SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TMLAI TRANSACTIONS ON
MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Volume 6 No. 1
ISSN 2054-7390

SportsBuzzer: Detecting Events at Real Time in Twitter using Incremental Clustering

[1]Jeyakumar Kannan, [2]AR. Mohamed Shanavas, [3]Sridhar Swaminathan
[1,2]Department of Computer Science, Jamal Mohamed College, Tiruchirappalli, India;
[3]Department of Computer Science Engineering, Bennett University, Greater Noida, India;
meetjey@gmail.com; arms3375@gmail.com; sridhar.swaminathan@bennett.edu.in

ABSTRACT

In the recent past, twitter users are highly regarded as social sensors who can report events and Twitter has been widely used to detect social and physical events such as *earthquakes* and *traffic jam*. Real time event detection in Twitter is the process of detecting events at real time from live tweet stream as soon as an event has happened. Real time event detection from sports tweets, such as Cricket is an interesting, yet a complex problem. Because, an event detection system needs to collect live sports tweets and should rapidly detect key events such as *boundary* and *catch* at real-time when the game is ongoing. In this paper, a novel framework is proposed for detecting key events at real time from live tweets of the Cricket sports domain. Feature vectors of live tweets are created using TF-IDF representation and tweet clusters are discovered using Locality Sensitive Hashing (LSH) where the post rate of each cluster based on the volume of tweets is computed. If the post rate is above the predefined threshold, then a key event recognized from that cluster using our domain specific event lexicon for Cricket sports. The predefined threshold helps to filter out small spikes in the tweets volume. The proposed real-time event detection algorithm is extensively evaluated on 2017 IPL T20 Cricket live tweets using ROC evaluation measure. The experimental results on the performance of the proposed approach show that the LSH approach detects sports events with nearly 90% true positive rate and around 10% false positive rate. The results have also demonstrated the influence of different parameters on the accuracy of the event detection.

Keywords: Social media; Twitter; Sports event detection; locality sensitive hashing; incremental clustering.

1 Introduction

Communication between people in modern world is now happening digitally through online social media such as Facebook and Twitter, with the use of high speed internet, web and mobile technologies. Online social media have drastically changed the way of communication between people, groups, and communities [1]. Users of these social media often share information and express individual as well as collective opinions on different issues in the world. Microblogging services such as Twitter is one such form of famous and most widely used social media where diverse group of users share small digital content such as short texts, links, images, or videos [2]. Twitter allows people to share the content in the form of a short text called *Tweet* which is no longer than 140 characters, quickly and easily to the rest of the world. Tweets shared by the people contain diverse information such personal opinions, news, general

DOI: 10.14738/tmlai.61.3861
Publication Date: 26th November, 2017
URL: http://dx.doi.org/10.14738/tmlai.61.3861

information based on their individual behaviors and interests [3]. In a way, this makes the social media users as sensors which share information to the world.

Despite being a medium for users' personal information, Twitter also helps people, groups and organizations to be well informed with live information around the world. Precious knowledge can be gained by monitoring and analyzing the Twitter content continuously. Numerous organizations have now started exploiting Twitter to analyze customers' opinion on their products and services using sentiment analysis. Other than business related applications, Twitter has also become an easier source of information for societal related applications such as retrieval of real-life events, viral news content, prediction of election results and crimes. Recent studies have found that the information provided by human sensors in Twitter can be exploited for detection of real-life events. Due to a large volume of Twitter data and redundancy among Tweets representing the same events, an automation of event detection becomes inevitable.

Recent research has found that major social and environmental events such as earthquakes, deaths of celebrities, and elections can be detected using Twitter [4]. Unlike a regular Television and paper based news medium, reporting of events and news is rapid and quick in the Twitter social media where the information reaches out rest of the world within few seconds. Hence Twitter can be exploited for real time event detection. Real time detection of events has lot of real-life applications in catastrophic situations, politics, entertainment, etc. In addition to the challenges in event detection such as limited length of tweets, rumors, noises like grammar errors, typos and abbreviations, real time event detection is considered much more challenging and complex due to a difficulty in the collection and processing of large volume of Twitter data.

Whilst several event detection approaches based on strategies ranging from term interestingness to topic modeling have been proposed these years, these approaches suffer from high computational cost. Even though research on event detection has been studied well for over a decade, only a limited number of research work have been carried out in the domain of sports. Unlike detection of general events with high profile and global interests, sports event detection is targeted at the detection of events happening within a game by considering a burst of tweets in a smaller scale at a specific time, where traditional event detection approaches are slower and fail~~s~~ to cope with theses scalability issues. In addition, the research on event detection in sports domain focused mostly on NFL soccer game and used offline twitter datasets. Only a few recent research work have aimed to detect key events from NFL games at real-time. However, there is no research work on real-time event detection for *Cricket* sports. Other than the common challenges involved in generic real-time event detection, Cricket event detection poses additional challenges due to large volume of tweets representing Cricket events that happen frequently and rapidly almost every minute during the game.

In this research work, we study the problem of event detection at real time from live Cricket tweets. We propose a novel event detection approach by adopting LSH technique [5] for the domain of Cricket sports. For the incoming live tweets, feature vectors are computed using TF-IDF scores and clustered into different buckets that are indexed on tweet signatures. An event is detected from each active cluster leveraging post rate and is recognized utilizing our Cricket event lexicon. The architecture of the proposed event detection framework is depicted in figure 1.

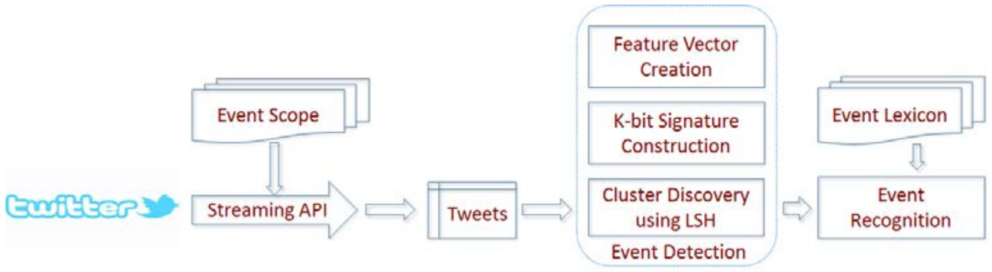


Figure 1. System architecture of *SportsBuzzer*

Our work is unique in a way that we apply LSH to the domain of Cricket sports where events are reported frequently for every minute. To the best of our knowledge, ours is the first of its kind that adopts LSH for sports with rapid events like Cricket, for discovering tweet clusters at real-time from huge volume of live tweets. The major contributions of this paper are summarized as follows:

1. Unlike previous approaches which used offline datasets and focused on NFL games, we present a novel approach which detects events rapidly in real-time from live Cricket tweets based on LSH and event lexicon. Our approach is computationally fast as it adopts LSH technique to discover tweet clusters. Since similar tweets of a particular event will fall into the same bucket, duplicate events are greatly reduced.
2. Similar to NFL soccer sports, Cricket has been one of the popular sports and attract a lot of viewers to the game. Since many viewers post tweets of key events, a widely agreed event lexicon for Cricket sports will help future research. Therefore, we propose an event lexicon that has not been reported before in any previous literature. The event lexicon represents 37 key events for the Cricket sports.

The rest of the paper is organized as follows. In section 2, we survey the related work on Twitter event detection. We introduce our data collection method and tweet preprocessing steps for creating feature vectors in section 3. In section 4, we construct signatures to represent tweet feature vectors in LSH. We describe our LSH based real-time event detection method in section 5 and examine the performance of the proposed approach in section 6. Finally in section 7, we conclude the paper and describe our future work.

2 Related Work

A considerable number of recent research work can be found in Twitter event detection. Based on the domain of which events occur, event detection approaches can be generally classified into different categories such as social, political, environmental, and sports event detections. Recent approaches on event detection can be categorized based on the different classes of solution such as term-interestingness based, incremental clustering based, topic modeling based and frequency based approaches [6]. The readers are recommended to refer to some recent surveys [2, 6] for more detailed comparison of approaches on Twitter event detection. However, recent research on twitter event detection is discussed in this section.

Previous work in twitter event detection focused mostly on detecting physical and social events. Sakaki et al [7] and Qu et al [8] aimed to detect earthquake incidents using Twitter. In another work, Vieweg et al [9] studied detection of natural events such as *grassfire* and *floods* using microblogs. TwitterStand [10] clustered tweets to discover news topics from the Twitter data. Sakaki et al [11] proposed an approach for detecting events such as *earthquakes* and *typhoons* where the authors exploited SVM which is trained on manually annotated Twitter data containing positive and negative tweet samples. Popescu and Pennacchiotti [12] analyzed public discussions happening in Twitter for detecting controversial events related to celebrities. In a related work [13], the authors proposed concert event detection based on factor graph model where the clusters representing events are formed automatically and a canonical value is produced for each event. One of the main drawbacks of the approaches discussed so far is that they detect events from Twitter only several minutes after the actual event.

Becker et al [14] proposed an approach for detecting planned events from twitter data which is filtered using precise queries representing the events where the queries are formed by combining simple rule-based query building strategies. Becker et al. [15] proposed a centrality based approach for extracting high-quality and useful Tweets that are related to different events. The authors in [16] proposed an event detection approach based on generative language modeling using quality indicators of microblogs where query expansion is used to collect messages from microblogs. Weerkamp and de Rijke [17] extracted quality indicators which are useful for event detection such as different emoticons, tweet post length, expressions, word capitalization and URLs. Gu et al. [18] proposed an N-gram based event modeling approach called ETree which uses content analysis approaches for grouping large volume of tweets.

Some approaches exploit geo-locations associated with the tweets for event detection. Valkanas and Gunopoulos [19] proposed an event detection system which clustered users based on their geo-locations where the event detection is achieved by monitoring a sudden change in the emotional state of the user groups. Lee and Sumiya [20] proposed a geo-social event detection system which detects local festivals based on modelling and monitoring of crowd behavior in Twitter. The approach analyzes the regularity in geographical locations using geo-tags of the twitter data.

Since words and their frequencies in tweets are highly correlated to the specific events in general, several term interestingness based approaches have been proposed. Twevent [21] detects nontrivial word segments using statistical information of continuous and non-overlapping word segments in tweets. Bursty event segments are extracted using a fixed window based frequency detection approach where relevant event segments are clustered to filter events based on newsworthiness score using Wikipedia sources. TwitInfo [22] detects spikes in the twitter data and labels them automatically using meaningful and most frequently occurring terms. Gathering of initial tweets are achieved using input keywords from the user where the relevant tweets are crawled. Finally the peaks in the large twitter volume are labeled as sub-events by the system.

TwitterMonitor [23] system detects emerging topics by considering whether high frequency terms co-occur within a small time window that has tweets with bursty terms, and finally applies a greedy strategy to generate groups to reduce computational costs. In a similar system enBlogue [24], emerging topics are detected by measuring window based tag pair statistics where tag correlation shifts in unusual manner are marked as emerging topics. Weng and Lee [25] developed an event detection system EDCoW, exploited analysis of wavelet on word frequencies by calculating new features of words. The authors in [26] aimed to detect emerging topics in Twitter by comparing the frequency of current words and previous words using a directed graph comprising terms that belong to emerging topics.

Some previous work exploit the concept of topic modelling for event detection in Twitter. In an event detection system TwiCal [27], structured representation of important events are extracted from the twitter stream using a latent variable based model for open-domain event detection. A similar system called LECM (Latent Event and Category Model) [28] utilizes semantic concepts to classify different types of events. Hannon et al. [29] exploited post rate of tweets to produce highlights of a World Cup game in an offline mode. However their system was not able to detect the specific events from the game. Chakrabarti and Punera [30] utilized Hidden Markov Models which are trained for representing events in a game. It should be noted that most of the previous work in the domain of sports have not focused on real-time event detection.

Incremental clustering algorithm [31] is utilized for detecting events from Twitter stream where the similarities between a tweet and event clusters are computed for identifying newsworthy events using SVM. Named Entity Recognition was exploited [32] for event detection and tracking where bursty events are detected using named entities in tweets. Few approaches [33], [34], [35] exploited Locality Sensitive Hashing to measure the novelty of a tweet by comparing with previous tweets. Based on the novelty of a tweet, it is further processed for new event detection.

It can be seen that the event detection is accomplished using different strategies. The earlier approaches focused more on detecting physical and social events such as celebrity events, natural disasters, elections etc. Few approaches exploited the network features such as geographical locations of the tweets while others used strategies such as measuring term interestingness and exploiting topic modeling for event detection. It should be noted that most of these previous work were performing event detection on offline datasets. Even though a few work have been carried out in the domain of sports mainly focusing on NFL soccer games, there is no previous work on event detection for Cricket sports, to the best of our knowledge.

3 Preprocessing of sports tweets

With a length of just 140 characters, Twitter has the shortest delay in delivering user comments to citizens, compared to other social media platforms such as *blogs*. As tweets are highly noisy, which contain URLs, mentions, replies and others, preprocessing has been a fundamental step in detecting key events from live tweet streams.

3.1 Collecting live tweets

Cricket fans and audience of a live game post tweets about interesting moments throughout game time. So, these twitter users can be considered as sensors who can deliver current updates about key events (e.g. Boundary, Sixer, Catch) in a game at real-time. *SportsBuzzer* relies on and leverages these sensors to collect data and perform robust detection and recognition of key events at real-time.

SportsBuzzer requires live tweets which can be filtered based on some relevant keywords and without any maximum limit for streaming. Hence, *Streaming API* of Twitter (www.twitter.com) is the most suitable type of data collection for our real time event detection task. With the help of Streaming API, we will be able to collect live tweets continuously, based on the scope of events such as hashtags. For instance, we have used a keyword *RCBvRPS* to stream all live tweets at real-time when the game was ongoing. The keyword *RCBvRPS* denotes an IPL T20 cricket game between the teams *Royal Challengers Bangalore* (RCB) and *Rising Pune Supergiant* (RPS) that was held during April 2017 in India. Our *SportsBuzzer* runs continuously collecting tweets without any break during the entire game time, detects events from tweets at real-time and also archives all gathered tweets in JSON format for later offline analysis.

3.2 Removing noise from tweets

Figure 2 depicts the work flow of converting a raw cricket tweet into a feature vector. The feature vector will be used as input for the clustering process. The creation of a feature vector consists of two steps - preprocessing a given raw tweet and preparation of the feature vector from the preprocessed tweet. In the preprocessing step, a set of features (i.e. unigrams) are extracted from each tweet. For simplicity, only unigram features from a tweet are considered in this work. Later, each preprocessed tweet is converted to its equivalent feature vector. Both steps are explained below in greater detail.

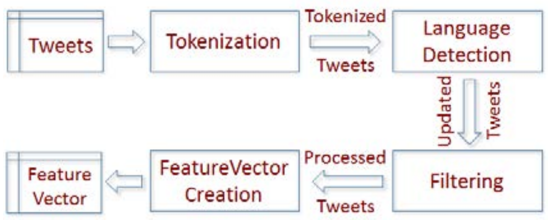


Figure 2. Work flow for building tweet feature vector

The preprocessing step eliminates noise which reduces the accuracy of an event detector. Here, each tweet is first tokenized into a sequence of terms. Only tweets posted in English are considered for further processing. In Information Retrieval (IR), the most commonly occurring words in a text document are called *stop words*. The tokenized tweet is checked against a standard stop word list. The list of stop words from Python's Natural Language Tool Kit (NLTK) is used to eliminate all terms that are least meaningful and less contributing to event detection. For instance, articles in English such as *a*, *an* and *the* are removed from tweets. The URLs and all token starting with '@' (i.e. a mention or reply) are also removed from the raw tweets. Tokens that contain only alphanumeric characters are treated as valid tokens. At the end of this process, each tweet has only features that will be included in the vector space model.

3.3 Creating tweet feature vector

Mathematically, a vector is represented by its direction and magnitude. Many term weighting schemes use values or magnitudes to represent the features of a text document such as tweet. Term Frequency - Inverse Document Frequency (TF-IDF) has been one of the most fundamental term weighting schemes in IR, which is used to represent the features of a tweet as numerical values. TF-IDF is formally defined as a product of term frequency (TF) and inverse document frequency (IDF). Here, TF represents the importance of a term in a document and IDF represents the importance of the term in the entire document corpus. The TF-IDF weighing scheme assigns weight to term *w* in document *d* using equation 1 and 2 [36]:

$$tf - idf(w,d) = tf(w,d) * idf(w) \tag{1}$$

$$idf(w) = log\frac{N}{df(w)} \tag{2}$$

Here, $tf - idf(w,d)$ is known as TF-IDF weight of a term *w* in document *d*. The term frequency $tf(w,d)$ represents the number of times the term *w* occurs in a document *d*. The inverse document frequency $idf(w)$ helps to scale down the term frequency of w, if the term w occurs in almost all documents in a data corpus. Here, N is the number of documents in a data corpus and $df(w)$ denotes the number of documents in which the term w occurs at least once. Then, the features of a tweet are mapped to the vocabulary in order to generate the tweet feature vector where TF-IDF weight is assigned to a vocabulary term that appears in the tweet. For simplicity, we keep our vocabulary as a static dictionary.

4 Incremental Clustering of sports tweets

In this section, we first describe the problem of finding nearest neighbors and approximate nearest neighbors. Then, we examine the suitability of locality sensitive hashing for the incremental clustering of tweets. Incoming live tweets have to be represented as feature vectors so that they can be clustered in LSH. We finally explain the process of construction of signatures for tweet feature vectors which are the indices for the LSH approach.

4.1 Approximate nearest neighbors

Given a set of N points P = $\{P_1,P_2,P_3,\ldots,P_N\}$ represented as a matrix M and a query point Q, a nearest neighbor search finds a point in P that is closest to Q ∈ M [5]. In case of vector space model, points are documents and a query point is the document to be searched. The nearest point to a query point can be found by simply computing the distance between all points in *P* to *Q* and selecting the one point P_i ∈ P which is the closest to *Q*. The nearest neighbour approach is computationally expensive for high dimensional data such as documents represented in vector space model. This problem is known as *Curse of Dimensionality* [5].

To alleviate this problem, a variation known as Approximate Nearest Neighbour (ANN) search was proposed. The ANN search finds an approximate nearest neighbor point P' in P that is the closest to Q within a radius r, as shown in equation 3.

$$\forall P' \in P, d(P', Q) < (1+\varepsilon)d(P', Q) \tag{3}$$

Here, $d(P', Q)$ is the distance between P' and Q and $(1+\varepsilon)$ is a constant factor [5]. Locality Sensitive Hashing (LSH) [5] is a popular approach to address the problem of ANN search. In this paper, we have adopted LSH technique to find tweet clusters from which events are detected by computing the post rate of a cluster. Once an event is detected, then it is recognized using the event lexicon of Cricket sports.

4.2 LSH for incremental clustering

The LSH technique has been applied to information retrieval, pattern recognition, dynamic closest pairs and fast clustering problems [5]. The key idea of LSH is to apply hash functions in such a way that the probability of collision is much higher for similar objects than for dissimilar objects [5], i.e. the objects close to each other will most likely fall into the same bucket. Intuitively, a hash function is locality sensitive if two points that are close under the similarity distance measure are more likely to collide.

In LSH, initially a set of points will be preprocessed and stored into L number of buckets. Each point is hashed using *k* hash functions and stored in L buckets. The concept of bucket can be implemented in several ways. Hash table is an obvious choice for representing a bucket. The hash value of data points acts as an index of a hash table. With the hash value of a query point *q*, all buckets are searched to retrieve the points that are similar to *q*. The similarity distances between *q* and each of the retrieved similar points are computed and the one point that is close to *q* is selected as a nearest neighbor for the query point *q*. The LSH scheme proposed by Charikar [37] applied cosine similarity metric to compute the similarity between two document vectors, where cosine similarity is a dot product of feature vectors normalized by their norms. The cosine similarity will become 1 if the document vectors are parallel and 0 if the document vectors are orthogonal to each other.

4.3 Signature for tweet feature vector

Traditional event detection methods do not provide sufficient solutions to handle the exponentially growing social media streams where LSH has become a successful solution for processing these large data streams. The LSH approach applies hash functions such that the probability of collision, i.e., falling into the same bucket, is much higher for similar tweets than that of dissimilar tweets. The gap between two dissimilar tweets should be larger enough so as to prevent the collision of dissimilar tweets into the same bucket. In this research, our proposed methodology for discovering clusters of tweets uses hash table based LSH for computing the nearest neighbours.

We adopt the LSH approach proposed by Charikar [37] that defines a hash function *h* to generate a *k*-bit signature for the tweet feature vector. The hash function (equation 4) computes the dot product between the tweet feature vector *u* and *m*-dimensional random unit vector *r* and retains the sign of the resulting product. Each dimension in *r* is drawn from Gaussian distribution with mean 0 and variance 1.

$$h(u) = \begin{cases} 1, if\ r.u\ \geq 0 \\ 0, if\ r.u < 0 \end{cases} \tag{4}$$

The *k*-bit signature reduces the dimension of the original tweet feature vector. As it is a low dimensional vector, LSH approach clusters large number of tweet vectors very fast. Charikar applied cosine similarity metric to compute the similarity between two document vectors and is defined in equation 5.

$$\cos\big(\theta(u,v)\big) = \cos\big((1 - \Pr[h(u) = h(v)])\pi\big) \tag{5}$$

Here, $\theta(u,v)$ is the cosine angle between the vectors u and v and is proportional to the hamming distance of their signature vectors while preserving the cosine similarity in high dimensional space. $\Pr[h(u) = h(v)]$ is the probability that a random hyper plane separates two vectors, which is proportional to the cosine angle between them. The hamming distance is the number of bits that differ between two binary vectors.

5 Proposed Approach

In this section, we first describe our proposed approach for real-time Cricket event detection. We utilize locality sensitive hashing method for implementing the online incremental clustering of sports tweets. With the detected clusters, key events are recognized by leveraging our event lexicon. While game spectators would continue to tweet about the same event for a long time, our detector would alert it repeatedly, assuming a new event. Hence, we also elaborate on handling duplicate event alerts.

5.1 Event detection using LSH

Our event detection framework *SportsBuzzer*, strives to analyze cricket tweet streams to detect events, such as *boundary* and *sixer*, accurately and as early as possible. We utilize locality sensitive hashing method to cluster event related tweets at real time. Post rate of a cluster will be computed by considering the volume of tweets at a given time. If it is above a predefined threshold, an event is declared detected. We delete the cluster once an event is detected from it. The particular event will be recognized by analyzing the representative tweets from a cluster. An event that has occurred most number of times in those selected tweets, is the recognized event. The proposed algorithm for LSH based event detection and recognition is depicted in figure 3.

```
Input: Live tweets, similarity threshold ST, buckets L, signature length K, post rate T
Output: Event name and its tweets
1:  create event lexicon for pre-determined event types
2:  build TF-IDF based dictionary D using lexicon
3:  for each bucket i ϵ L do
4:      create hash table ht[i]
5:      create random vector rv using Gaussian distribution
6:  end for
7:  repeat
8:      for each incoming tweet t do
9:          construct tweet feature vector tv for t using D
10:         create k-bit signature ts for tv
11:         for each bucket i ϵ L do
12:             get collision for ts
13:             add tv with key ts in ht[i]
14:         end for
15:         get nearest neighbor NN for tv from collisions
16:         if similarity(tv, NN) < ST then
17:             create new cluster c
18:             addTweetVectorToCluster(tv, c)
19:         else
20:             if tv not in NN's cluster $c_{NN}$ then
21:                 addTweetVectorToCluster(tv, $c_{NN}$)
22:             end if
23:         end if
24:     end for
25: until connection closed

26: for each cluster c ϵ C do
27:     if postRate(c) > T then
28:         get text of all tweets in cluster c
29:         select an event with highest document frequency using lexicon
30:         display event name and its tweets using lexicon
31:         delete cluster c
32:     end if
33: end for
```

```
Function: addTweetVectorToCluster(tv, c)
1:  c.tweetFrequency += 1
2:  c.tweetVector += tv

Function: postRate(c)
1:  sort timestamps of tweets based on c.tweetFrequency
2:  cFirst ← count(tweets) in first half timestamps
3:  cSecond ← count(tweets) in second half timestamps
4:  return cSecond / cFirst
```

Figure 3. Proposed algorithm for LSH based key events detection

To improve the fidelity of the detector, we do not consider clusters that contain a single tweet. Similarly, we delete all clusters whose life span is more than five minutes, because we expect an event might occur within five minutes itself in Cricket sports. An important requirement for a real time event detection system is that it should detect and report events in near real time to the needy people. Our online incremental clustering approach clusters similar tweets together so as to detect and recognize key events quickly. To obtain the optimal post rate, similarity threshold, number of buckets and number of hash functions, we iterate our incremental clustering method with different parameter values. Because, the choice of these values impact the accuracy of the event detection. The evaluation results with different parameter setup will be explained in detail in section 6.

5.2 Lexicon-based event recognition

Once the post rate of a cluster is above the predefined threshold, *SportsBuzzer* assumes that some event has occurred in that cluster. The event recognizer then identifies the specific event in that cluster based on document frequency measure. The document frequency of each key event is computed in a case insensitive way, considering all tweets in the middle timestamp of the particular cluster. The event recognizer selects an event which has the highest document frequency and declare that event a winner. It should be noted that the document is characterized by the representative tweets of a cluster.

For an accurate event recognition, it is crucial that we design a domain specific lexicon describing all game terminologies for Cricket sports. Two important requirements should be considered while designing the lexicon. The event names should be more descriptive, as every game viewer tweets about the same event in different ways, using different words. Every game viewer just uses the event name to describe the happening of an event, because of the limited size of a tweet. Our event lexicon (a portion of which is shown in figure 4) describes 37 Cricket sports events, such as *bowled out*, *run-out*, *lbw* and *leg bye*. The lexicon is populated with different event terminologies collected from ESPNCricInfo (www.espncricinfo.com/ ci/content/story/ 239756.html) website.

```
BOUNDARY = ['boundary', 'four', 'fours', '4']
SIXER = ['sixer', 'six', '6']
CATCH = ['catch', 'c']
```

Figure 4. Lexicon for few events of *Cricket* sports

Our event lexicon is easy to implement and a better choice for a real time event detection from live tweets. Because, it does not require any training for the event recognition as required by other statistical event recognition models. Furthermore, there are real time applications which do not have training data such as celebrity deaths and terrorist attacks. Therefore, it is very practical to adopt lexicon based approach for event recognition.

5.3 Preventing duplicate event alerts

Duplicate event is an event that is repeatedly reported as a new event for a long time, even after the actual event has occurred a long back. The LSH algorithm groups tweets of an event such as boundary, into the same cluster until that event is detected (i.e., the post rate of that cluster is above the threshold). New cluster will not be created until a new tweet is sufficiently dissimilar from existing clusters. Thereby, all similar tweets will go into the same cluster and thus an event alert happens only once.

Nevertheless, there is still an issue with the duplicate event reporting. After an event is detected and reported for a cluster, the cluster will be deleted. Sometimes, because of the intense discussion of the current event among the audience of the game, a new cluster will be created once again for the same event based on the new set of tweets describing the same event. Hence, the event detector will report this duplicate event as a new event. We solve this problem by comparing the timestamp of an event that is already reported with the timestamp of the new cluster. If the time difference of these two timestamps is less than 60 seconds, then we ignore the new cluster and do not alert this event as a new event. Our assumption is that an event of the same type cannot happen once again within 60 seconds. Because, in cricket sports, an over containing six balls should be delivered within 5 to 6 minutes. So, the process of bowling and batting should be finished within an average time frame of 60 seconds.

6 Experimental Results

In this section, we will present the experimental results of the proposed Cricket key events detection approach. We evaluate the *SportsBuzzer* approach using tweets of IPL T20 2017 cricket sports. The proposed approach has been implemented in Python and Pika. The evaluation proves that LSH based event detection method can detect events with more than 90% true positives and less than 10% false positives. We will now present the data set, evaluation criteria and parameter setup and evaluation results.

6.1 Dataset

Twitter's Streaming API was used to crawl live tweets at real time using official hashtags of games provided by Indian Premier League (*www.iplt20.com*) in 2017 IPL T20 season held during April 2017 in India. Our dataset collection includes tweets of 44 games with a file size of over 6GB. Out of these games, we selected a game RCBvsRPS held on 16 April 2017 as it was considered an interesting and most anticipated match. Table 1 shows the description of RCBvRPS game.

Table 1. Game statistics of RCBvRPS

RCBvRPS game	Total	Total min	Mean (re)tweets per min	Min (re)tweets per min	Max (re)tweets per min	Standard deviation
Tweets	34967	232	150.72	38	354	67.4779909455
Retweets	16162	232	69.664	13	176	34.4786150528

Figure 5 depicts the tweet post rate of RCBvRPS game. It shows that the volume of tweets posted during the end of the game is high. Also, it contains several exciting moments throughout the game time.

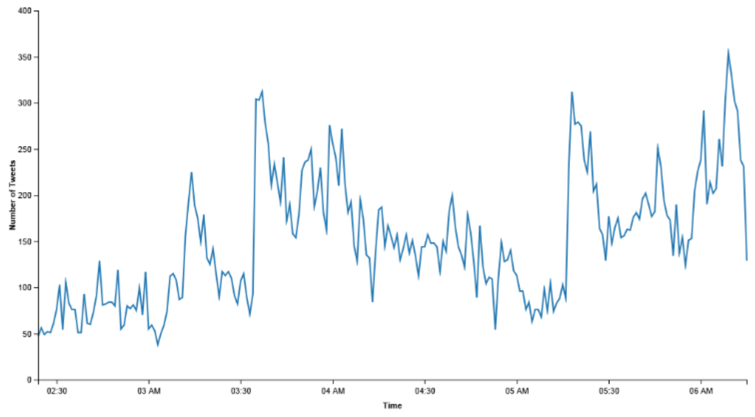


Figure 5. Post rate of tweets of game RCBvRPS

We have collected ground truth of all events from IPL live commentary site (*www.iplt20.com*). We have also cross-verified the time of each event with other live commentary websites. Table 2 shows the description of ground truth events for the RCBvRPS game.

Table 2. Summary of events in ground truth

Game	No. of ground truth events	No. of Boundaries	No. of Catches	No. of Sixers	Other events
RCBvRPS	81	24	6	9	42

6.2 Evaluation Criteria and Parameter Setup

Using all 24 boundaries, 9 sixes and 6 catches happened in the RCBvRPS game of 2017 IPL T20 cricket season, we illustrate the effectiveness of our LSH-based event detection method using Receiver Operating Characteristics (ROC) curves.

The results generated by our event detector are compared against the ground truth of RCBvRPS game. We define four evaluation windows with different times namely 1min, 5min, 10min and 15mins for comparison. Accordingly we compute the number of hits and misses for each evaluation window. A detection is considered a *hit* if the detected event is reported within a particular evaluation window, otherwise it is a *miss*.

Like any binary classifier, our detector can make two types of errors: reporting an event when nothing happens (i.e., false positive) and reporting nothing when an event happens (i.e., false negative). True Positive Rate and False Positive Rate (equation 6 and 7) are computed as follows:

$$TPR = \frac{TP}{TP+FN} \tag{6}$$

$$FPR = \frac{FP}{FP+TN} \tag{7}$$

For a particular study, different set of TPRs and FPRs are computed with various parameter settings. There are four parameters to be adjusted namely Post Rate (PR = 0.2, 0.5, 0.8), nearest neighbor Similarity Threshold (SIM = 0.2, 0.5, 0.8), number of hash tables (L = 5, 10, 15) and number of projections (K = 5, 11, 13, 19). For an experiment with a particular parameter setup with different Post Rates results in a set of TPRs and FPRs. The RoC curves are plotted using these rates and Area Under ROC curves (AUROC) are calculated. The AUROC curve represents the accuracy of the event detector. A high AUROC denotes a high true positive rate and low false positive rate while a low AUROC denotes a low true positive rate and high false positive rate

6.3 Results

We evaluate *SportsBuzzer* system using 2017 IPL T20 cricket games and present the accuracy of the event detection for key events such as *boundary*, *catch*, *sixer*, *boundary+catch+sixer, boundary+catch, boundary+sixer, catch+sixer*. We also show the influence of various parameters such as similarity threshold in finding nearest neighbor (SIM), number of hash tables (L) and number of projections (K), besides the effect of retweets in real time event detection. Finally, we compare the average computation times to find a nearest neighbor from buckets.

6.3.1 Performance on detecting events

Figure 6 shows the ROC curves that illustrate the performance of our LSH approach in detecting different Cricket events such as *boundary*, *catch*, *sixer*, and major events (*boundary+catch+sixer, boundary+catch, boundary+sixer, catch+sixer*) in the RCBvRPS game. The evaluation is conducted for different evaluation window sizes such as 1, 5, 10 and 15 minutes, with fixed parameter values SIM=0.5, L=10 and K=13.

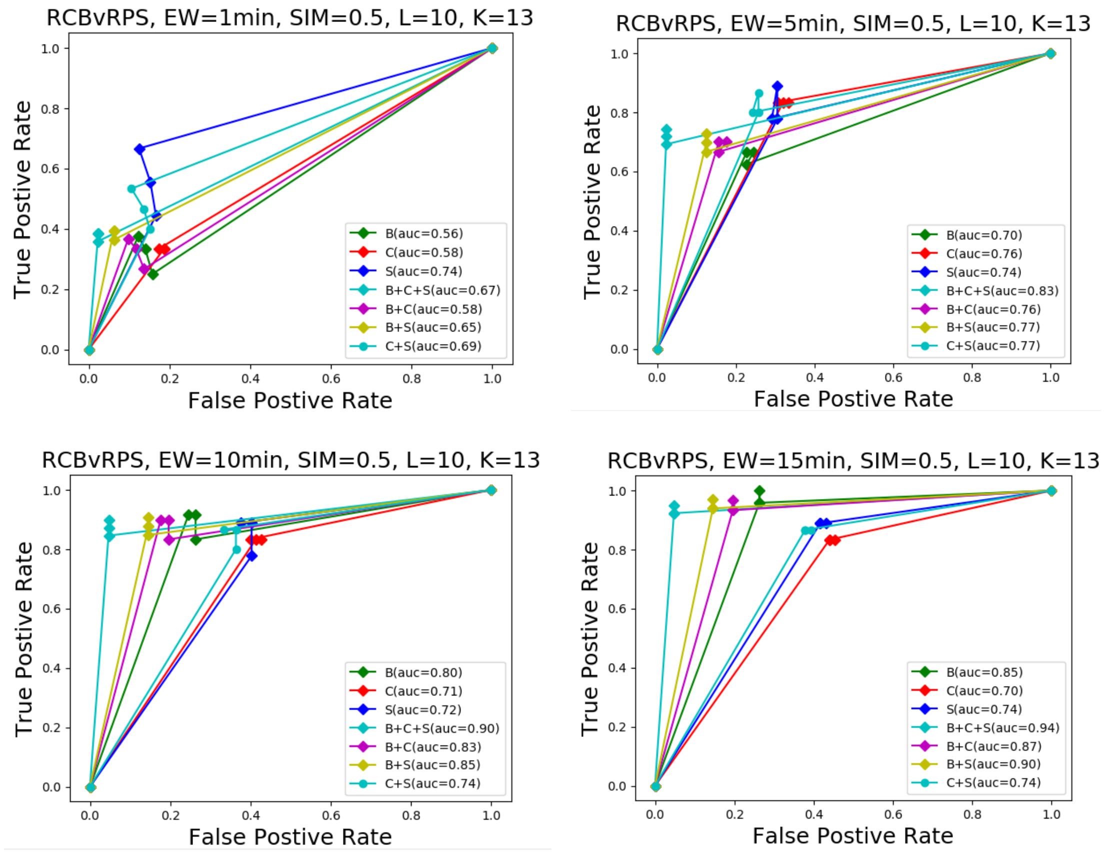


Figure 6. Detection performance of individual events

Results show that our LSH based approach delivers a decent performance in detecting the key game events. The event *sixer* is detected fast within 1 min evaluation window than other events, because sixer is considered a highly exciting event by the viewers of this game. When size of the evaluation window increases, boundary is detected well, as twitter users reported this event with smaller delay. Due to a high initial excitement among twitter users, catch is detected well within 5 minutes and the performance decreases for 10 and 15 min windows. Major events (*boundary+catch+sixer* combination) of the game are also detected well with our LSH approach. Major events (*boundary+sixer* combination) performs better than other combinations. From these graphs, we can observe that almost all key events are detected with a decent accuracy (with 80 percent true positives in AUROC) within an evaluation window of 10 minutes. Also, the performance for evaluation window of 15 minutes is highly similar to that of 10 minutes window. So, we can conclude that most of the key events are detected and reported well even within 10 minutes from the actual happening of those events.

6.3.2 Influence of different similarity thresholds

Figure 7 shows the ROC curves that illustrate the performance of our LSH approach in detecting key events for different evaluation windows, under various similarity thresholds. The influence of various similarity thresholds (0.2, 0.5 and 0.8) on event detection is evaluated for key events and a combination of key events under a fixed L and different evaluation window size of 1, 5, 10 and 15mins.

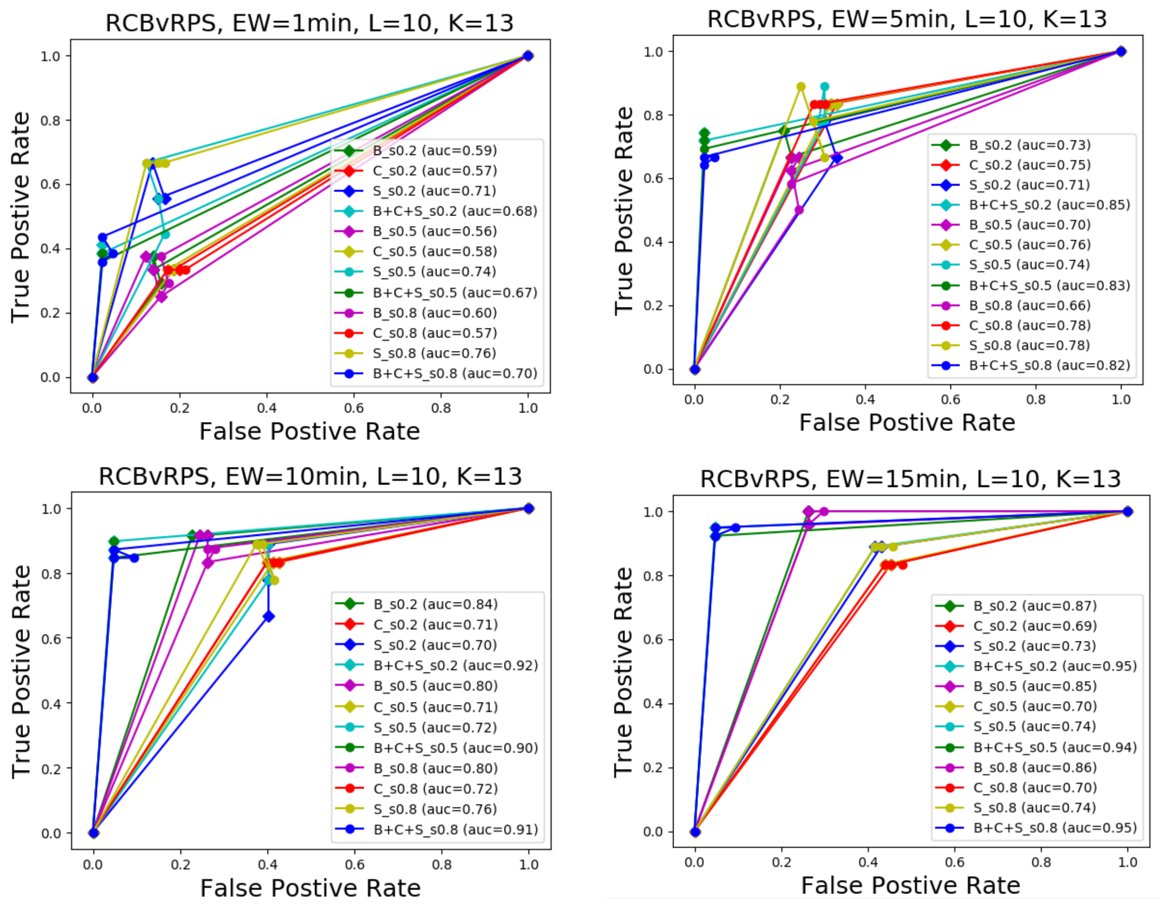


Figure 7. Detection performance for different similarity thresholds

From these graphs, we can observe that the overall performance is improved as the window size increases. Even though individual events present slightly different performances, the effect of similarity thresholds can be assessed using a combination of major events such as B+C+S. When the evaluation window is small (i.e. 1 min), high similarity threshold is preferable, whereas for higher evaluation windows (like 10 or 15mins), lower similarity threshold is better. However, no significant difference can be noticed in the performances with any of these thresholds, because of the consistency in the representation of tweets using LSH based projections. Therefore, a reasonable threshold of 0.5 greatly strikes a balance for our approach in detecting all key events.

6.3.3 Influence of hash table size

Figure 8 shows the ROC curves that illustrate the performance of our LSH approach in detecting key events for different evaluation windows, under various hash table sizes. The influence of various hash table sizes (5, 10 and 15) on event detection is evaluated for key events and a combination of key events under a fixed similarity threshold and K and different evaluation window size of 1, 5, 10 and 15mins.

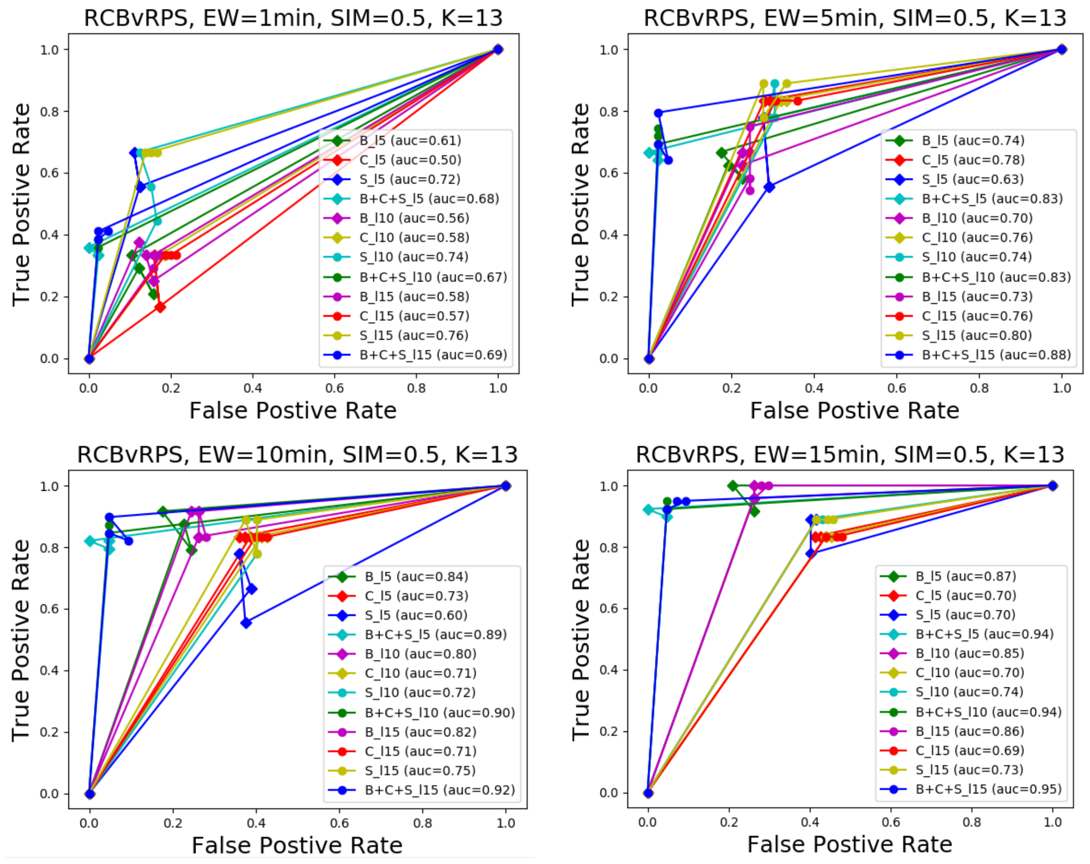


Figure 8. Detection performance for different sizes of hash tables

From the results shown in the ROC curves, it is evident that the accuracy of event detection improves in many cases when the number of hash tables increases, especially in major events such as B+C+S. For higher number of hash tables, the true positive rate reaches nearly 90% and the false positive rate is only 10%. Obviously, higher number of hash tables increases the chance of detecting the appropriate nearest neighbor. In a way that if one hash table misses out the correct nearest neighbor, other hash tables with different projections are more likely to consider it as a nearest neighbor. It is recommended that the number of hash tables should be selected carefully as it might increase the search time of the nearest neighbor. For a better tradeoff between speed and accuracy, it is preferable to keep L to a medium value.

6.3.4 Influence of different number of projections

Figure 9 shows the ROC curves that illustrate the performance of our LSH approach in detecting key events for different evaluation windows, under various projections. The influence of the number of projections (5, 11, 13 and 19) on event detection is evaluated for key events and major events under a fixed similarity threshold and L values and different evaluation window size of 1, 5, 10 and 15mins.

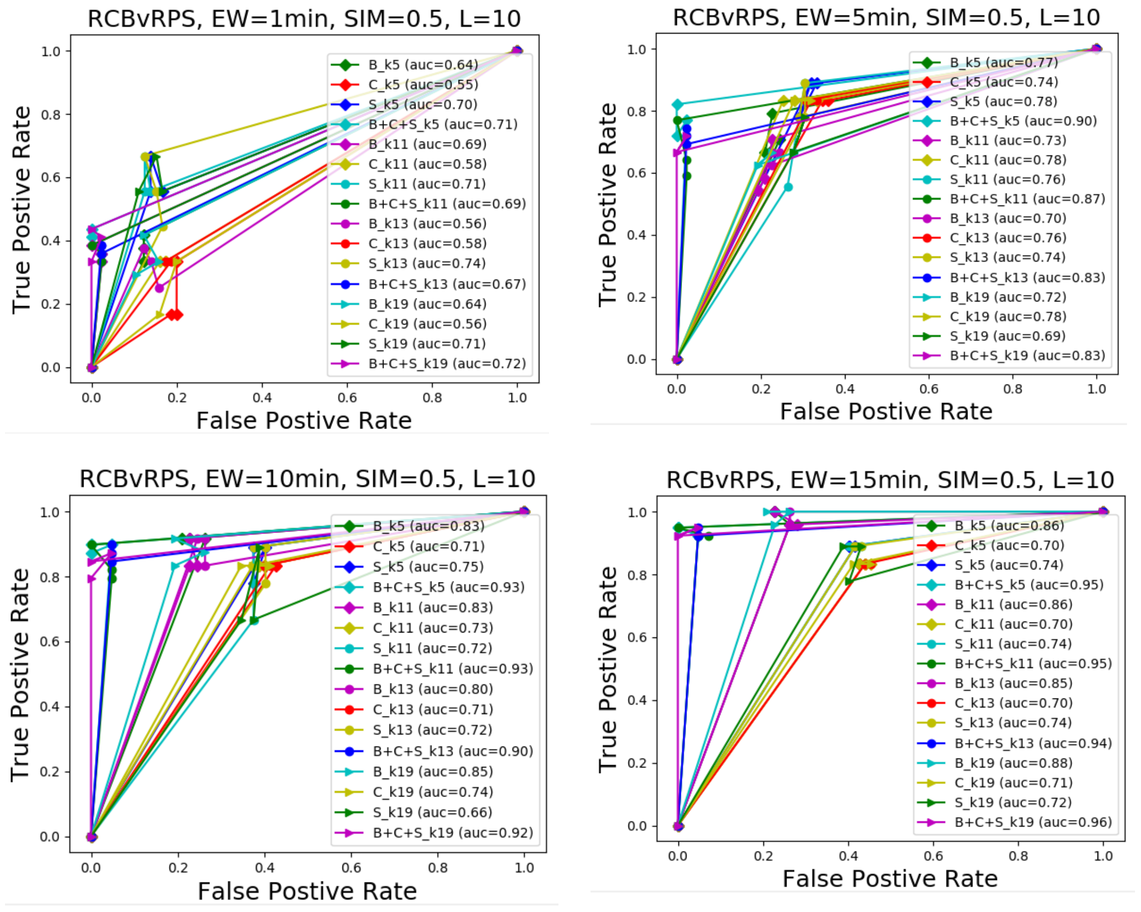


Figure 9. Detection performance for different number of projections

From the results shown in the ROC curves, our proposed approach achieves better accuracy for smaller number of projections, which can be noticed easily in the case of major events (B+C+S). The intuition is that when the number of projections are low, two relevant tweets might be indexed with the same signature. When the number of projections increases, location sensitivity of a tweet increases, thereby two relevant tweets might get different signatures. However, it is likely that the signature for both relevant and irrelevant tweet might be same because of lower number of projections, which results in both tweets falling into the same bucket. Similar to parameter L, number of projections is directly proportional to the search time of a nearest neighbor. Therefore, value for K should be chosen accordingly. Based on our experiments, we can conclude that K = 11 is a reasonable value to balance speed and accuracy.

6.3.5 Analysis on hash table size and number of projections

Figure 10 shows the ROC curves that illustrate the performance of our LSH approach in detecting key events for different evaluation windows, under three combinations of hash tables and projections. The influence of a number of hash tables (5, 10 and 15) and projections (5, 11 and 19) on event detection is evaluated under a fixed similarity threshold and different evaluation window size of 1, 5, 10 and 15mins.

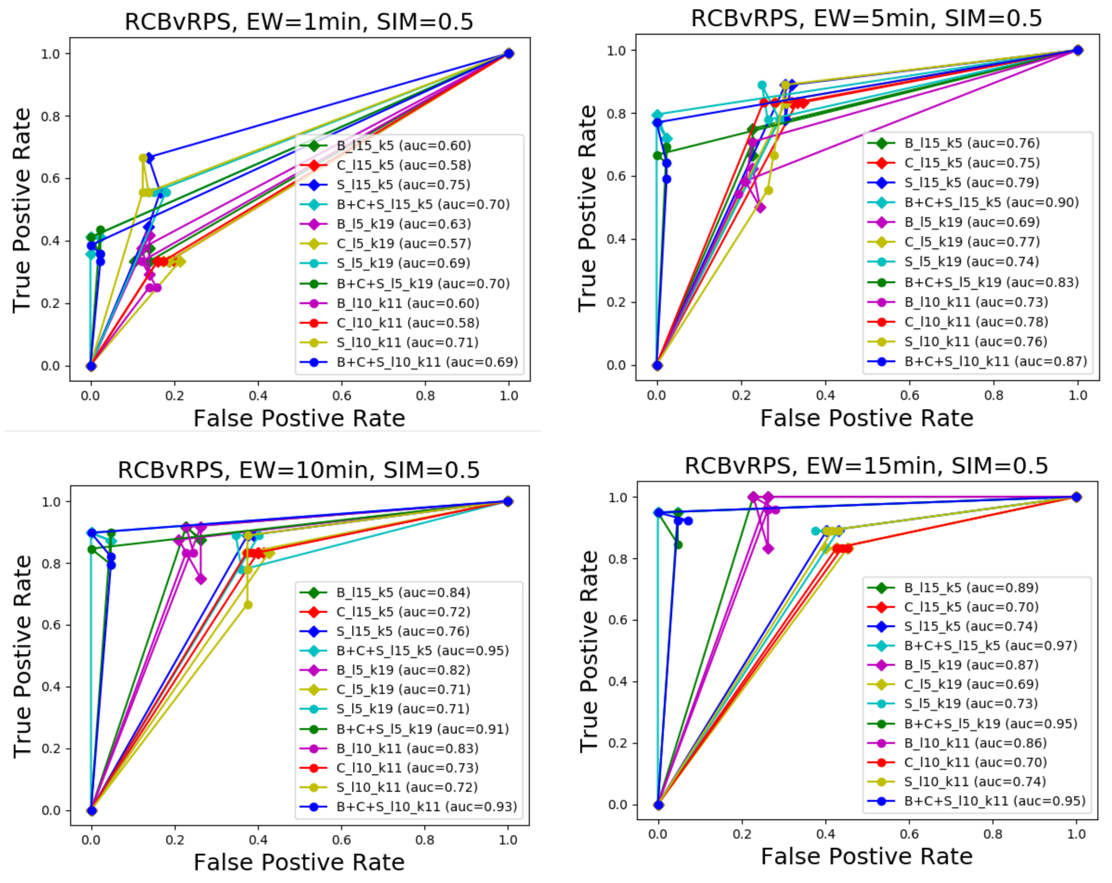
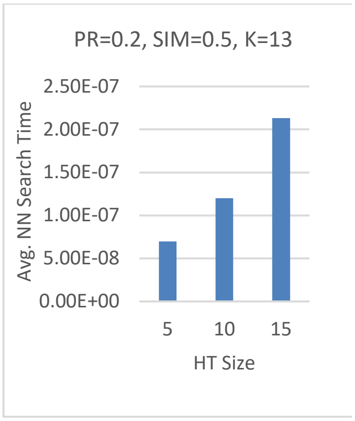


Figure 10. Detection performance for different number of hash tables and projections

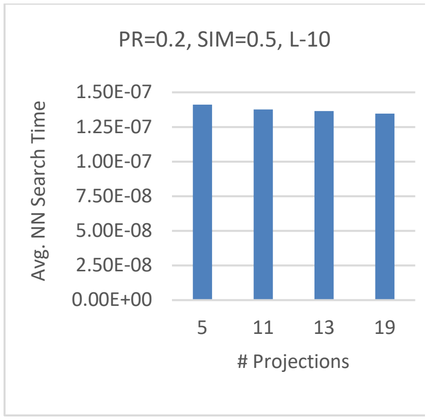
From the results shown in the ROC curves, it is clear that true positive rate improves for larger evaluation windows, as seen in the previous sections. Our LSH based proposed approach achieves a better accuracy for higher number of hash tables (L=15) and smaller number of projections (K=5). As discussed in the previous sections, higher number of hash tables increases the chance of getting correct nearest neighbours, while lower number of projections gives the same signature to relevant and similar tweets. Similarly, medium number of hash tables (L=10) and projections (K=11) achieves the second best performance, whereas lower number of hash tables (L=5) and higher number of projections (K=19) decreases the accuracy. Therefore, a higher number of hash tables and lower number of projections is preferred.

We have also computed the average search time of finding nearest neighbours from 35000 tweets of the game RCBvRPS. The search time increases linearly when the number of hash tables (HT) increase (figure 11a). The number of projections (P) almost remains a constant (figure 11b). While considering both (HT and P), search time is high for case1 (HT=15, P=5), low for case2 (HT=5, P=19) and medium for case3

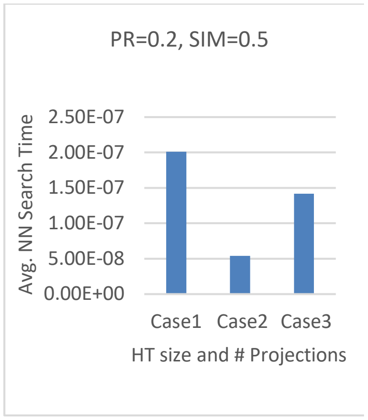
(HT=10, P=11) (figure 11c). We can see that the hash table size has more influence in deciding the speed of the nearest neighbor search.



(a)



(b)



(c)

Figure 11a, 11b & 11c. Nearest neighbor search time

6.3.6 Performance under different evaluation windows

Figure 12 shows the ROC curves that illustrate the performance of our LSH approach in detecting key events under different evaluation window of size 1, 5, 10 and 15mins. Since there is a significant delay between the actual event time and the time twitter users post tweets, the influence of an evaluation window greatly impacts the accuracy of our LSH approach.

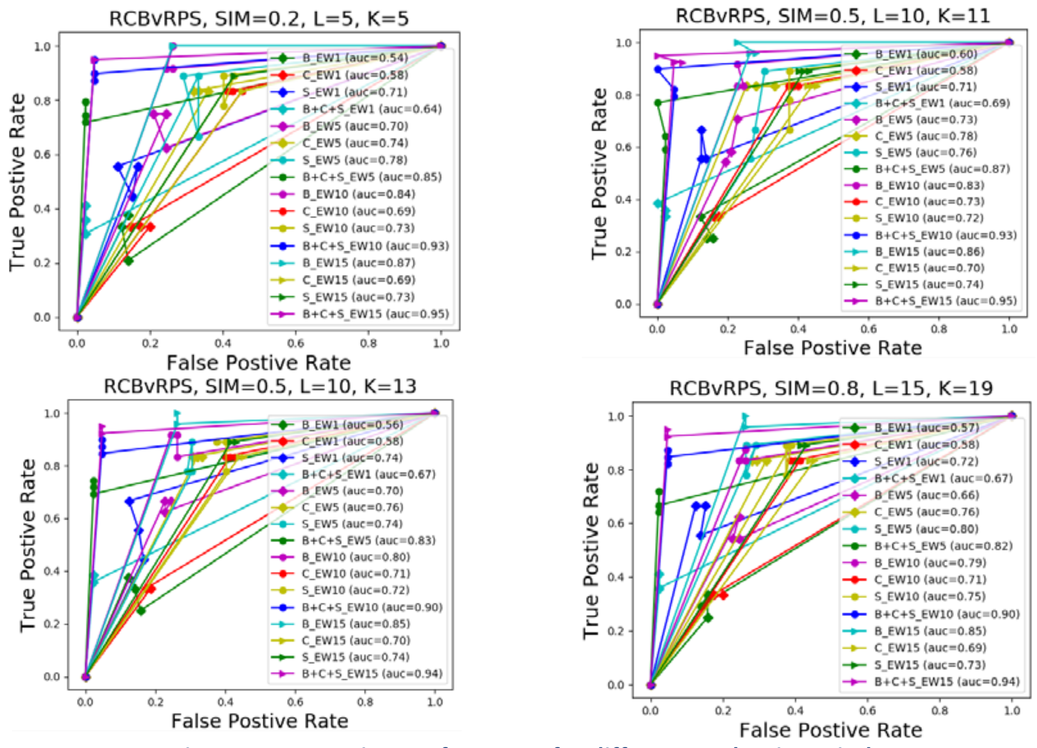


Figure 12. Detection performance for different evaluation windows

From the results shown in the ROC curves, our LSH approach detects most of the events within a window of 5 minutes, after the event has happened. Since a larger evaluation window allows some delay in detecting events, true positive rate~~s~~ reaches around 80% from nearly 60%. Hence, accuracy can be improved if considerable time delay in detection is permissible.

6.3.7 Performance of all tweets vs. no retweets

Figure 13 shows the ROC curves that illustrate the performance of our LSH approach in detecting key events with fixed SIM, L and K values. Since incoming tweets may also include their retweets, influence of retweets in detecting events is analyzed by evaluating the LSH approach with *all tweets* and with *no retweets*. The evaluation is conducted for different evaluation window of 1, 5, 10 and 15mins.

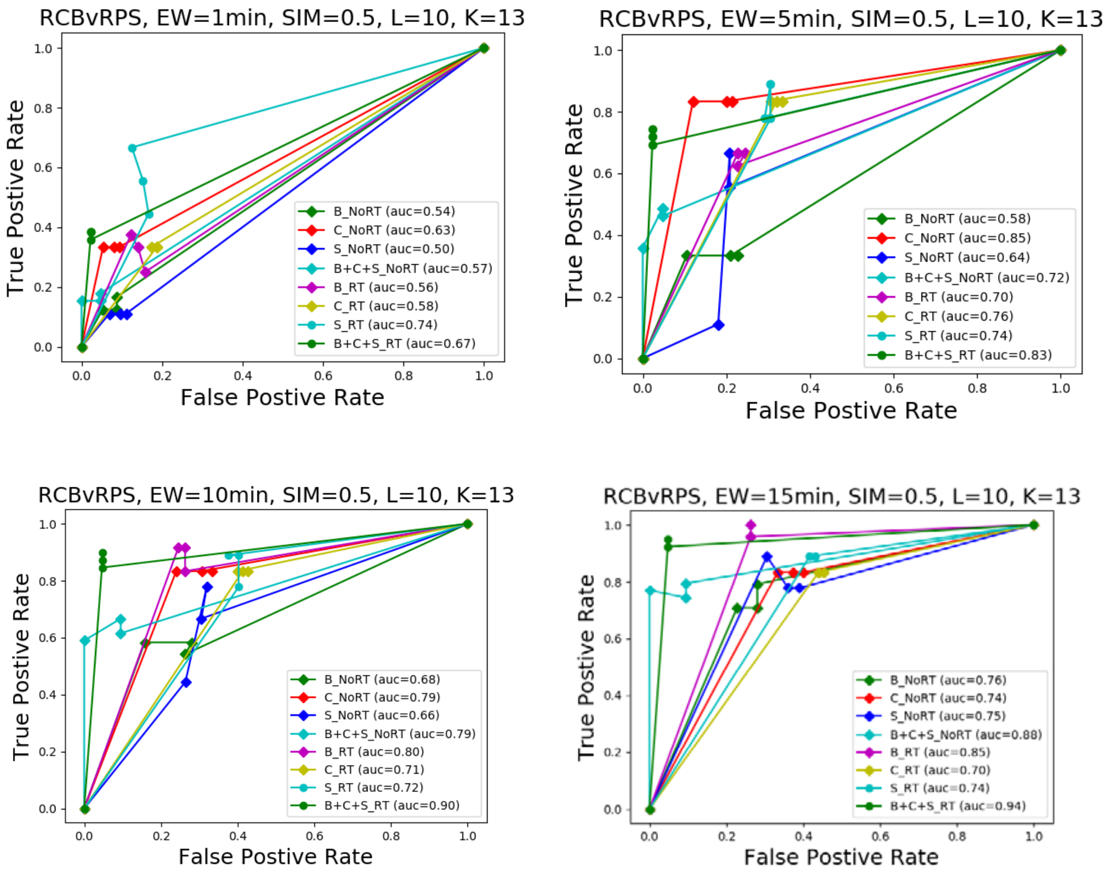


Figure 13. Detection performance for *all tweets* and *no retweets*

From the results shown in the ROC curves, it is apparent that the inclusion of retweets generally improves the accuracy of event detection, which we can observe easily based on the performance in major events detection (B+C+S). Because many twitter users have the habit of retweeting the tweet that is already posted by others. Also, they just want to approve that an event has actually happened, by way of retweeting. Therefore a large number of retweets creates a burst in the tweet volume which makes the event detection easy for the detector. Nevertheless, retweets do not help in detecting catch events. This may be due to a delayed retweeting of catch event long time after it has occurred. Whether retweets to be included or not for event detection can be sometimes decided based on the event we want to detect.

6.4 Limitations of SportsBuzzer

The performance of our real time event detection framework, *SportsBuzzer*, is impacted by various factors including latency or delay in the flow of signals from twitter users. There are three types of delays encountered in Twitter social media namely human delay, Twitter delay, and processing delay [3]. In this research work, we have addressed the issue of the processing delay. Processing delay occurs due to the processing time involved in data collection and analysis of a large volume of data. Our SportsBuzzer significantly reduces the processing time for the analysis of data by adopting LSH for implementing incremental clustering concepts. In our real time system that collects live tweets, data collection time is very minimal because of the recent fast processors.

7 Conclusion

In contrast to existing event detection approaches for sports domain, *SportsBuzzer*, a novel real-time event detection approach is presented in this paper. SportsBuzzer adopts LSH for discovering tweet clusters. A new cluster is created when an incoming live tweet is sufficiently dissimilar from existing clusters. An event is declared detected when the post rate of an active cluster exceeds the pre-defined threshold. Then, the event represented within the cluster is recognized utilizing our event lexicon for Cricket sports. Also a cluster is considered active if it contains at least two tweets. A cluster will be deleted once an event is detected from it or its life span is more than five minutes. We fix this time based on the assumption that at least one event would happen in one over which will last nearly five minutes.

Results of the extensive experiments demonstrated the efficacy of the LSH approach for event detection. As many twitter users take few minutes to post their tweets after an event has occurred, an evaluation window of five minutes was sufficient to detect most of the events. LSH effectively discovered tweet clusters with appropriate values for threshold, number of hash tables and number of projections. Influence of these parameters were also analyzed in the experiments. For a better tradeoff between speed and accuracy, a medium value for number of hash tables and number of projections is recommended.

In future, we will investigate whether we can improve the event detection by characterizing event lexicon as a dynamic lexicon instead of the present static lexicon. Similarly, we will study whether slope of the tweet rate curve can be exploited, instead of choosing the middle time for recognizing the detected event. Further, it would be interesting to explore other data structures that may speed up the clustering process, besides comparing our LSH approach with other widely popular clustering algorithms.

REFERENCES

[1]. Boyd, D. M and N. B. Ellison. Social network sites: Definition, history, and scholarship. Journal of Computer-Mediated Communication, 2007. 13(1): p. 210–230.

[2]. Atefeh, F and Khreich, W. A survey of techniques for event detection in twitter. Computational Intelligence, 2015. 31(1): p. 132-164.

[3]. Zhao, D and M. B. Rosson. How and why people Twitter: The role that micro-blogging plays in informal communication at work. In Proc. ACM International Conference on Supporting Group Work, GROUP ’09, ACM, New York, NY, 2009. p. 243–252.

[4]. Zhao, S., Zhong, L., Wickramasuriya, J and Vasudevan, V. Human as real-time sensors of social and physical events: A case study of twitter and sports games. ArXiv preprint, 2011. arXiv:1106.4300.

[5]. P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Proc. Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, Texas, USA, 1998. p. 604–613.

[6]. Hasan, M., Orgun, M. A and Schwitter, R. A survey on real-time event detection from the Twitter data stream. Journal of Information Science, 2017. 0165551517698564.

[7]. T. Sakaki., M. Okazaki and Y. Matsuo. Earthquake shakes Twitter users: real-time event detection by social sensors. In Proc. ACM WWW '10, 2010.

[8]. Y. Qu., C. Huang., P. Zhang and J. Zhang. Microblogging after a major disaster in China: a case study of the 2010 Yushu earthquake. In Proc. ACM 2011 conference on Computer supported cooperative work, 2011.

[9]. S. Vieweg., A. L. Hughes., K. Starbird and L. Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In Proc. ACM CHI '10, 2010.

[10]. J. Sankaranarayanan., H. Samet., B. E. Teitler., M. D. Lieberman and J. Sperling. TwitterStand: news in tweets. In Proc. ACM SIGSPATIAL, 2009.

[11]. Sakaki, T., M. Okazaki and Y. Matsuo. Earthquake shakes Twitter users: Real-time event detection by social sensors. In Proc. 19th International Conference on World Wide Web, WWW '10, ACM, New York, NY, 2010. p. 851–860.

[12]. Popescu, A. M and M. Pennacchiotti. Detecting controversial events from Twitter. In Proc. 19th ACM International Conference on Information and Knowledge Management, CIKM '10, ACM, New York, NY, 2010. p. 1873–1876.

[13]. Benson, E., A. Haghighi and R. Barzilay. Event discovery in social media feeds. In Proc. 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, HLT '11, Association for Computational Linguistics, Stroudsburg, PA, 2011. p. 389–398.

[14]. Becker, H., F. Chen., D. Iter., M. Naaman and L. Gravano. Automatic identification and presentation of Twitter content for planned events. In Proc. International AAAI Conference on Weblogs and Social Media, Barcelona, Spain, 2011.

[15]. Becker, H., M. Naaman and L. Gravano. Selecting quality Twitter content for events. In Proc. International AAAI Conference on Weblogs and Social Media, Barcelona, Spain, 2011b.

[16]. Massoudi, K., M. Tsagkias, M. De Rijke and W. Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In Proc. 33rd European Conference on Advances in Information Retrieval, ECIR'11. Springer-Verlag: Berlin, Heidelberg, 2011. p. 362–367.

[17]. Weerkamp, W and M. De Rijke. Credibility improves topical blog post retrieval. In Proc. ACL, Columbus, OH, 2008. p. 923–931.

[18]. Gu, H., X. Xie, Q. Lv, Y. Ruan and L. Shang. ETree: Effective and efficient event modeling for real-time online social media. In Proc. Web Intelligence and Intelligent Agent Technology, WI-IAT 2011, IEEE/WIC/ACM International Conference, 2011. 1: p. 300–307.

[19]. Valkanas, G and Gunopulos, D. How the Live Web Feels About Events. In Proc. In Proc. 22nd ACM International Conference on Information and Knowledge Management CIKM, 2013. p. 639–648.

[20]. Lee, R and K. Sumiya. Measuring geographical regularities of crowd behaviors for Twitter-based geo-social event detection. In Proc. 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks, LBSN '10, ACM, New York, NY, 2010. p. 1–10.

[21]. C. Li., A. Sun., and A. Datta. Twevent: Segment-based event detection from tweets. In Proc. ACM International Conference on Information and Knowledge Management, ser. CIKM '12. ACM, 2012. p. 155–164.

[22]. Marcus, A., Bernstein, M. S., Badar, O., Karger, D. R., Madden, S and Miller, R. C. TwitInfo: Aggregating and Visualizing Microblogs for Event Exploration. In Proc. CHI, 2011. p. 227–236.

[23]. Mathioudakis, M and Koudas, N. TwitterMonitor: Trend Detection over the Twitter Stream. In Proc. SIGMOD/ PODS, 2010. p. 1155–1158.

[24]. F. Alvanaki., M. Sebastian., K. Ramamritham and G. Weikum. Enblogue: Emergent topic detection in web 2.0 streams. In Proc. ACM SIGMOD, SIGMOD '11, New York, USA, 2011. p. 1271–1274.

[25]. Weng, J and Lee, B.-S. Event Detection in Twitter. In Proc. ICWSM, 2011. p. 401–408.

[26]. Shane Fitzpatrick. Improving new event detection in social streams. 2014. Master Thesis.

[27]. A. Ritter., Mausam., O. Etzioni and S. Clark. Open domain event extraction from Twitter. In Proc. 18th ACM SIGKDD, KDD '12, New York, USA, 2012. p. 1104–1112.

[28]. D. Zhou., L. Chen and Y. He. An unsupervised framework of exploring events on Twitter: Filtering, extraction and categorization. In Proc. AAAI Conference on Artificial Intelligence, 2015. p. 2468–2475.

[29]. J. Hannon., K. McCarthy., J. Lynch and B. Smyth. Personalized and automatic social summarization of events in video. In Proc. ACM IUI, 2011.

[30]. D. Chakrabarti and K. Punera. Event Summarization using Tweets. In Proc. AAAI ICWSM, 2011.

[31]. Becker, H., Naaman, M and Gravano, L. Beyond Trending Topics: Real-Wrold Event Identification on Twitter. In Proc. ICWSM, 2011. 11: p. 438–441.

[32]. A. J. McMinn and J. M. Jose. Real-time entity-based event detection for Twitter. In Proc. Experimental IR Meets Multilinguality, Multimodality, and Interaction. CLEF '15, Springer, 2015. p. 65–77.

[33]. Cataldi, M., Di Caro, L and Schifanella, C. Emerging Topic Detection on Twitter Based on Temporal and Social Terms Evaluation. In Proc. MDM/KDD, 2010. p. 4:1–10.

[34]. Petrovi´c, S., Osborne, M and Lavrenko, V. Streaming First Story Detection with Application to Twitter. In Proc. NAACL HLT, 2010. p. 181–189.

[35]. M. Hasan., M.A. Orgun and R. Schwitter. TwitterNews: real time event detection from the Twitter data stream. PeerJ PrePrints, 2016.

[36]. M. A. Russell. Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites. O'Reilly Media Inc, 2011.

[37]. M. S. Charikar. Similarity estimation techniques from rounding algorithms. In Proc. 34th Annual ACM Symposium on Theory of Computing, Montreal, Quebec, Canada, 2002. p. 380-388.