Tweet #	Detected Keywords	Ground Truth
May 31, 00:42	127	get:0.99	-
May 31, 01:24	253	get:1.00	-
May 31, 03:24	384	get:1.00	-
May 31, 04:00	134	game:0.98, time:0.14, warrior:0.11	-
May 31, 04:06	269	game:0.99	-
May 31, 04:48	293	klay:0.92, thompson:0.38	1
May 31, 04:54	890	draymond:0.61, green:0.59, adam:0.44, get:0.17, steven:0.13	2
May 31, 05:00	135	curry:0.98, make:0.11, layup:0.11	23
May 31, 05:30	282	curry:0.86, adam:0.44, guard:0.18, get:0.11	-
May 31, 05:42	157	game:0.98, play:0.13	-
May 31, 05:48	277	game:0.97, win:0.13, warrior:0.12	5
May 31, 05:54	502	game:0.96, win:0.21	-
May 31, 06:12	132	game:0.99	-
May 31, 06:18	281	game:0.99	-
May 31, 06:18	128	curry:0.99	-
May 31, 06:24	481	warrior:0.77, win:0.41, cav:0.39, game:0.14, fan:0.14	5
May 31, 06:24	154	back:0.85, come:0.45, warrior:0.17, let:0.11	5
May 31, 06:24	176	golden:0.70, state:0.70	5
May 31, 16:00	130	latest:0.69, opening:0.51, view:0.28, read:0.25, open:0.18	-
May 31, 20:36	129	get:1.00	-
May 31, 21:18	261	get:1.00	-
Jun 01, 17:24	145	great:0.60, fit:0.60, interest:0.32, near:0.32, might:0.28	-
Jun 01, 18:00	125	latest:0.68, opening:0.55, read:0.28, view:0.24, team:0.16	-
Jun 01, 19:12	129	get:1.00	-

Continued E.1 Clustering-Based Detection Outputs of USA

Date (2016)	Tweet #	Detected Keywords	Ground Truth
Jun 01, 20:36	268	get:1.00	-
Jun 01, 23:36	139	get:1.00	-
Jun 02, 00:30	266	get:1.00	-
Jun 02, 14:30	127	take:0.46, nba:0.46, win:0.46, final:0.46, cavalier:0.29	6
Jun 02, 14:36	266	take:0.46, nba:0.46, win:0.46, final:0.46, cavalier:0.29	6
Jun 02, 18:00	138	latest:0.68, opening:0.48, join:0.25, see:0.25, team:0.25	-
Jun 02, 19:18	134	get:1.00	-
Jun 02, 20:00	132	get:1.00	-
Jun 02, 21:06	141	get:1.00	-
Jun 02, 22:12	214	take:0.46, nba:0.46, win:0.46, final:0.46, cavalier:0.23	6
Jun 02, 22:18	441	take:0.46, nba:0.46, win:0.46, final:0.46, warrior:0.23	6
Jun 02, 22:36	135	get:1.00	-
Jun 02, 22:48	137	get:1.00	-
Jun 02, 23:18	133	get:1.00	-
Jun 03, 01:24	130	get:0.99	-
Jun 03, 02:30	126	get:0.98	-
Jun 03, 03:30	250	final:0.49, nba:0.46, win:0.46, take:0.44, warrior:0.23	6
Jun 03, 03:30	154	get:1.00	-
Jun 03, 03:36	379	final:0.48, nba:0.46, win:0.46, take:0.44, warrior:0.23	6
Jun 03, 03:54	149	get:0.99	-
Jun 03, 04:00	322	get:0.99, let:0.12	-
Jun 03, 04:00	242	legend:0.70, john:0.69	9
Jun 03, 04:06	529	get:0.99, let:0.15	-
Jun 03, 04:18	142	step:0.64, jab:0.63, break:0.19, curry:0.19, thompson:0.18	22
Jun 03, 05:48	132	love:0.96, kevin:0.27	11
Jun 03, 05:54	219	ball:0.75, delly:0.59, play:0.17, wrong:0.13	12
Jun 03, 06:06	172	bench:0.73, warrior:0.54, cav:0.36, get:0.14, starter:0.12	29
Jun 03, 06:06	314	livingston:0.83, shaun:0.53	14
Jun 03, 06:12	251	love:0.92, kevin:0.39	11
Jun 03, 06:18	234	game:0.97, cav:0.12	-
Jun 03, 06:18	170	cav:0.93, play:0.19, warrior:0.18, bench:0.16	6
Jun 03, 06:30	310	game:0.95, win:0.20, cav:0.14	6
Jun 03, 06:36	534	game:0.94, win:0.22, cav:0.17, warrior:0.12	6

Continued E.1 Clustering-Based Detection Outputs of USA

Date (2016)	Tweet #	Detected Keywords	Ground Truth
Jun 03, 18:00	127	latest:0.69, opening:0.50, view:0.31, read:0.20, open:0.20	-
Jun 03, 20:06	259	get:1.00	-
Jun 04, 07:18	367	ali:0.66, muhammad:0.57, rip:0.45, rest:0.13, greatest:0.11	15
Jun 04, 07:24	1072	ali:0.65, muhammad:0.55, rip:0.49, greatest:0.14	15
Jun 04, 07:24	135	rip:0.74, greatest:0.55, time:0.29, alus:0.15, legend:0.12	15
Jun 04, 07:24	175	butterfly:0.52, sting:0.50, float:0.47, bee:0.47, rip:0.12	15
Jun 04, 07:24	139	get:1.00	-
Jun 04, 07:24	134	rest:0.74, peace:0.54, champ:0.25, ali:0.20, greatest:0.20	15
Jun 04, 07:30	2238	ali:0.64, muhammad:0.56, rip:0.49, greatest:0.14	15
Jun 04, 07:30	361	rip:0.70, greatest:0.58, time:0.32, alus:0.14, boxer:0.12	15
Jun 04, 07:30	464	butterfly:0.51, sting:0.50, bee:0.48, float:0.47, rip:0.13	15
Jun 04, 07:30	375	rest:0.71, peace:0.61, ali:0.24, champ:0.16, greatest:0.16	15
Jun 04, 07:30	144	alus:0.69, muhammad:0.63, pass:0.16, rip:0.14, away:0.13	15
Jun 04, 07:36	703	butterfly:0.51, sting:0.50, bee:0.48, float:0.47, rip:0.13	15
Jun 04, 08:12	130	get:1.00	-
Jun 05, 00:30	242	get:1.00	-
Jun 05, 18:00	132	latest:0.68, opening:0.44, view:0.28, join:0.26, see:0.26	-
Jun 05, 20:42	131	get:1.00	-
Jun 05, 21:36	265	get:1.00	-
Jun 06, 03:54	186	love:0.78, kevin:0.60, get:0.11, soft:0.11	39
Jun 06, 04:48	236	call:0.67, travel:0.55, lebron:0.42, travels:0.15, finally:0.13	18
Jun 06, 04:54	165	cav:0.93, play:0.28, look:0.14, warrior:0.13	6
Jun 06, 05:00	196	game:0.97, get:0.14, warrior:0.11, cav:0.11	6
Jun 06, 05:30	141	game:0.98	-
Jun 06, 08:12	125	hack:0.67, kylie:0.54, get:0.39, twitter:0.33	20
Jun 06, 08:42	258	get:1.00	-

Table E.2: Clustering-Based Detection Outputs of Canada

Date (2016)	Tweet #	Detected Keywords	Ground Truth
May 31, 04:36	43	get:0.98	-
May 31, 04:48	82	get:1.00	-
May 31, 04:54	98	green:0.65, draymond:0.65, adam:0.36, steven:0.13	2
May 31, 05:00	32	curry:0.96, shot:0.15, knee:0.13	23
May 31, 05:12	44	get:0.99	-
May 31, 05:18	77	get:1.00	-
May 31, 05:30	65	curry:0.83, adam:0.48, get:0.15, guard:0.15, steven:0.15	3
May 31, 05:42	41	get:0.98, game:0.15	-
May 31, 05:54	79	get:0.99, game:0.11	-
May 31, 06:30	42	get:0.97, lebron:0.16	-
May 31, 06:36	80	get:0.99	-
May 31, 19:42	33	usa:0.38, june:0.38, greek:0.38, frat:0.38, every:0.38	-
May 31, 23:30	32	celebrate:0.45, global:0.45, june:0.45, day:0.45, parent:0.45	-
Jun 01, 03:06	31	get:0.98, back:0.15, love:0.13	-
Jun 01, 05:06	63	get:0.99	-
Jun 01, 05:42	95	get:0.99	-
Jun 01, 15:54	45	please:0.83, try:0.28, meet:0.28, chance:0.28, hard:0.28	-
Jun 02, 02:36	43	love:0.58, miss:0.32, hug:0.28, back:0.28, anytime:0.28	-
Jun 02, 04:18	34	love:0.34, get:0.34, twitter:0.34, deep:0.34, wave:0.34	-
Jun 02, 04:54	35	get:0.99	-
Jun 02, 05:00	67	get:0.99	-
Jun 02, 05:24	108	get:1.00	-
Jun 02, 08:18	32	love:0.81, much:0.30, thur:0.28, air:0.28, rock:0.21	-
Jun 02, 16:36	34	take:0.46, nba:0.46, win:0.46, final:0.46, cavalier:0.26	6
Jun 02, 16:54	71	take:0.46, nba:0.46, win:0.46, final:0.46, cavalier:0.24	6
Jun 02, 17:06	107	take:0.46, nba:0.46, win:0.46, final:0.46, cavalier:0.25	6
Jun 03, 02:18	49	guitar:0.58, many:0.58, lie:0.58	-
Jun 03, 02:24	99	guitar:0.58, many:0.58, lie:0.58	-
Jun 03, 03:36	60	get:1.00	-
Jun 03, 04:00	36	john:0.68, legend:0.68, anthem:0.19, national:0.13, sing:0.13	9
Jun 03, 04:06	31	love:0.39, get:0.39, deep:0.39, end:0.39, banger:0.38	-
Jun 03, 04:42	31	get:0.99, barbosa:0.12	-
Jun 03, 04:48	74	get:0.99	-
Jun 03, 06:06	41	livingston:0.88, shaun:0.37, barbosa:0.18, mvp:0.15, man:0.11	14
Jun 03, 06:06	33	bench:0.72, warrior:0.59, cav:0.33, starter:0.13	29
Jun 03, 06:18	40	get:0.99, cav:0.14	-
Jun 03, 06:24	75	get:1.00	-
Jun 03, 06:30	45	game:0.94, win:0.16, get:0.14, warrior:0.11, say:0.11	6

Continues E.2: Clustering-Based Detection Outputs of Canada

Date (2016)	Tweet #	Detected Keywords	Ground Truth
Jun 03, 06:36	33	get:0.93, cav:0.30, game:0.12	-
Jun 03, 06:54	74	get:0.99, cav:0.14	-
Jun 03, 07:54	36	chapter:0.50, follow:0.50, since:0.50, please:0.50	-
Jun 03, 08:00	83	follow:0.50, since:0.50, please:0.50, chapter:0.49	-
Jun 04, 05:18	34	get:0.99, back:0.11	-
Jun 04, 07:18	38	ali:0.60, muhammad:0.59, rip:0.50, greatest:0.14	15
Jun 04, 07:24	134	ali:0.62, muhammad:0.58, rip:0.51	15
Jun 04, 07:30	279	ali:0.63, rip:0.55, muhammad:0.53, greatest:0.14	15
Jun 04, 07:30	31	rest:0.67, peace:0.56, ali:0.32, greatest:0.24, champ:0.15	15
Jun 04, 07:30	38	butterfly:0.50, bee:0.50, sting:0.50, float:0.49	15
Jun 04, 07:36	77	butterfly:0.51, float:0.49, sting:0.49, bee:0.49, rip:0.13	15
Jun 04, 07:54	34	butterfly:0.51, float:0.51, bee:0.47, sting:0.45, rip:0.16	15
Jun 04, 08:00	32	ali:0.67, muhammad:0.65, rip:0.31	15
Jun 04, 09:00	38	flow:0.69, height:0.69, brazo:0.11, trinity:0.11	-
Jun 06, 04:06	71	get:0.99	-
Jun 06, 04:48	32	travel:0.66, call:0.55, lebron:0.41, jame:0.14, finally:0.14	18
Jun 06, 05:06	34	get:0.98, cav:0.17	-
Jun 06, 05:12	67	get:0.99, cav:0.15	-
Jun 06, 08:00	73	fdb:0.41, official:0.41, ringtone:0.41, video:0.41, music:0.41	8

Appendix F

OUTPUTS OF HYBRID EVENT DETECTION METHOD

Table F.1: Hybrid Event Detection Outputs of USA

Date (2016)	Tweet #	Detected Keywords	Ground Truth
May 31, 03:42	36	rust:0.76, bryan:0.60, score:0.17	21
May 31, 03:54	43	anthem:0.70, national:0.69, sing:0.11, get:0.11	-
May 31, 04:06	101	adam:0.73, steven:0.67	-
May 31, 04:18	33	waiter:0.79, dion:0.53, curry:0.29	-
May 31, 04:24	84	ankle:0.72, curry:0.55, break:0.40	22
May 31, 04:54	937	draymond:0.65, green:0.59, adam:0.39, get:0.18, dirty:0.11	2
May 31, 04:54	73	adam:0.82, steven:0.41, foul:0.25, get:0.23, call:0.15	2
May 31, 05:30	45	switch:0.68, adam:0.39, curry:0.36, stop:0.32, get:0.22	-
May 31, 05:36	46	livingston:0.78, shaun:0.59, let:0.13, dunk:0.11	25
May 31, 05:48	42	bonino:0.99, nick:0.11	4
Jun 01, 17:18	46	anyone:0.69, recommend:0.69, manager:0.12, service:0.11	-
Jun 01, 17:18	31	detail:0.70, click:0.70	-
Jun 01, 17:18	71	great:0.60, fit:0.60, interest:0.31, near:0.31, might:0.30	-
Jun 01, 17:18	40	apply:0.61, click:0.61, see:0.34, latest:0.34	-
Jun 01, 17:18	72	latest:0.67, opening:0.54, view:0.39, read:0.28, open:0.14	-
Jun 01, 23:24	99	anyone:0.70, recommend:0.70, manager:0.12	-
Jun 01, 23:24	71	detail:0.71, click:0.71	-
Jun 01, 23:24	153	great:0.60, fit:0.60, might:0.31, interest:0.30, near:0.30	-
Jun 01, 23:24	121	apply:0.61, click:0.61, see:0.36, latest:0.36	-
Jun 01, 23:24	190	latest:0.69, opening:0.52, view:0.28, read:0.25, open:0.18	-
Jun 02, 03:48	42	lyft:0.60, lyftontwitter:0.30, use:0.30, promo:0.30, ride:0.30	-
Jun 02, 04:30	33	hbk:1.00	4
Jun 02, 14:30	127	take:0.46, nba:0.46, win:0.46, final:0.46, cavalier:0.29	6

Continued F.1: Hybrid Event Detection Outputs of USA

Date (2016)	Tweet #	Detected Keywords	Ground Truth
Jun 03, 04:00	47	anthem:0.66, national:0.64, sing:0.30, william:0.15, get:0.11	9
Jun 03, 04:00	248	legend:0.70, john:0.69, anthem:0.11	9
Jun 03, 04:12	58	harrison:0.72, barne:0.69	-
Jun 03, 04:18	136	jab:0.65, step:0.64, curry:0.18, break:0.18, thompson:0.18	10
Jun 03, 04:30	32	clock:0.69, shot:0.61, violation:0.31, cav:0.15	-
Jun 03, 04:30	55	flop:0.74, draymond:0.54, green:0.38	32
Jun 03, 04:30	36	kobe:0.75, commercial:0.48, ghostbuster:0.42	34
Jun 03, 04:42	37	barbosa:0.92, get:0.27, boy:0.20, move:0.12	29
Jun 03, 04:54	83	iggy:0.94, hand:0.18, get:0.17, mvp:0.14, final:0.14	12
Jun 03, 05:06	62	flop:0.96, andy:0.19	33
Jun 03, 05:36	32	steve:0.70, kerr:0.70, mad:0.13	30
Jun 03, 05:48	36	delly:0.79, get:0.53, ass:0.28	12
Jun 03, 06:00	31	denzel:0.70, movie:0.59, new:0.34, washington:0.16, look:0.11	31
Jun 03, 06:06	409	livingston:0.87, shaun:0.45, barbosa:0.11	14
Jun 04, 07:18	48	alus:0.66, muhammad:0.57, pass:0.34, away:0.27, age:0.11	15
Jun 04, 07:18	366	ali:0.65, muhammad:0.58, rip:0.45, greatest:0.12, rest:0.11	15
Jun 04, 07:18	42	rest:0.74, peace:0.55, greatest:0.26, champ:0.19, ali:0.17	15
Jun 04, 07:18	51	greatest:0.68, rip:0.56, time:0.39, boxer:0.18, alus:0.14	15
Jun 04, 07:18	33	butterfly:0.49, sting:0.49, bee:0.49, float:0.46, muhammad:0.14	15
Jun 05, 06:00	54	diaz:0.69, mcgregor:0.65, august:0.20, ufc:0.14	37
Jun 05, 07:48	32	holy:0.54, shit:0.54, bisping:0.51, michael:0.36, bisp:0.11	-
Jun 06, 00:48	47	ave:0.74, div:0.63, unknown:0.11, trouble:0.11	-
Jun 06, 03:00	119	anthem:0.69, national:0.66, play:0.17, santana:0.14, uruguay:0.13	17
Jun 06, 03:42	47	shumpert:0.72, hair:0.62, iman:0.27	38
Jun 06, 03:54	74	ref:0.80, call:0.41, get:0.28, golden:0.17, state:0.17	-
Jun 06, 03:54	164	love:0.70, kevin:0.68, get:0.16	39
Jun 06, 03:54	43	head:0.60, elbow:0.59, back:0.38, foul:0.23, get:0.21	39
Jun 06, 04:30	45	sprint:0.64, verizon:0.55, hear:0.40, switch:0.27, commercial:0.21	41
Jun 06, 04:36	59	vamo:0.58, con:0.58, nalgita:0.57	-
Jun 06, 04:42	74	boy:0.68, backstreet:0.68, miss:0.18, usa:0.16	43
Jun 06, 04:48	77	travels:0.66, call:0.64, lebron:0.32, finally:0.14, nba:0.12	18
Jun 06, 04:48	55	rafa:0.72, marquez:0.67	42

Table F.2: Hybrid Event Detection Outputs of Canada

Date (2016)	Tweet #	Detected Keywords	Ground Truth
May 31, 04:54	83	draymond:0.65, green:0.63, adam:0.36, steven:0.14	2
May 31, 05:00	22	curry:0.96, shot:0.21	23
May 31, 19:42	33	usa:0.38, june:0.38, greek:0.38, frat:0.38, every:0.38	-
Jun 03, 06:06	8	barbosa:0.67, livingston:0.67, mvp:0.17	14
Jun 03, 06:06	17	shaun:0.70, livingston:0.70	14