International Journal of Computer Applications (0975 – 8887)
Volume 180 – No.13, January 2018

Real Time Event Detection Adopting Incremental TF-IDF based LSH and Event Summary Generation

Jeyakumar Kannan
Dept. of Computer Science
Jamal Mohamed College (Auto)
Tiruchirappalli, India

Ar Md Shanavas
Dept. of Computer Science
Jamal Mohamed College (Auto)
Tiruchirappalli, India

Sridhar Swaminathan
Dept. of Computer Sci & Engg
Bennett University
Greater Noida, India

ABSTRACT
Recently, twitter users are leveraged to detect social and physical events such as *festivals* and *traffic jam* at real time. Real time event detection and summarization from Cricket sports is the process of detecting events such as *boundary* at real time from live Cricket tweet stream as soon as event happens and generating a quick game summary. This is an interesting, yet a complex problem. Because of the need for rapid detection of sports events and for the generation of a concise summary from huge volume of tweets for Cricket enthusiasts. In this paper, a novel framework is proposed for detecting key events from live Cricket tweets and for generating a game summary using the crawled tweets. Feature vectors of live tweets are created using incremental TF-IDF representation and tweet clusters are discovered using Locality Sensitive Hashing (LSH) where the post rate of each cluster determines the key event. A key event is recognized from that cluster using our domain specific event lexicon. Then, important moments from the crawled tweets are computed by identifying the spikes in the tweets volume. Top-*k* tweets from each moment are selected by ranking tweets on top-*k* words. Representative tweets from top-*k* tweets are identified using Jaccard similarity. The evaluation on 2017 IPL T20 Cricket live tweets using ROC measure shows that the proposed incremental TF-IDF based LSH approach detects key events with nearly 95% true positive rate and around 5% false positive rate. The proposed game summarization algorithm generates summaries which are readable and competitive to human tailored summaries.

General Terms
Machine Learning, Incremental Clustering, Social Media, Twitter, Data Analytics

Keywords
Event Detection, Incremental TF-IDF, Locality Sensitive Hashing, Live Sports Tweets, Event Summarization

1. INTRODUCTION
Modern day communication between people is happening with the use of high speed internet and social media sites such as Facebook and Twitter. Communication between people, groups has changed due to the invasion of online social media [1]. Microblogging services such as Twitter act as a famous platform for users and groups to share diverse digital content as short texts, links, images, or videos [2]. The tweets contain variety of information ranging from personal data, images to public data such as news, events where the information shared by them are based on their individual behaviors and interests [3]. In addition, Twitter is now used as a news platform to inform the world about live real-life events, viral news, crimes. It also helps everyone to be well informed with live data on different events. Lot of organizations are now utilizing the Twitter data for analyzing customer's opinions on products, social issues. Recent research as considered that the humans act as sensors who can be used for detecting live real-life events. Automation in event detection become unavoidable due to availability of huge amount of Twitter data and redundancy among Tweets describing the same events.

Recent research have shown that social and environmental events such as earthquakes, deaths of celebrities, and elections can be detected using Twitter [4]. Detecting events from Twitter in real-time has lot of applications in real life. Real-time event detection has challenges such as collection and processing of large volume of data in addition to the regular challenges such as limited tweet length, misinformation, typographical and grammatical errors. In addition, the underlying corpus used as training data should be updated frequently based on new stream of live data. It is also hard for users to follow large stream of tweets, and to filter out spams, irrelevant content and rumors. Thus, there is a need for an automatic summarization approach which can generate summary describing or highlights of events in a context.

Recently few research work have been proposed for the domain of sports. Detecting sports events need focusing on a smaller scale of data. Traditional event detection approaches will often fail to deal with the scalability issues. Also, only a few research work have been proposed for real time event detection in sports domain mostly on NFL soccer games. However, there is no real-time event detection and game summarization approach for the Cricket sports.

To address this demand, this paper proposes a novel event detection approach based on incremental TF-IDF and LSH techniques. In addition, we propose a novel event summary generation approach by using Jaccard similarity measure. To the best of our knowledge, ours is first of its kind that adopts incremental TF-IDF based LSH approach for Cricket sports domain to detect events from live tweets at real time. The major contributions of this paper are:

1. Unlike previous approaches which used offline datasets, we propose a novel approach to detect key events from live Cricket tweets using incremental TF-IDF based LSH method.
2. We also present a novel game summarization approach which generates a game summary utilizing the crawled tweets. Representative tweets are selected by scoring method and diversity of tweets is achieved using Jaccard similarity measure.

The rest of the paper is organized as follows. Section 2 discusses the related work on twitter event detection and game summarization. Section 3 introduces the event detection approach, while game summary generation approach is presented in section 4. Experimental results of the proposed approach are presented in section 5. Finally, section 6 concludes the paper with future work.

2. RELATED WORK

Existing event detection can be classified based on different domains such as social, political, environment and sports. For more detailed comparative study the readers are recommended to refer recent survey [2]. Whilst earlier work were focused more on physical events, recent approaches concentrate on social event detection. News topics are discovered [5] by clustering a large volume of data clustered tweets to discover news topics from the Twitter data. In the domain of sports, Hannon et al. [6] produced highlights of the world cup game using poste rate of tweets. These approaches works on offline dataset and detect events several hours after the actual event happened. Incremental clustering algorithm [7] is exploited for Twitter event detection where the similarities between event clusters and a tweet are calculated for detecting newsworthy events.

Very few work have been proposed for game summarization in Twitter. Chakrabarti et al [8] generated game summaries of rich events based on hypothesis that multiple events will share same structure. In the sports domain, Nichols et al [9] generated journalistic summary from tweets of World Cup football game. All of the above approaches focus only on Soccer sports and none of them on Cricket sports domain.

Since tweets flow continuously at real time, our system needs to update the dictionary and clusters dynamically. Our approach addresses these issues efficiently. It adopts incremental feature representation, online clustering and Jaccard similarity techniques for real time event detection and summarization.

3. REAL TIME EVENT DETECTION

Real time event detection is the process of detecting key events from live tweet streams at near real time. The proposed real time event detection framework is depicted in figure 1. After preprocessing of live tweets, feature vectors are created using incremental TF-IDF representation and clustered into buckets of LSH. Events are detected from active clusters based on the predefined post rate threshold. Then, events are recognized by applying our domain specific event lexicon. The following sections will explain the individual steps in detail. Finally, a summary of the game is generated based on Jaccard similarity between tweets.

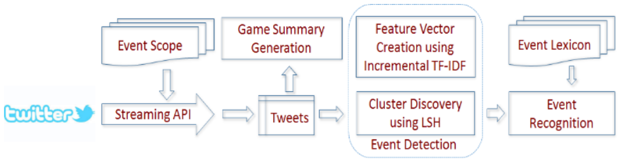

Figure1: System architecture

3.1 Preprocessing of live tweets

Our real time event detection system requires live tweets so as to detect and report all key events throughout the game time. Fortunately, game viewers and twitter users keep posting interesting tweets of game events such as *Boundary* which are leveraged by the system.

Twitter supports three types of API for gathering tweets for any application. As our approach needs live tweets continuously during the game time, we use Streaming API to crawl live tweets using game specific keywords. We use official keywords as hashtags for collecting live tweets. We run our system during IPL T20 2017 game between the teams Royal Challengers Bangalore (RCB) and Rising Pune Supergiant (RPS). Our system is 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 game summary generation.

The raw tweets are initially preprocessed for URLs, mentions, replies and stop words using Python NLTK toolkit. Then, feature vector is created for every preprocessed tweet, considering only unigram features of a tweet. Preprocessing has been an important step in the event detection process, otherwise the accuracy of detection will be affected at large.

3.2 Incremental TF-IDF for tweets features

Term Frequency - Inverse Document Frequency (TF-IDF) has been the popular term weighting schemes in text mining. TF-IDF is the product of term frequency (TF) and inverse document frequency (IDF). Here, TF describes the importance of a term in a document and IDF represents the importance of the term in the entire document collection. The TF-IDF weighing scheme assigns weight to term *w* in document *d* using equation 1 and 2:

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

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

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.

As ours is a real time system where tweets are flowing in continuously, the number of tweets in our corpus will be dynamic. Thereby, the document frequency will be different whenever a new tweet arrives. This impacts idf of a term and subsequently document clustering. Therefore, the solution is to incrementally re-compute idf value of each term a new tweet is arrived.

The new idf values are recomputed considering the past tweets and the tweets of the current chunk (say, 100 tweets) incrementally. Hence, the equations for incremental TF-IDF are rewritten as shown below (equation 3 and 4):

$$tf * idf_i\,(w.d) = tf(w,t) * idf(w) \tag{3}$$

$$idf(w) = log\,\frac{N_i}{df_i(w)} \tag{4}$$

Here, *tf-idf*$_i$*(w,t)* is the tf-idf value for term *w* in tweet *t* from tweets of past chunks and current chunk. The *idf(w)* is an inverse document frequency of the term *w*. The *df*$_i$*(w)* is the number of tweets N_i (of previous chunks and the current chunk) in which the term *w* occurs at least once. Finally, tweet features are mapped to the dictionary in order to generate its tweet feature vector where weight is assigned to dictionary terms that appear in the tweet.

3.3 Event detection using Incremental clustering

We first introduce the idea of approximate nearest neighbors and locality sensitive hashing technique that clusters feature vectors created from the preprocessed live tweets. Also, we explain the construction of signatures for tweet feature vectors for indexing buckets of LSH. Finally, we explain how key events are detected from tweet clusters based on the post rate of a cluster.

Given a set of N points P = {P_1,P_2,P_3,…,P_N} represented as a matrix M and a query point Q, a nearest neighbor search computes the distance between all points in P to Q and selects the one point $P_i \in$ P which is the closest to Q ∈ M [10]. The nearest neighbor approach is computationally expensive for high dimensional data. To alleviate this problem, a variation known as Approximate Nearest Neighbor (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 5.

$$\forall P' \in P, d(P', Q) < (1+\varepsilon) d(P', Q) \qquad (5)$$

Here, $d(P', Q)$ is the distance between P' and Q and $(1+\varepsilon)$ is a constant factor [10]. Locality Sensitive Hashing (LSH) [10] is a popular approach to address the problem of ANN search.

The key assumption of LSH is that 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 into the same bucket [10]. Recently, LSH has become a successful solution for clustering large social media streams, such as tweets. 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.

Our proposed methodology for discovering clusters of tweets uses hash table based LSH for computing nearest neighbors. In our approach, each tweet feature vector is hashed using k hash functions and stored in L buckets. We use hash table to represent a bucket. The hash value of tweet feature vectors acts as an index of a hash table. The nearest neighbor of a feature vector of an incoming tweet is found by retrieving similar tweet feature vector from each bucket and selecting the one with a highest cosine similarity distance value. The cosine distance is a dot product of tweet feature vectors normalized by their norms. The cosine distance will be 1 if two feature vectors are parallel and 0 if orthogonal to each other.

The *k*-bit signature for each tweet feature vector is generated using the hash function proposed by Charikar [11] (as shown in equation 6). It 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.

$$h(u) = \begin{cases} 1, if\ r.u \geq 0 \\ 0, if\ r.u < 0 \end{cases} \qquad (6)$$

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 7.

$$\cos(\theta(u,v)) = \cos((1 - \Pr[h(u) = h(v)])\pi) \qquad (7)$$

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.

The proposed algorithm for incremental TF-IDF based LSH for event detection and recognition is depicted in figure 2. Here, incoming tweets are preprocessed, feature vectors are created for the preprocessed tweets based on incremental TF-IDF representation and clustered using LSH. Simultaneously, IDF values are updated once the number of incoming tweets reaches the maximum chunk size. Post rate of a cluster is computed which is the number of tweets at a given time. If it is above a predefined threshold, an event is declared detected. A cluster is deleted once an event is detected from it. An event that has occurred most number of times in the representative tweets of a cluster, is the recognized event.

Algorithm: Real time event detection and recognition
Input: Live tweets, similarity threshold ST, buckets L, signature length K, post rate T, chunk size CHUNK
Output: Event *name* and its *tweets*

```
1:  create event lexicon for pre-determined event types
2:  build TF-IDF dictionary □ 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:          re-build TF-IDF dictionary □ using CHUNK tweets
10:         construct tweet feature vector tv for t using □
11:         create k-bit signature ts for tv
12:         for each bucket i ϵ L do
13:             get collision for ts
14:             add tv with key ts in ht[i]
15:         end for
16:         get nearest neighbor NN for tv from collisions
17:         if similarity(tv, NN) < ST then
18:             create new cluster c
19:             addTweetVectorToCluster(tv, c)
20:         else
21:             if tv not in NN's cluster c_NN then
22:                 addTweetVectorToCluster(tv, c_NN)
23:             end if
24:         end if
25:     end for
26: until connection closed

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

Algorithm: Game summary generation
Input: Offline tweets, k, m
Output: Game summary

```
1: compute peaks using tweet frequency per second
2: Let moments be filtered peaks using 3*median + std. deviation
3: for each moment ϵ moments do
4:     select tweets around moment
5:     find top-k words from tweets
6:     get top-k tweets ranked on top-k words from tweets
```

```
7:      rank top-k tweets using Jaccard similarity
8:      display m tweets from ranked tweets
9: end for
```

Figure 2: Proposed algorithm

We obtain an optimal number of clusters by characterizing the size and life span of each cluster. 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 would 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 at near real time to the interested people. With optimal clusters, our online incremental clustering approach detects and recognizes key events at near real time.

3.4 Lexicon based event recognition

Our real time event detector continuously monitors clusters and computes the post rate of each cluster. If the post rate of a cluster is more than the predefined threshold, then the system assumes that some key event, defined in our event lexicon, has happened in that cluster. Therefore, the event recognizer computes the tweet frequency of each key event and selects an event whose tweet frequency is the maximum. Tweet frequency of an event is the number of tweets that contain this event.

Care has been taken to design an event lexicon for Cricket sports, as event recognizer recognizes key events that are defined in the event lexicon. Since tweeting style of every game viewer is unique, describing an event with proper event name is very crucial. That is, event names should be more descriptive, as every game viewer tweets about the same event in different ways, using different words. Our event lexicon 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.

The domain specific event lexicon is the preferred representation for event recognition at real time. Unlike statistical event recognition models which require training, our lexicon based recognition is free from training and easy for implementation. Further, there is no training data for some applications such as *celebrity deaths* and *terrorist attacks*, which results lexicon based approach a natural choice.

Reporting of duplicate events has been an important issue for the event detector. The LSH approach solves this issue by clustering similar tweets into the same bucket. It also prevents the creation of new clusters until a new tweet is sufficiently dissimilar from existing clusters. As we adopted incremental TF-IDF features for live tweets representation, the continuous update of the vector space model greatly improves the similarity computation between a new tweet and tweets of existing clusters and hence the most similar nearest neighbor is found for the incoming new tweet. Since, our incremental TF-IDF achieves better similarity than the standard TF-IDF model, all similar tweets will go into the same cluster and thus the number of event alerts is minimized.

4. GAME SUMMARY GENERATION

Throughout game time, many interesting events (aka, moments) happen. At the end of the game, many sports enthusiasts would be interested to know a gist of the game. Game summarization is nothing but a generation of textual summary utilizing important key words from important tweets of the entire game. The proposed game summarization workflow is depicted in figure 3.

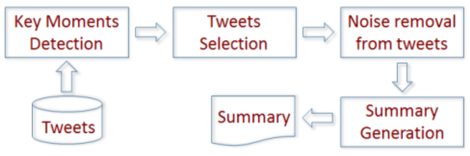

Figure 3: Game summarization workflow

First, key events are detected based on the spikes in the tweets volume. Then, game moments are selected by considering the tweets of the spikes that are above the threshold. Representative tweets of the game moments are preprocessed for noise such as spam, URLs and non-English tweets. They are ranked on top-*k* words using Jaccard similarity measure and tweet summary is presented to users.

4.1 Important moments detection

In addition to detecting key events at real time, we also simultaneously archive crawled tweets as JSON objects for offline processing. We leverage the offline tweets for generating a game summary once the game is finished. Cricket sports events consists of a sequence of moments (for simplicity, we call *moments* as just events) where each moment represents the actions by players, referee or audience of the game. In the Twitter streams, sudden increases or spikes in the volume of tweets indicate the occurrence of some key events in the game that the game viewers found interesting.

Shamma et al. [12] suggested that key moments can be detected from Twitter when the volume of tweets increases sharply. Over the game time, the sharp increase of tweets volume happens several times as shown in the tweets volume graph of our game RCBvRPS (figure 4).

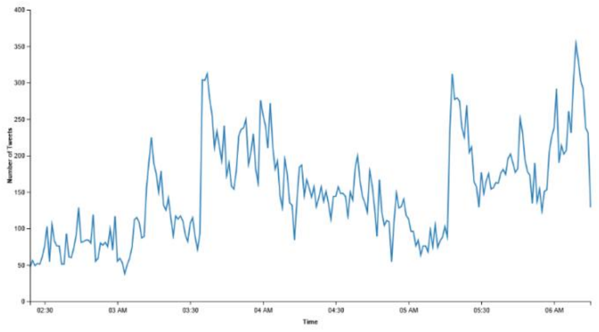

Figure 4: RCBvRPS tweet volume graph

Key moments can be detected from the volume of tweets in two ways – either by considering the absolute value of the volume or the spike in the volume. We choose the later approach because sometimes the tweet volume stays high for several minutes and have several local peaks. Also, game viewers sometime miss tweeting some key events resulting a less traffic.

Our algorithm to detect the key moments of a game is based on the approach described in [9]. We consider a granularity of minutes for characterizing the tweet volume. Our algorithm computes the threshold for the entire game considering all slopes for the entire tweet volume. We compute the threshold for the slope with the formula *3 * median + standard deviation*. We have tried several variations and empirically arrived at this equation, which provides best results for our domain. After identifying all slopes from offline tweets that exceed our threshold, the list of spikes which correspond to the key moments of a game is generated. We consider spikes as points in the tweet volume graph where the slope changes from positive to negative and the absolute number of tweets in that minute exceeds our threshold.

4.2 Tweets selection

After generating all spikes corresponding to key moments of the game, the next task is to identify representative tweets for each key moment that will be used to generate the game summary. It is important that the selection of representative tweets greatly influence the quality of the game summary. Obviously, a very large spread of tweets in each key moment will result into more false positives (ie. selecting a tweet that does not describe a key moment). On the other hand, a narrow spread of tweets in a key moment will lead to more false rejections (ie. ignoring a tweet that describes a key moment). Our algorithm selects all tweets that are near to the peak of each key moment. An interval of k-minutes defines the range for including the tweets. We observed best results when the value of k =2. Also, our algorithm assumes that the interval need not be symmetric about the peak of a key moment.

4.3 Noise removal from tweets

Noise removal has been a very crucial preprocessing step for generating game summaries. Once all representative tweets of important key moments are selected, noise removal techniques should be applied to filter out spam and other irrelevant tweets. We remove tweets containing up to 3 words, as these tweets may not contribute to describing the game summary. Spam tweets are removed using a dictionary of common terms. Stop words are removed using NLTK Python library. We also consider URLs and other personal pronouns as noises and our algorithm removes all of them.

4.4 Game summary presentation

Upon removing all noises from the representative tweets, the final step is to generate a summary for the game. Our approach follows a simple scoring method to rank the tweets. Initially, our algorithm computes top-k words (T) based on their frequency from the tweets of a moment. If the list of top-k words contain any word whose frequency is up to 2, then the algorithm deletes those words from the list. It also deletes spam words and other common words of Cricket sports that do not have discriminating power. The score for each representative tweet of a key moment is calculated (as shown in equation 8) based on the occurrences of top-k words in each tweet t.

$$Score(t) = \sum_{i=1}^{n} freq(tt_i) \quad (8)$$

Where $tt_i \in T$. Note that the score for a term of a tweet (tt) that does not appear in T will be 0. Based on the tweet score, our algorithm ranks all tweets and selects the top-m tweets as part of the game summary.

However, this simple scoring method suffers from some weaknesses. For example, if scores of several tweets are same, then all these tweets will be considered as top ranked tweets for this specific moment. Thereby, all these tweets will be part of the result. Therefore, the algorithm should choose diverse tweets for a key moment. The algorithm selects a set of k diverse tweets from the ranked top-m tweets using Jaccard similarity measure. The Jaccard similarity between tweets t_i and t_j is denoted as follows (equation 9):

$$Jaccard(t_i, t_j) = \frac{|W_i \cap W_j|}{|W_i \cup W_j|} \quad (9)$$

Here, W_i and W_j refer to the set of words in tweets t_i and t_j respectively. The algorithm considers all possible sets of k tweets from the ranked top-m tweets, computes a sum of Jaccard similarity scores for each set of k tweets and selects the set with a least similarity score. This way the algorithm selects k diverse tweets and prevents redundant tweets of each key moment.

5. EXPERIMENTAL RESULTS

In this section, we will present the experimental results of our incremental TF-IDF based key events detection approach. We evaluate the proposed approach using the live tweets of IPL T20 2017 cricket sports. The proposed approach has been implemented in Python. The evaluation proves that the proposed approach can detect events at real time with nearly 95% true positives and around 5% false positives. We will also present the results of our Jaccard similarity based game summarization approach. In the following subsections, we will now present the data set, evaluation criteria and parameter setup and evaluation results.

5.1 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 contains 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 details of tweets (T) and retweets (RT) of RCBvRPS game. 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. The ground truth events for the RCBvRPS game include 24 boundaries, 6 catches, 9 sixes and 42 other events.

Table 1. RCBvRPS game statistics

	Total	**Total min**	**Mean/ min**	**Min/ min**	**Max/ min**	**Std. Dev**
T	34967	232	150.7	38	354	67.5
RT	16162	232	69.6	13	176	34.5

5.2 Evaluation criteria

Using RCBvRPS game from 2017 IPL T20 cricket season, we illustrate the effectiveness of our incremental TF-IDF based LSH approach for event detection 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 10 and 11) are computed as follows:

$$TPR = \frac{TP}{TP+FN} \quad (10)$$

$$FPR = \frac{FP}{FP+TN} \quad (11)$$

For a particular study, different set of TPRs and FPRs are computed with various parameter settings. The parameters for

LSH are kept fixed for all experiments. We fix nearest neighbor similarity threshold as 0.5, number of hash tables as 10 and number of projections as 13 for all experiments. We consider three post rates namely, 0.2, 0.5 and 0.8. The tweet chunk size is fixed as 100, 250, 500, 1000 or 2000 tweets. 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 (AUC) are calculated. The AUC represents the accuracy of the event detector. A high AUC denotes a high true positive rate and low false positive rate while a low AUC denotes a low true positive rate and high false positive rate.

5.3 Results of event recognition

We evaluate the proposed incremental TF-IDF approach 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 chunk sizes for tweets on event detection. We also compare our incremental TF-IDF based LSH approach against the naïve TF-IDF based LSH approach.

5.3.1 *Performance on detecting events*

Figure 5 shows the ROC curves that illustrate the performance of our incremental TF-IDF based 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 a tweet chunk size of 100 and for different evaluation window sizes such as 1, 5, 10 and 15 minutes.

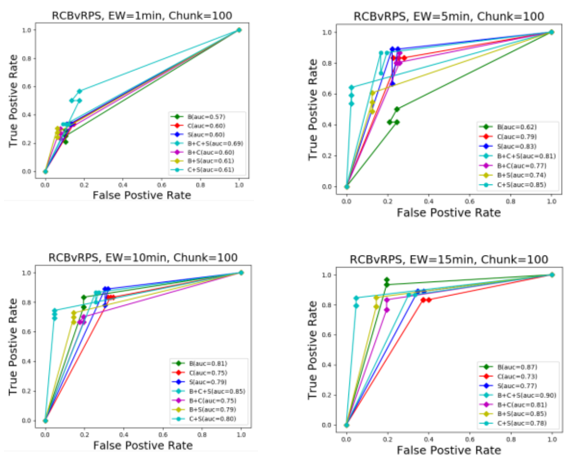

Figure 5: Detection performance on individual events

Results show that our proposed approach delivers good performance in detecting game events for chunk size of 100 tweets. Within 1 min evaluation window, almost all individual events are showing similar performance where the combined events are also detected similarly where major events (B+C+S) show slightly better and appreciable accuracy. When size of the evaluation window increases, boundary is detected well, as twitter users reported this event with a smaller delay. This user delay is treated normal as the event boundary is considered frequent and rapidly happening. Due to a high initial excitement among twitter users, *catch* and *sixer* are detected well within 5 minutes. The performance slightly decreases for 10 and 15 min windows. This is due to users' behavior in delayed reporting of interesting events such as *catch* and *sixer* even after the actual happening of the event for long time. Major events (*boundary+catch+sixer* combination) of the game are also detected well with our approach. Other major events also perform better for a 5 min window. From these graphs, we can observe that almost all key events are detected with a decent accuracy (with 85 percent true positives and less than 20 percent false positives) within an evaluation window of 5 minutes itself. So, we can conclude that most of the key events are detected and reported well even within 5 minutes from the actual happening of those events.

5.3.2 *Detection under different chunks*

Figure 6 shows the ROC curves that illustrate the performance of incremental TF-IDF based LSH approach for a chunk size of 250 tweets. The evaluation is conducted for a tweet chunk size of 250 and for different evaluation window sizes such as 1, 5, 10 and 15 minutes.

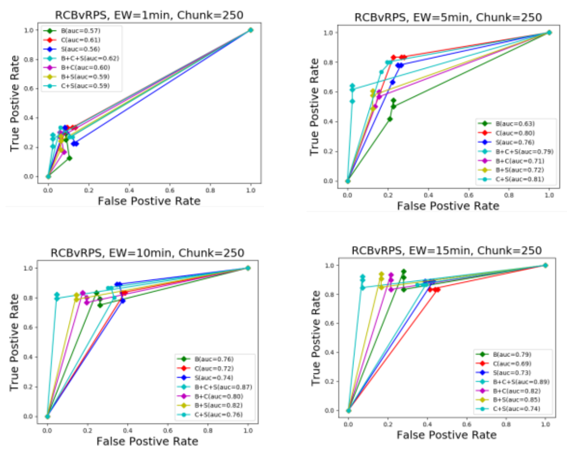

Figure 6: Detection performance with chunk size 250

These results under parameter setup of chunk size 250 show that all events (*boundary*, *catch* and *sixer*), including major events are detected well within 5 minutes itself. For *boundary*, true positives improves consistently when the size of the evaluation window increases. Because, game viewers tweet this event with a smaller delay. Other individual events also show performance improvement in most cases when the evaluation windows increases. Major event (Boundary+Catch+Sixer combination) is detected best with over 90 percent true positives in our approach.

Figure 7 shows the ROC curves that illustrate the performance of incremental TF-IDF based LSH approach for a chunk size of 500 tweets. The evaluation is conducted for a tweet chunk size of 500 and for different evaluation window sizes such as 1, 5, 10 and 15 minutes.

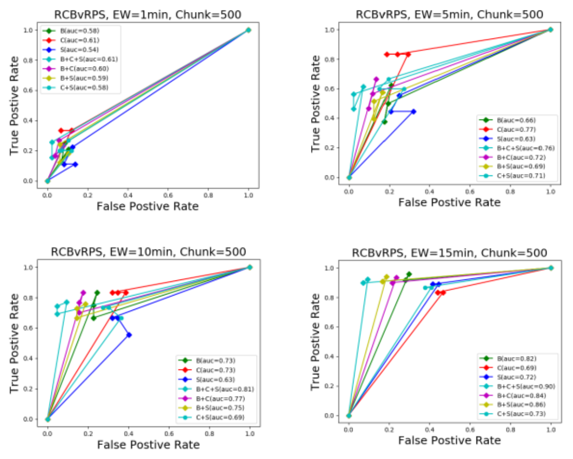

Figure 7: Detection performance with chunk size 500

Results show that the event *catch* is detected well within 5 mins itself. The event *boundary* shows the best performance when the size of the window increases. Similarly, *boundary* and *sixer* also show consistent performance for increased window sizes. Major event (B+C+S combination) is detected well with over 90 percent true positives and less than 5 percent false positives, within 15mins. Other major events are also detected well with over 80 true positives.

Figure 8 shows the ROC curves that illustrate the performance of incremental TF-IDF based LSH approach for a chunk size of 1000 tweets. The evaluation is conducted for a tweet chunk size of 1000 and for different evaluation window sizes such as 1, 5, 10 and 15 minutes.

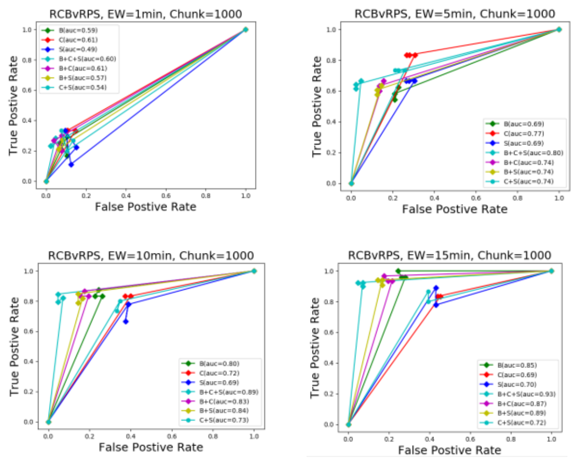

Figure 8: Detection performance with chunk size 1000

Results show that the performance of events (B, C and S) are similar to the performance with a chunk size of 500 tweets. All major events are detected well when the size of the evaluation window increases, except C+S combination, where the performance degrades slightly for the evaluation window of 15 mins.

Figure 9 shows the ROC curves that illustrate the performance of incremental TF-IDF based LSH approach for a chunk size of 2000 tweets. The evaluation is conducted for a tweet chunk size of 2000 and for different evaluation window sizes such as 1, 5, 10 and 15 minutes.

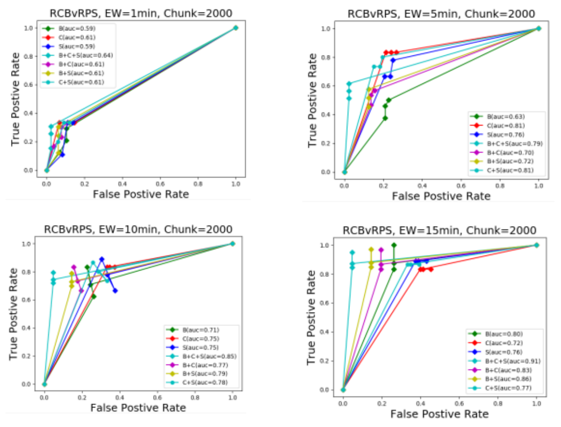

Figure 9: Detection performance with chunk size 2000

Results show that the event *sixer* is detected well within 5 minutes and its performance remains constant for other window sizes, as the tweeting behavior of audience remains same. The event *catch* is detected better within 5 minutes and afterwards its performance degrades slightly. All major events are detected well with over 85 percent true positives within 10 minutes itself.

5.3.3 *Influence of chunk size*

Figure 10 shows the ROC curves that illustrate the performance of our incremental TF-IDF based LSH approach in detecting key events for different evaluation windows. The influence of various chunk sizes (100, 250, 500 and 1000) on event detection is evaluated for key events and a combination of key events under different evaluation window sizes such as 1, 5, 10 and 15 minutes.

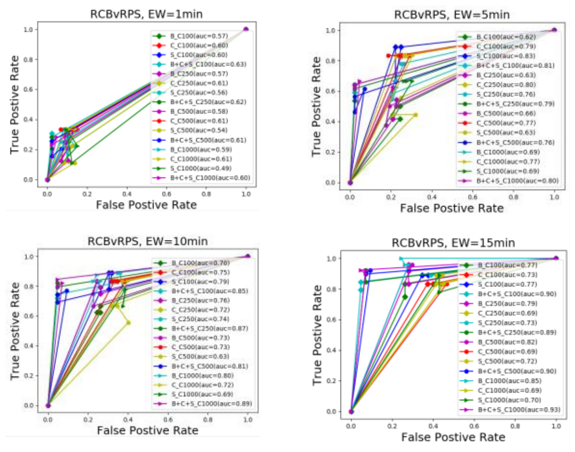

Figure 10: Influence of chunk size on event detection

From the results shown in the ROC curves, our proposed approach achieves better accuracy when the chunk size is small which can be noticed easily in the case of major events (B+C+S). The intuition is that when the chuck size is small, the document frequency of the dictionary terms will be updated frequently and hence TF-IDF values will improve during retraining with smaller corpus. Practically, for larger chunks, our approach performs similar to LSH event detection approach with naïve TF-IDF features. Also, the computational complexity increase when the chunk size is set too low. From the experiments it can be seen the chunk sizes 250 and 500 perform fairly and can balance the speed-accuracy tradeoff.

5.3.4 *Performance of w/o incremental tfidf*

Figure 11 shows the ROC curves that illustrate the performance of our incremental TF-IDF based LSH approach against the naïve TF-IDF based LSH approach in detecting key events and a combination of key events under different evaluation window sizes such as 1, 5, 10 and 15 minutes.

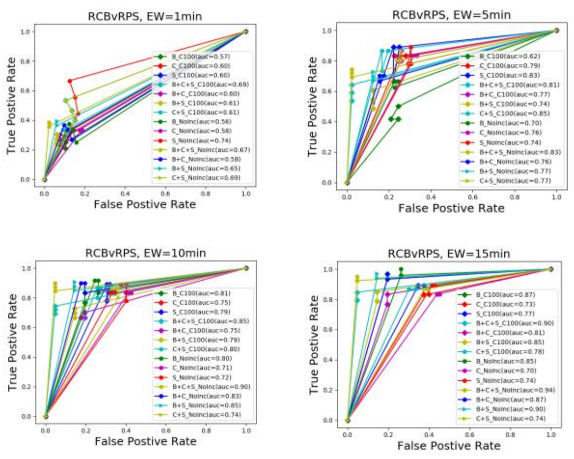

Figure 11: Performance w/o incremental TF-IDF

From the results shown in the ROC curves, it is apparent that incremental TF-IDF based LSH generally improves the accuracy of event detection, which we can observe easily based on the performance of *catch* event detection. Similarly, the event *sixer* is detected well within 5 minutes itself for our proposed approach. However, performance of *sixer* degrades for No Incremental TF-IDF approach consistently for the evaluation window of 5, 10 and 15 minutes. The *boundary* event is detected well within 5 minutes and the accuracy is constantly better with a permissible small delay in tweeting. Similar to *boundary* event, major event (C+S) is also detected well within 5 minutes. The accuracy is consistently better for the evaluation window of 5, 10 and 15 minutes. For other cases, the standard TF-IDF performs slightly better than our proposed approach, because of the amount of duplicates which is 90 percent. This is something undesirable for any event detector. However, processed tweets are generally well represented in our proposed approach. This helps the event detector to easily find the nearest neighbors and eventually reduces the false alarm rate in event detection.

5.4 Results of game summary generation

We evaluate the performance of the proposed Jaccard similarity based game summarization approach with the crawled tweets. We evaluate the proposed approach subjectively. As we do not have any gold standard for evaluating the summarization approach, ROUGE summarization approach is not used. The articles in many sports websites contain only overview of a game and not any moment wise summary. Hence, we create a manual summary and compare against the results of our approach as shown in table 2. It can be noticed that our game summary is meaningful and non-redundant. It also achieves good coverage of many key moments of the game.

Table 2. Manual summary *vs* generated summary

Manual Summary	RPS asked to bat first, RPS 50 in 5 overs, Rahane bowled out by Badree, Tripati caught out by Kohli, Amazing four by Dhoni, Super six by Dhoni, Dhoni bowled out by Watson, Smith bowled off immediately, Manoj Diwari struck two sixes and three boundaries, Virat Kholi and AB de Villers dismissed in eleventh over
Generated Summary	Bangalore opt to bowl, RPS have raced off to 50 in 5 overs, Badree strikes and Rahane has been cleaned up, Stunning catch at cover, One more amazing shot by Dhoni goes for a 4, Huge six from Dhoni, The big fish is caught - msd bowled out on watto’s delivery, Dhoni and Smith bowled off back to back deliveries

6. CONCLUSION

In contrast to existing event detection approaches, a novel incremental TF-IDF based LSH approach for real time event detection and Jaccard similarity based game summarization methods are presented in this paper. The proposed approach first detects key events from live tweets and generates a game summary using the crawled tweets immediately after the game.

Since tweets flow continuously, our system needs to update the dictionary and clusters dynamically. Our proposed approach addresses these issues efficiently by adopting incremental feature representation and online clustering techniques. Also, representative tweets are ranked using scoring and diversity for game summary is achieved using Jaccard similarity.

Results of the extensive experiments demonstrated the efficacy of the proposed event detection and summarization approach. Most of the events were detected with a chunk size of just 100 tweets. It shows that frequent re-computation of IDF values improves the accuracy of event detection. We have investigated the influence of various tweet chunk sizes. Based on the comparison, we can note that incremental TF-IDF approach, not only improves the accuracy, but also minimizes the amount of duplicate event reporting greatly. In future, we would investigate the ways of improving feature representation and clustering methods.

7. REFERENCES

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

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

[3] Zhao, D, Rosson, M.B. 2009. 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, 243–252

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

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

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

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

[8] Deepayan Chakrabarti, Kunal Punera. 2011. Event Summarization Using Tweets. ICWSM

[9] Nichols, J, Mahmud, J, Drews, C. 2012. Summarizing sporting events using twitter. In proc. ACM IUI

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

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

[12] Shamma, D.A., Kennedy, L., Churchill, E.F. 2011. Peaks and persistence: modeling the shape of microblog conversations. In Proc. of CSCW