Summarization Of Football Events using Tweets

B.Tech Project submitted in partial fulfillment
of the requirements for the degree of
Bachelors of Technology
in
CSE
by

Sandeep Kasa , T.V Ravi Teja
201301145 , 201301047
sandeep.kasa@students.iiit.ac.in , venkata.takkella@students.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA

December 2016

Contents

Chapter Page

1 Introduction 3

2 Motivation 4

3 Challenges 4

4 Related Work 5

5 Procedure 7

5.1 SummAll Text 7

5.2 SummTimeint Text 8

5.3 SummHmm Text 8

5.4 Spike Detection 10

6 Work Done 11

7 Results and Evaluation 17

8 Future Work 20

9 Conclusions 20

10 References 20

11 Miscellaneous 21

Learning Various Twitter Problems

11.1 Discovering Similar users and grouping them on Twitter 21

11.2 Click-through prediction for advertising on Twitter timeline 21

11.3 Localized twitter opinion mining using sentiment analysis 22

12 Important Links 22

I. INTRODUCTION

The status updates posted to social networks, such as Twitter and Facebook, contain a myriad of information about what people are doing and watching. During events, such as sports games, many updates are sent describing and expressing opinions about event.

Twitter has become exceedingly popular, with hundreds of millions of tweets being posted every day on a wide variety of topics. This has helped make real-time search applications possible with leading search engines routinely displaying relevant tweets in response to user queries. Recent research has shown that a considerable fraction of these tweets are about "events", and the detection of novel events in the tweet-stream has attracted a lot of research interest. However, very little research has focused on properly displaying this real-time information about events. For instance, the leading search engines simply display all tweets matching the queries in reverse chronological order. In this paper we argue that for some highly structured and recurring events, such as sports, it is better to use more sophisticated techniques to summarize the relevant tweets.

Sports is particularly suited to this sort of analysis since there is an ever-present and active audience for all sports. By analysing the content of their conversation (both volume and vocabulary) we show that it is possible to achieve very good event detection and classification within sports video through faster but equally accurate methods than audio-visual analysis alone. We also show that we can display what the audience themselves found to be the most interesting and exciting moments; this is not possible using traditional audio-visual approaches.

On Twitter, user's compose and send short messages called "tweets", putting the medium to a wide array of uses.Recent research has shown that one of the big use-cases of Twitter is users reporting on events that they are experiencing. However, the state of the art in real-time usage of this stream of event-tweets is still rather primitive. In response to searches for ongoing events, today's major search engines simply find tweets that match the query terms, and present the most recent ones. This approach has the advantage of leveraging existing query matching technologies, and for simple one-shot events such as earthquakes it works well.

However, for events that have "structure" or are long-running, and where users are likely to want a summary of all occurrences so far, this approach is often unsatisfactory.Consider, for instance, a query about an ongoing game of Football (our project is working only on a football event). Just returning the most recent tweets about the game is problematic for two reasons: (a) the most recent tweets could be repeating the same information about the event (say, the
most recent "assist"), and (b) most users would be interested in a summary of the occurrences in the game so far.This approach can clearly be improved upon using information about the structure of the event. In the case of Football,this corresponds to knowledge about the game play.

This motivates our goal of summarizing long-running structure-rich events.We assume that a new event has been detected; our goal is to extract a few tweets that best describe the chain of interesting occurrences in that event.We want our approach to learn from previous games in order to better summarize a recent (perhaps even an ongoing) game; queries about this game can then present a full summary of the key occurrences in the game, as well as the latest updates.

The problem can be defined as summarizing long-running structure-rich football events using tweets on social networking site 'Twitter' . Detect all the sub-events and extract only a few relevant tweets that describe interesting occurrences in that event .

II. MOTIVATION

The detection and easy navigation of highlights within sports media is something that is both appreciated and desired by viewers . The combination of both audio and visual video content features has been able to successfully detect many of the most important events within a match .

Our main motivations for using Twitter as a source of information are speed and enrichment. By looking at the words that people are using within their tweets, we are able to create tags that best describe and classify the highlights to which they are temporally aligned. Performing audio- visual analysis is time-consuming; Twitter provides a real-world approximation of the interest, allowing for the real-time segmentation and display of highlights.Using the Twitter stream alone however ignores any of the content features themselves, leading to disorientating initial frames that appear to jump into the middle of the action. To counteract this, we combine the tweet information with a video shot boundary detection algorithm (SBD) so as to provide more intelligently bounded highlight videos.

Our proposed solution was a two-step process. We first design a modified Hidden Markov Model that can segment the event time-line, depending on both the burstiness of the tweet-stream and the word distribution used in tweets. Each such segment represents one distinct “sub-event”, a semantically distinct portion of the full event. We then pick key tweets to describe each segment judged to be interesting enough, and combine them together to build the summary.

However , the assumption of the availability of multiple similar events may not be possible in practice , especially at the granularity of the moments they considered within their events.For example, an HMM is unlikely to capture all possible sequences of sub-events in an event. Also certain vocabulary items, such as the names of participants, cannot be learned unless the same two teams have played multiple times, which is not common in some sports or in non-sporting events.Our new approach is not limited by the availability of multiple similar events. Our approach is also unsupervised, whereas supervision is required to train an HMM model.

III. CHALLENGES

Our algorithm must tackle several challenges while implementing the problem to summarize the tweets of a football event.

(1) Events are typically “**bursty**”. Some types of sub-events generate far more tweets per unit time than others. Our algorithm should balance the number of tweets picked for the summary from low activity and bursty periods.

(2) Separate sub-events may not be **temporally far apart**. Our algorithm should solve the problem by automatically learning language models for common types of sub-events. This allows it to separate different types of sub-events even when they are temporally close.

(3) Tweets are **noisy**. Tweets are very short (less than 140 characters) and noisy, rife with spelling mistakes. Our algorithm should achieve robust results in the face of these issues.

(4) **Strong empirical results**. We empirically compare our model against several baselines; in particular, we should show that our algorithm achieves better recall over the set of important occurrences, and also gives a better sense of the context of these occurrences and the players associated with them.

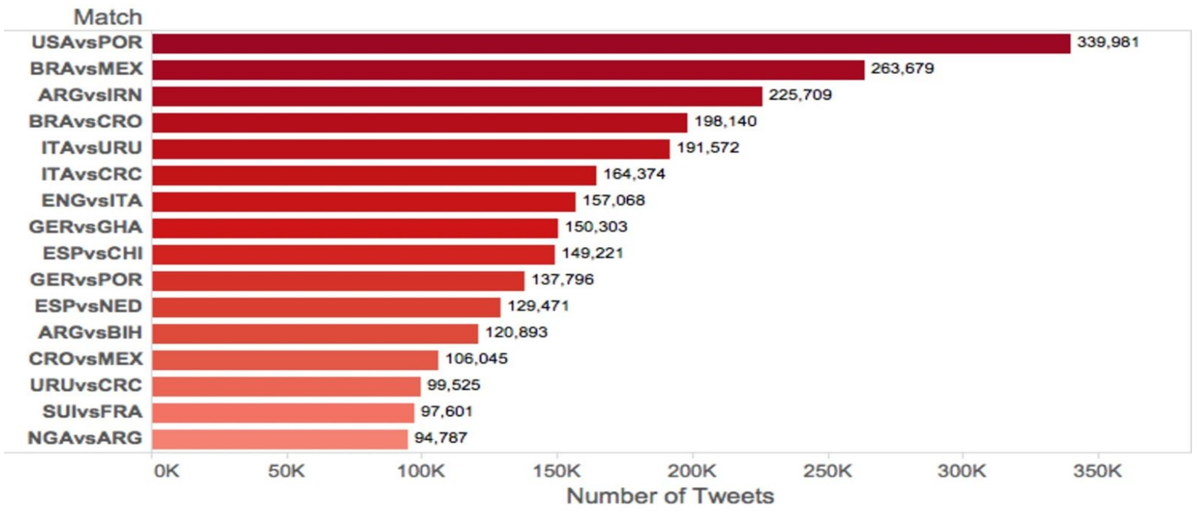
IV. RELATED WORK

Twitter allows its users to publish updates or 'tweets' about any topic they like, but limits the length of these tweets to 140 characters. The Twitter API allows for the focussed search of this public stream based on issued queries. Similar to standard information retrieval systems like Google, Bing, and Yahoo!, all tweets returned in answer to these queries contain the query terms or hashtags('#' keyword). We filter the Tweets from the public timeline by hashtags and only retrieving those relevant to each match.

Twitter has already been used to show large correlations between its tweeting population and the population and attitudes of the real world (Sakaki, Okazaki, and Matsuo 2010).Shamma et al.'s work (Shamma, Kennedy, and Churchill
2009) is closest to our own: they use the volume of tweets from people watching the first presidential debate in 2008 as an indication of the level of interest. Their research centred around a live televised event much the same as our research.
The event itself however has an important difference; political debate can give rise to heated conversation and discussion. With sports we have found that interest (as signalled by the volume of tweets) is more intermittent, centred on the key moments from within the sports event. We consider any event that changes the score of the sports event, or bookings or other disciplinary actions by the referee(s) as important.

For the purposes of this experiment we identify tweets referring to Football games by looking for certain hash-tags . Let's take a football match , "barcelona" vs "real madrid" as an ongoing event . In particular, we plot the number of occurrences of tweets containing the hash-tags "#barcelona", "#realmadrid", "#fcbvsrmfc", and ; these refer to the names of teams that played each other relating to that event.
In general , for a match , we consider the hash tags of the team names and also the general trending hash tags about the match.

The below figure shows the number of tweets for various football games by finding them using hashtags in the format mentioned above.



MICROBLOGGING AND TWITTER . There has been much recent interest on identifying and then tracking the evolution of events on Twitter and other social media websites,e.g., discussions about an earthquake on twitter (Sakaki,Okazaki, and Matsuo 2010), detecting new events (also called first stories) in the tweet-stream (Petrovi̇́ c, Osborne,and Lavrenko 2010), visualizing the evolution of tags (Dubinko et al. 2006), and other events on Flickr, Youtube, and Facebook (Becker, Naaman, and Gravano 2010; Chen and Roy 2009). The problem has also been approached from the point of view of efficiency: (Luo, Tang, and Yu 2007) propose indexing and compression techniques to speed up event detection without sacrificing detection accuracy. However, to the best of our knowledge, we are the first to study the summarization of events using user updates on Twitter.

SEQUENTIAL MODELING . There are a variety of methods to model sequence information. Change-point models try to detect instants of time when there is some marked change in the behavior of the sequence, e.g., the intensity with which observations arrive suddenly changes (Mannila and Salmenkivi 2001; Kleinberg 2002). Thus, change-points give a segmentation of the sequence into chunks of different intensities.Another way to segment the sequence is on the basis of differences in distribution of the observations themselves.A classical technique for this is the Hidden Markov Model (HMM) (Rabiner 1989; Wang et al. 2010), which posits the existence of an underlying process that transitions through a sequence of latent states, with observations being generated independently by each state in the sequence. HMMs have been extremely successful in areas as varied as speech recognition, gesture recognition, and bioinformatics. The various ways in which our proposed approach enhances the classical HMM are described in detail in the Algorithms section.

SUMMARIZATION : Summarization of text documents has been well studied in the IR community. A common method is based on computing relevance scores for each sentence in the document and then picking from among the best (Gong and Liu 2001). More complex methods are based on latent semantic analysis, hidden markov models, deep natural language analysis, among others (Gong and Liu 2001;Das and Martins 2007). However, these are primarily aimed at standalone web pages and documents. While the summarization of a sequence of tweets can be seen as a form of text summarization, there are nuances that are not considered by most prior algorithms. Still the main contribution of our work is in modeling the underlying hidden structure of events; most any off-the-shelf summarization method can then be then used to extract the important tweets to construct the summary.
(**Event Summarization using Tweets : Deepayan Chakrabarti and Kunal Punera**)

“Participant” is a general concept to denote the event participating persons, organizations, product lines, etc., each of which can be captured by a set of correlated proper nouns. For example, the NBA player “LeBron Raymone James” can be represented by {LeBron James, LeBron, LBJ, King James, L. James}, where each proper noun represents a unique mention of the participant. In this work, we automatically identify the proper nouns from tweet streams, filter out the infrequent ones using a threshold ψ, and cluster them into individual event participants. This process allows us to dynamically identify the key participating entities and provide a full-coverage for these participants in the event summary.
(**A Participant-based Approach for Event Summarization Using Twitter Streams : Chao Shen , Fei Liu , Fuliang Weng , Tao Li**)

Although there are many algorithms that can be used to summarize an event , all of them fail in detecting **sub-events** that might be of utmost interest for the user.
For example : When we consider a goal as an event in a football match , the user might be looking for the build up that has resulted in the goal , may be the team-work involved , the assist provider or may be even the creator.

V. PROCEDURE

We will discuss three algorithms to summarize tweets about events. In order to aid understanding of the issues involved and the design choices we make, we start by
presenting the simplest approach and progressively repair its shortcomings using more sophisticated methods. Also, while the algorithms are given for structured long-running events in general, we continue to use the running example of sporting events to elucidate ideas.

SUMMALL TEXT(Algo 1)

The straightforward approach to summarizing tweets is to simply consider each tweet as a document, and then apply a summarization method on this corpus. In particular, we associate with each tweet a vector of the **TF-logIDF** of its constituent words.We select those tweets which are closest to all other tweets from the event**.**

```
INPUT: Tweet corpus Z, tweet word vocabulary V , desired number of tweets n

OUTPUT: Set of key tweets T

for i ∈ Z, w ∈ V do
        z i (w) = tfidf(w, i, Z)
end for

for i ∈ Z do
         score(i) = j∈Z cosine(z i , z j )
end for

T = top n tweets with maximum score
```

ISSUES : While this algorithm has the advantage of simplicity, it has several defects. First, O(|Z|^2) computations are required to compute the scores (though this) could be reduced in practice if pruning techniques and approximations such as
LSH are used).

SUMMTIMEINT TEXT(Algo 2)

In order to pick tweets from the entire duration of the event,we need to combine summarization with a segmentation strategy. The simplest idea is to split up the duration into equal-sized time intervals (say, 2 minutes for football), and then select the key tweets from each interval. However, clearly, not all intervals will contain useful sub-events. We detect such intervals by their low tweet volume relative to the average, and do not select any key tweets from such intervals.

```
INPUT: Tweet corpus Z, tweet word vocabulary V , desired number of tweets n, minimum activity threshold ,
time segments TS

OUTPUT: Set of key tweets T

T S = {s ∈ T S | tweet volume in segment s > % of |Z|}

for each segment s ∈ T S do
    Z[s] = Z restricted to the same time as s
    T s = SUMM A LL T EXT (Z[s], V, n/|T S |)
end for

T = T s
```

ISSUES : This algorithm ensures that the selected tweets are spread across the entire duration of the event. Diversity within any given time segment now becomes less useful, because only a few tweets are picked from each segment. However, the segmentation based on equal-sized time windows is
far too simplistic.
Combining the data from multiple events allows us to (a) better learn SUMM HMM parameters, and (b) better detect state transitions in a new game, thus solving the cold-start problem.

SUMMHMM TEXT(Algo 3)

A two-step process:

a. Segment the event timeline.
b. Pick key tweets to form the summary and describe the event.

To segment an event, we turn to a model that has worked very well for many such problems: the Hidden Markov Model (HMM). However, the standard HMM is modified to SUMM HMM to be applicable to our problem.SUMM HMM takes multiple events of the same type as input, and learns the model parameters that best fit the data. The model parameters consist of multinomial word distributions θ (s) , θ (sg) , θ (bg) and the transition probabilities.Given Θ (model parameters) ,the optimal segmentation of the events can be quickly found by the standard Viterbi algorithm.

Three sets of symbol probabilities:

a. θ (s) - Specific to each state , same for all events.
b. θ (sg) - Specific to a particular state and particular game.
c. θ (bg) - Background distribution of symbols over all states and games(Irrelevant tweets) .

```
INPUT: Tweet corpus Z, tweet word vocabulary V , desired number of tweets n, minimum activity threshold

OUTPUT: Set of key tweets T

Learn Θ by iterating the update equations in (Chakrabarti and
Punera 2011) until convergence

Infer time segments T S by the Viterbi algorithm (Rabiner 1989)

T S = {s ∈ T S | tweet volume in segment s > % of |Z|}

for each segment s ∈ T S do
        Z[s] = Z restricted to time s
        T s = SUMMALL TEXT (Z[s], V, n/|T S |)
end for

T = T s
```

ISSUES : The assumption of the availability of multiple similar events may not be possible in practice, especially at the granularity of the moments that is considered within the events.For example, an HMM is unlikely to capture all possible sequences of sub-events (e.g., interception, touchdown in American football) in an event.Also, certain vocabulary items, such as the names of participants, cannot be learned unless the same two teams have played multiple times, which is not common in some sports or in non-sporting events.

SPIKE DETECTION (Algo 4)

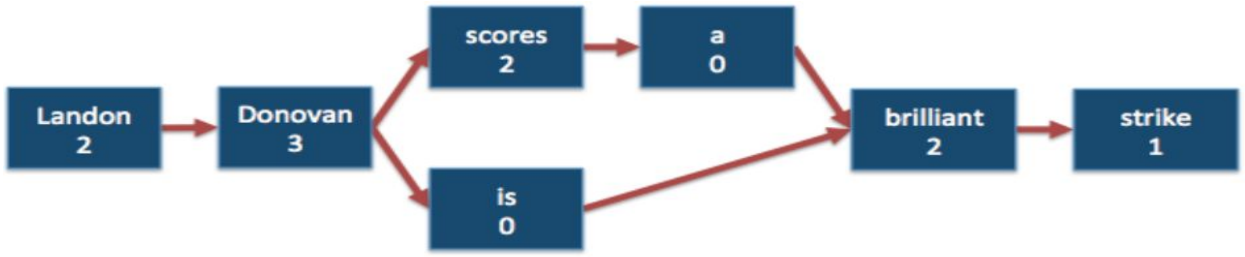
Sporting events consist of a sequence of moments, each of which may contain actions by players, the referee, the fans, etc. Our algorithm relies on Twitter users to collectively identify the important moments within an event and also describe those moments. At a high level, our algorithm relies on two properties of the Twitter stream :

- Sudden increases, or “spikes,” in the volume of tweets in the stream suggest that something important just happened because many people found the need to comment on it.
- A high percentage of the contents of the tweets at a “spike” in volume contain text describing what happened in the moment, and this text often contains repetitive elements, such as the names of players involved , types of event,etc.

Final breakdown of our algorithm:

- Status updates or tweets are collected using basic keyword filtering (using hash tags) .
- The important moments within an event are detected by searching for extreme changes in update volume on a per minute basis (detection of spikes).
- Extraction of relevant tweets from a set of tweets in a sub event is done by associating with each tweet a vector of the TF-log(IDF) . We select those tweets which are closest to all other tweets from the event.
- Several noise reduction algorithms are used to eliminate spam and off-topic status updates.
- Sentences from the status updates in each cluster are ranked using the phrase graph, and the top n sentences containing no overlapping tokens are returned. They are ranked based on the weight of the individual tokens.

Figure shows the phrase graph that would be generated from these status updates:
Landon Donovanscores!
Donovan scores a brilliant strike!
Landon Donovan is brilliant!
.



VI. WORK DONE

Mining Twitter Data to build datasets

We've collected and built datasets for 3 different football matches , namely , "Manchester United vs Manchester City" , "Everton vs Arsenal" , "Manchester United vs Liverpool" by mining data during the happening of ongoing match or event.

This is done with the help of Twitter API . The list of different ways to use Twitter could be really long, and with 500 millions of tweets per day, there's a lot of data to analyse and to play with. The Twitter API makes the job of collecting and playing with tweets easy.

Twitter provides REST APIs you can use to interact with their service. There is also a bunch of Python-based clients out there that we can use without re-inventing the wheel. In particular, Tweepy in one of the most interesting and straightforward to us .

The tweets that are generated for a particular football match can be separated by using the help of hash tags which was explained above in the "Related Work" section.

In order to authorise our app to access Twitter on our behalf, we need to use the OAuth interface (For this , first we need to register a dummy app using the credential details of our Twitter account) :

```
import tweepy
from tweepy import OAuthHandler

consumer_key = 'YOUR-CONSUMER-KEY'
consumer_secret = 'YOUR-CONSUMER-SECRET'
access_token = 'YOUR-ACCESS-TOKEN'
access_secret = 'YOUR-ACCESS-SECRET'

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth)
```

Depending on the search term, we can gather tons of tweets within a few minutes. This is especially true for live events with a world-wide coverage (football matches which is the domain we are working on), so keep an eye on the JSON file to understand how fast it grows and consider how many tweets you might need for your tests. The below script will save each tweet on a new line.

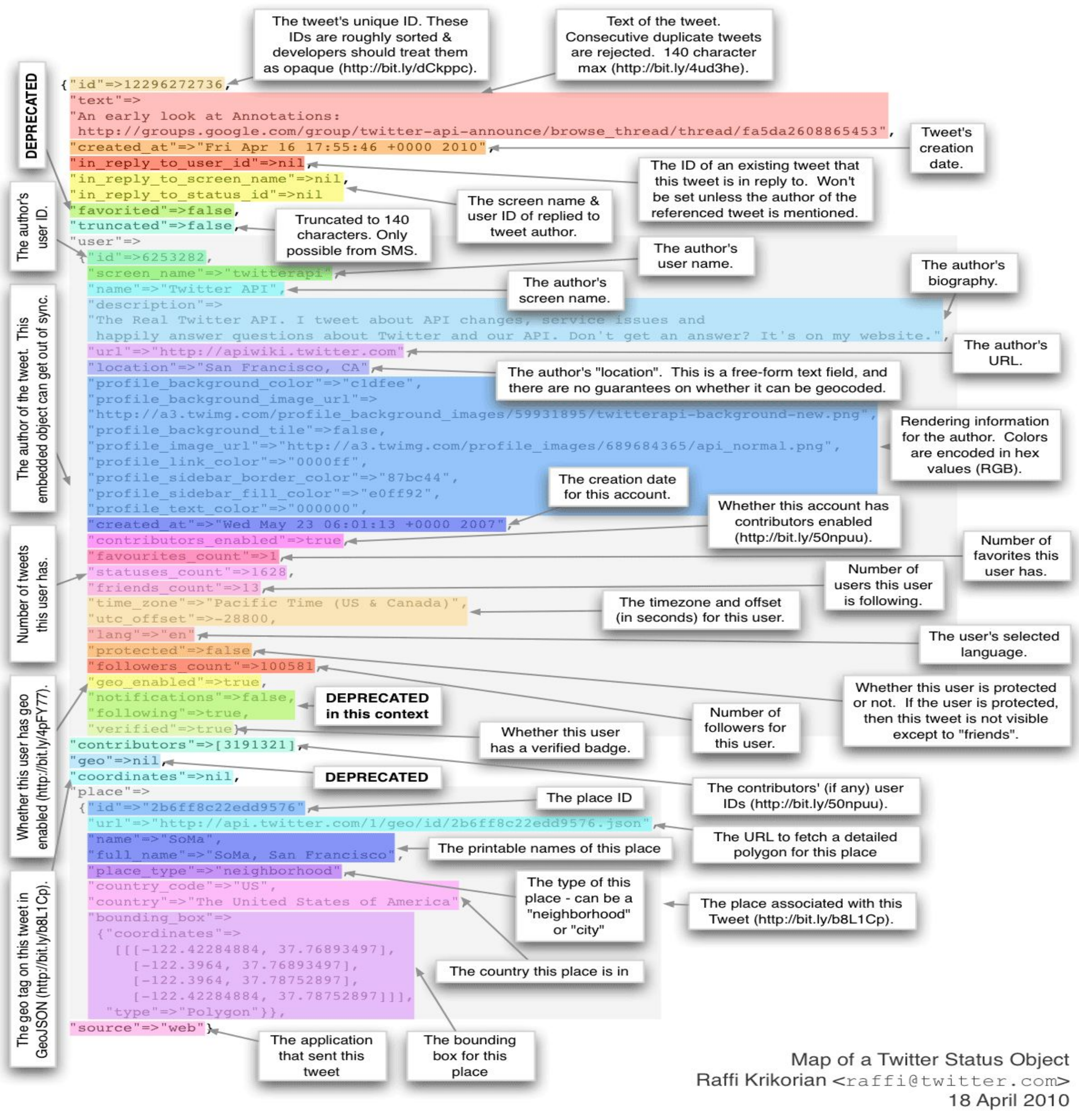
```
from tweepy import Stream
from tweepy.streaming import StreamListener

class MyListener(StreamListener):

            def on_data(self, data):
                try:
                    with open('python.json', 'a') as f:
               f.write(data)
               return True
                except BaseException as e:
                    print("Error on_data: %s" % str(e))
                 return True

            def on_error(self, status):
                print(status)
                return True

twitter_stream = Stream(auth, MyListener())
twitter_stream.filter(track=['#teamnames' or '#team1vsteam2'])
```

Noise Elimination of the Mined Data

The event-tweets stream obtained by the process described above is very noisy.Noise elimination is done on the streamed datasets to remove the irrelevant tweets. The chief sources of noise are given below:

1. Tweets not in the standard format are removed.
2. Spam-tweets are easily removed since they almost always have a URL in them.
3. Tweets of users that have less than 2 or greater than 100 tweets for any one football game are also eliminated.
4. Duplicate tweets like live stream , score card are also removed.
5. Finally, we place similar thresholds on number of occurrences of words: we remove (Porter-stemmed) words that occur fewer than 5 times or in more than 90% of the tweets for the football game.

Given below image represents the map of a tweet . Each line in the json dataset file collected represents a tweet which is in the format the below image represents.



```
{"id"=>12296272736,
 "text"=>
 "An early look at Annotations:
  http://groups.google.com/group/twitter-api-announce/browse_thread/thread/fa5da2608865453",
 "created_at"=>"Fri Apr 16 17:55:46 +0000 2010",
 "in_reply_to_user_id"=>nil,
 "in_reply_to_screen_name"=>nil,
 "in_reply_to_status_id"=>nil
 "favorited"=>false,
 "truncated"=>false,
 "user"=>
  {"id"=>6253282,
   "screen_name"=>"twitterapi",
   "name"=>"Twitter API",
   "description"=>
   "The Real Twitter API. I tweet about API changes, service issues and
    happily answer questions about Twitter and our API. Don't get an answer? It's on my website.",
   "url"=>"http://apiwiki.twitter.com",
   "location"=>"San Francisco, CA",
   "profile_background_color"=>"c1dfee",
   "profile_background_image_url"=>
   "http://a3.twimg.com/profile_background_images/59931895/twitterapi-background-new.png",
   "profile_background_tile"=>false,
   "profile_image_url"=>"http://a3.twimg.com/profile_images/689684365/api_normal.png",
   "profile_link_color"=>"0000ff",
   "profile_sidebar_border_color"=>"87bc44",
   "profile_sidebar_fill_color"=>"e0ff92",
   "profile_text_color"=>"000000",
   "created_at"=>"Wed May 23 06:01:13 +0000 2007",
   "contributors_enabled"=>true
   "favourites_count"=>1
   "statuses_count"=>1628,
   "friends_count"=>13
   "time_zone"=>"Pacific Time (US & Canada)",
   "utc_offset"=>-28800,
   "lang"=>"en"
   "protected"=>false
   "followers_count"=>100581
   "geo_enabled"=>true,
   "notifications"=>false,
   "following"=>true,
   "verified"=>true}
 "contributors"=>[3191321]
 "geo"=>nil,
 "coordinates"=>nil,
 "place"=>
  {"id"=>"2b6ff8c22edd9576"
   "url"=>"http://api.twitter.com/1/geo/id/2b6ff8c22edd9576.json"
   "name"=>"SoMa",
   "full_name"=>"SoMa, San Francisco",
   "place_type"=>"neighborhood"
   "country_code"=>"US",
   "country"=>"The United States of America"
   "bounding_box"=>
    {"coordinates"=>
      [[[-122.42284884, 37.76893497],
        [-122.3964, 37.76893497],
        [-122.3964, 37.78752897],
        [-122.42284884, 37.78752897]]],
     "type"=>"Polygon"}},
 "source"=>"web"}
```

The tweet's unique ID. These IDs are roughly sorted & developers should treat them as opaque (http://bit.ly/dCkppc).
Text of the tweet. Consecutive duplicate tweets are rejected. 140 character max (http://bit.ly/4ud3he).
DEPRECATED
Tweet's creation date.
The ID of an existing tweet that this tweet is in reply to. Won't be set unless the author of the referenced tweet is mentioned.
The screen name & user ID of replied to tweet author.
The author's user ID.
Truncated to 140 characters. Only possible from SMS.
The author's user name.
The author's biography.
The author's screen name.
The author's URL.
The author of the tweet. This embedded object can get out of sync.
The author's "location". This is a free-form text field, and there are no guarantees on whether it can be geocoded.
Rendering information for the author. Colors are encoded in hex values (RGB).
The creation date for this account.
Whether this account has contributors enabled (http://bit.ly/50npuu).
Number of favorites this user has.
Number of tweets this user has.
Number of users this user is following.
The timezone and offset (in seconds) for this user.
The user's selected language.
Whether this user has geo enabled (http://bit.ly/4pFY77).
DEPRECATED in this context
Whether this user is protected or not. If the user is protected, then this tweet is not visible except to "friends".
Number of followers for this user.
Whether this user has a verified badge.
DEPRECATED
The contributors' (if any) user IDs (http://bit.ly/50npuu).
The place ID
The URL to fetch a detailed polygon for this place
The printable names of this place
The type of this place - can be a "neighborhood" or "city"
The place associated with this Tweet (http://bit.ly/b8L1Cp).
The geo tag on this tweet in GeoJSON (http://bit.ly/b8L1Cp).
The country this place is in
The application that sent this tweet
The bounding box for this place

Map of a Twitter Status Object
Raffi Krikorian <raffi@twitter.com>
18 April 2010

Basic Format of reading and parsing a Json file :

```
import json

with open('data.json') as data_file:
    data = json.load(data_file)

print(data)

data["user"][0]["id"]
data["user"]["id"]
data["text"]
```

It is converted into a form of dictionary using which each category data can be extracted and noise elimination can be done in the ways mentioned above.

Plotting the absolute number of Tweets over the time interval

It is clear that the ebb and flow of the tweets is correlated to the happenings of the game. In addition to the frequency we can tag some of the frequency spikes with the common words in the tweets. As we can see the words in the tweets associated with the spikes do document the significant happenings at these events: the “sub-events”. Hence, our goal in this work is to construct a event summary by selecting tweets, from the set of relevant tweets, that describe these sub-events. Here we list some issues of this data that our approach will have to deal with.

Sub-events are marked by increased frequency of tweets. More precisely, it is the change in tweet-frequency as opposed to absolute levels that demarcate sub-events. Second, boundaries of sub-events also result in a change in vocabulary of tweets. The vocabulary specific to a sub-event tends to come from two sources: one part of it is sport specific comprising words like goal,
assist, etc., while another part is game-specific, such as player names. Finally, we consider recurring sport events with every repeated event sharing at least the sport-specific vocabulary, and hence our approach should learn from past games to do a better job of summarizing future ones.

The tool used for data analysis and this plotting with Python is **“Pandas”**, which also has a fairly decent support for time series.

```
import pandas
import json

dates_EFCvAFC = [ ]

# f is the file pointer to the JSON data set
f=open("efcvafc.json")

for line in f:
        tweet = json.loads(line)
        dates_EFCvAFC.append(tweet['created_at'])

# a list of "1" to count the hashtags
ones = [1]*len(dates_EFCvAFC)

# the index of the series
idx = pandas.DatetimeIndex(dates_EFCvAFC)
# the actual series (at series of 1s for the moment)
EFCvAFc = pandas.Series(ones, index=idx)

# Resampling / bucketing
per_minute = EFCvAFC.resample('1Min', how='sum').fillna(0)


########### Plotting ##############
```

```
time_chart = vincent.Line(per_minute)
time_chart.legend(title='Everton vs Arsenal(no noise)')
time_chart.axis_titles(x='Time Elapsed since start', y='Tweet Volume')
time_chart.to_json('time_chart.json')
```

The interesting moments of the match are observable from the spikes in the series.

Detection of sub-events through sudden increases , or “spikes” in the volume of tweets in the stream

We could have used a moment detection algorithm that relies on the absolute value of the volume,however we discovered two problems with this approach. First, sometimes the stream volume may stay high for several minutes and have multiple localized peaks. Second, some moments generate significantly less traffic than others and might be missed by a technique that relies on absolute values. To avoid the above problems, we chose to use a detection algorithm based on the change in volume (i.e. the slope of the volume graph).

We computed a threshold for the entire event from basic statistics of the set of all slopes for that event. For example, a threshold for a particular soccer match may be computed from the tweets recorded for that entire match.

Unfortunately, we found that for events with very large spikes the standard deviation in the slope is too large to find the smaller but still meaningful spikes. After some experimentation, we settled on a threshold of 3 * median, which produced results that closely matched a visual inspection of spikes.

After identifying all slopes that exceed the threshold, we generate a list of “spikes” that correspond to the important sub-events in the event. Each spike can be defined as the tuple of <Start Time, Peak Time, End Time>. For each slope above threshold, we calculate the start time by searching backwards in time until we find the point where the slope began going up. The peak time is calculated by searching forward until we find the point where the slopes start decreasing. Finally, we calculate the end time by searching forward from the peak time to find the point where the slope begins increasing again . Before returning the list of spikes, we remove any duplicates, which may happen for large spikes that include multiple above threshold slopes.

Finding the relevant tweets (pointing towards a sub event) by applying tf-idf in every sub event detected:

Extraction of relevant tweets from a set of tweets is done by associating with each tweet a vector of the TF-log(IDF) . We select those tweets which are closest to all other tweets from the event.

tf–idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.It is often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

One of the simplest ranking functions is computed by summing the tf–idf for each query term; many more sophisticated ranking functions are variants of this simple model.

Cosine similarity between two tweets is used to find the similarity between them by considering them as vectors . Weight score for each tweets is calculated by summing the similarities of that tweet with all the other tweets . Top n high score tweets are returned.

This is applied on every sub-event that is detected . From every sub-event,we find relevant tweets from a number of tweets which point to that event by using the concepts of tf-idf and cosine similarity between the tweets converted into vectors.

Outline Code to find similarity between two sentences:

```
def termFrequency(term, document):
   normalizeDocument = document.lower().split()
   return normalizeDocument.count(term.lower()) / float(len(normalizeDocument))

def inverseDocumentFrequency(term, allDocuments):
   numDocumentsWithThisTerm = 0
   for doc in allDocuments:
      if term.lower() in allDocuments[doc].lower().split():
         numDocumentsWithThisTerm = numDocumentsWithThisTerm + 1
    if numDocumentsWithThisTerm > 0:
      return 1.0 + log(float(len(allDocuments)) / numDocumentsWithThisTerm)
   else:
         return 1.0
```

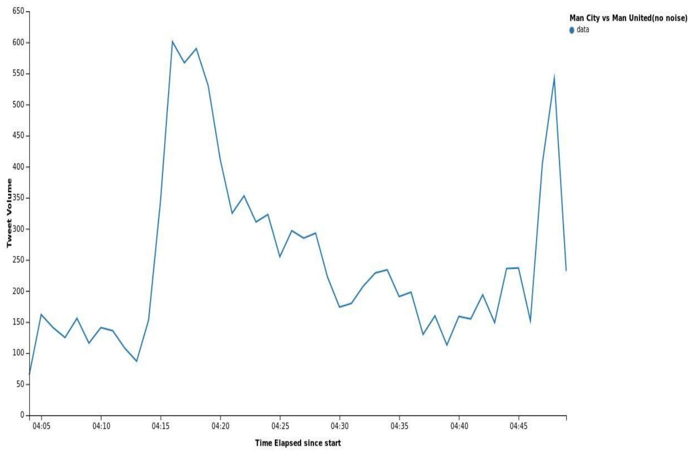
Ranking the tweets based on the phrase graph model and the weights of the tokens:

The final step is to find the N sentences contained in our set of status updates that best summarize the set. Our approach is to construct a phrase graph from the longest sentence in each status update, weight the graph according to frequency of words and a few other heuristics, and then to score the longest sentence using the phrase graph. We construct the phrase graph from only the longest sentence in each status update, rather than using the entire update. This helps to ensure that our generated summaries are well formed and eliminates some of the noise found in updates. We leverage the phrase graph to score the input sentences, each extracted from one tweet , and find the best scoring sentences. To score a sentence, we tokenize the sentence, remove any duplicate tokens, and sum the weights of all nodes matching a token to generate a score. Duplicate tokens are removed so that tokens that appear multiple times only contribute to the score once. Given the list of scored sentences, we then output the top N sentences.

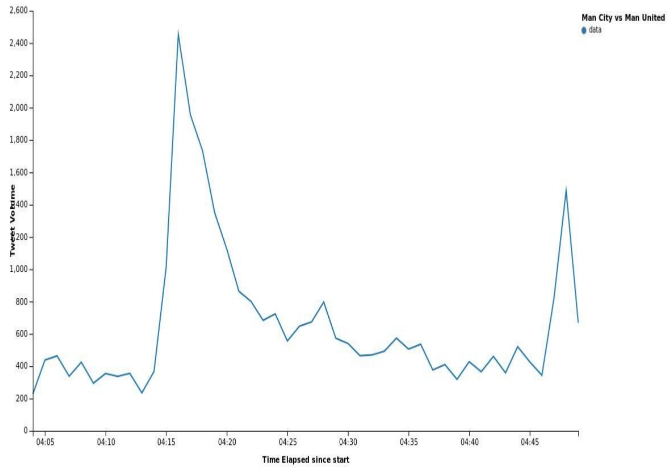
VII. RESULTS AND EVALUATION

Graphs plotted between the number of tweets over the time interval :

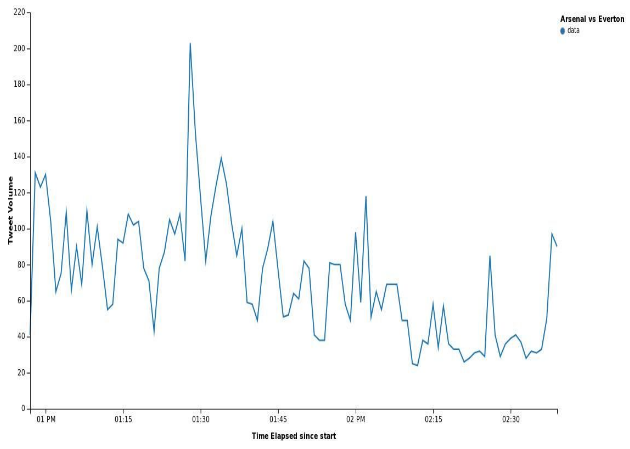
Man-city vs ManUtd (with noise)



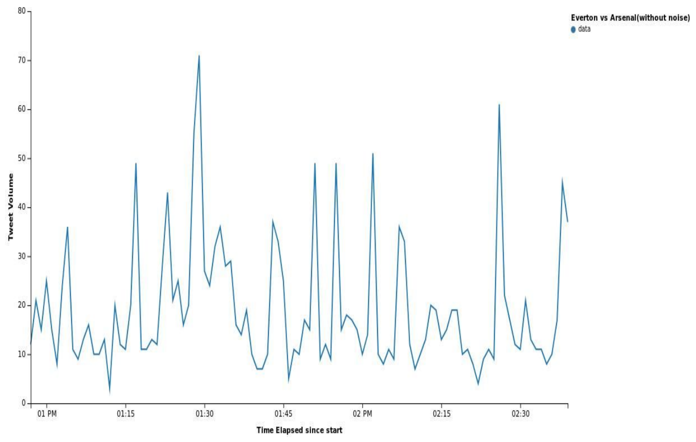
Man-city vs ManUtd (without noise)



Arsenal vs Everton (with noise)



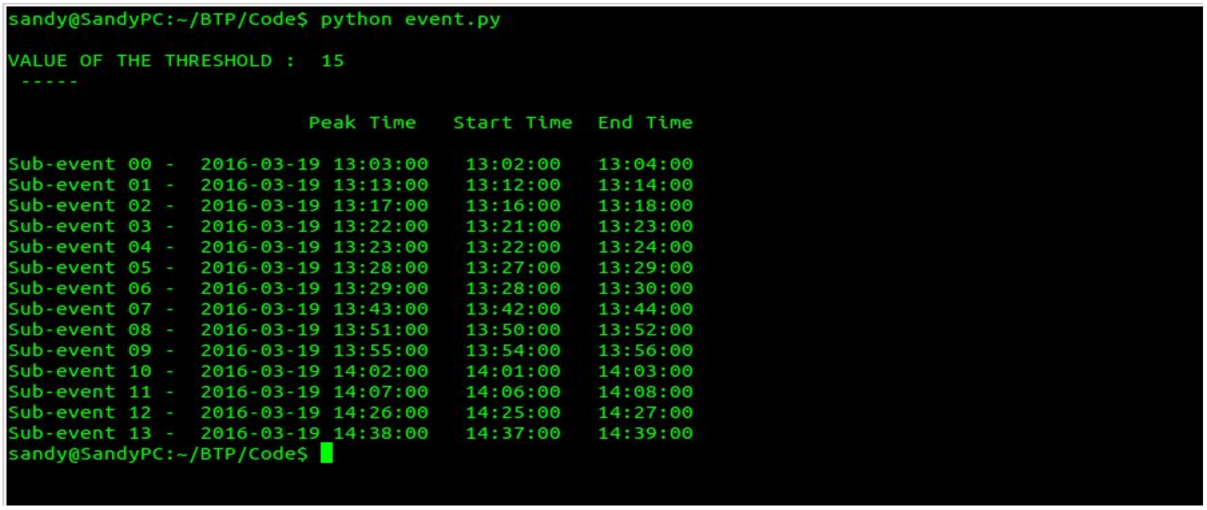
Arsenal vs Everton (without noise)



Observations :

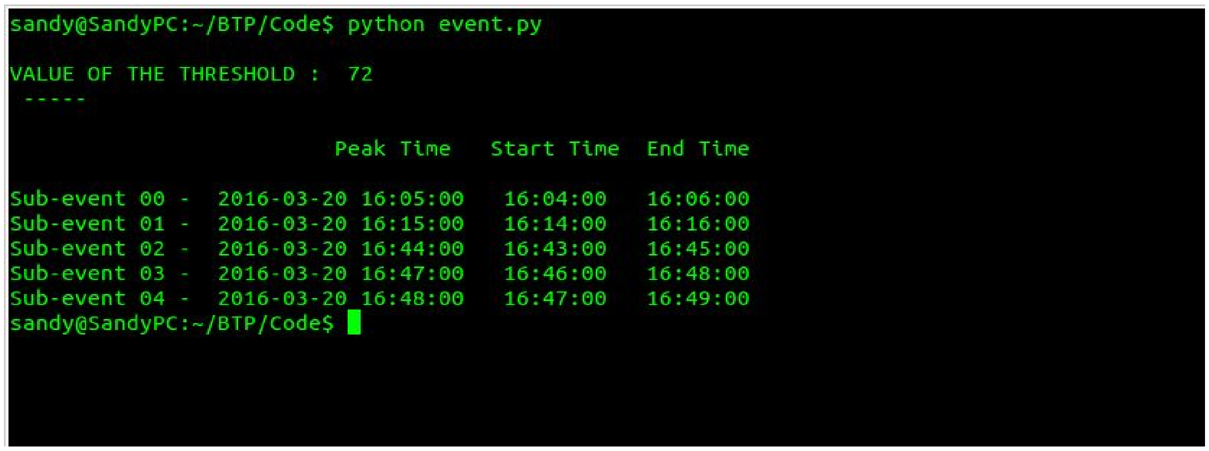
When the graphs are plotted for the tweet data stream containing noise , we found the number of peaks in the graph are very high as few of non interesting events have also been detected due to the gradual increase in the tweet volume as a result of spam tweets . So , such events are eliminated by removing the tweets containing noise . This can be easily observed in the graphs on the right sides where the peaks depicting various interesting occurrences have reduced.

Time Intervals of sub-events detected in Arsenal vs Everton match



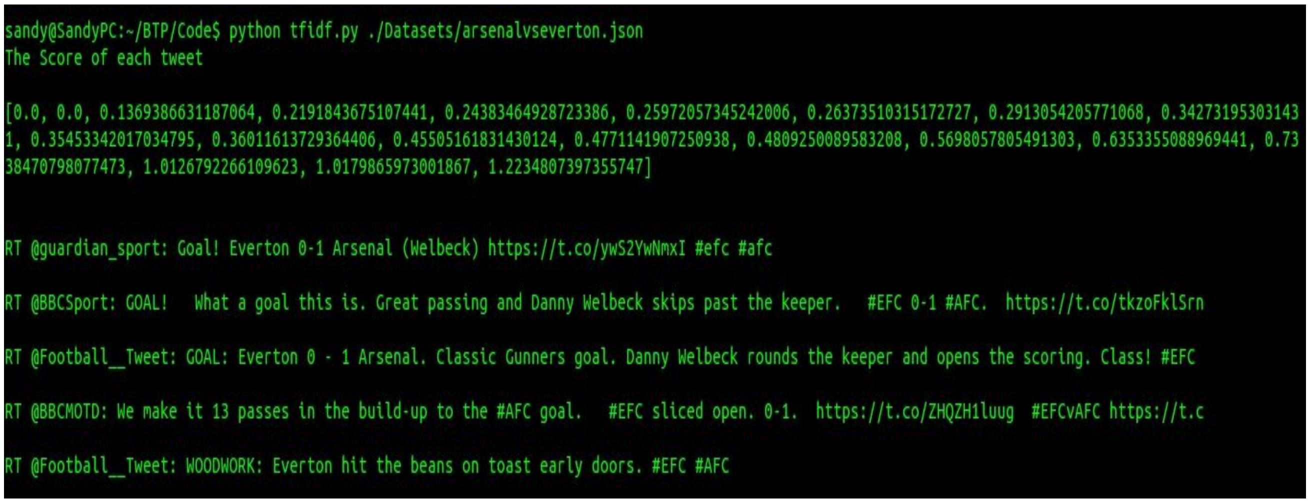
```
sandy@SandyPC:~/BTP/Code$ python event.py

VALUE OF THE THRESHOLD :  15
 -----

                          Peak Time    Start Time   End Time

Sub-event 00 -  2016-03-19 13:03:00   13:02:00   13:04:00
Sub-event 01 -  2016-03-19 13:13:00   13:12:00   13:14:00
Sub-event 02 -  2016-03-19 13:17:00   13:16:00   13:18:00
Sub-event 03 -  2016-03-19 13:22:00   13:21:00   13:23:00
Sub-event 04 -  2016-03-19 13:23:00   13:22:00   13:24:00
Sub-event 05 -  2016-03-19 13:28:00   13:27:00   13:29:00
Sub-event 06 -  2016-03-19 13:29:00   13:28:00   13:30:00
Sub-event 07 -  2016-03-19 13:43:00   13:42:00   13:44:00
Sub-event 08 -  2016-03-19 13:51:00   13:50:00   13:52:00
Sub-event 09 -  2016-03-19 13:55:00   13:54:00   13:56:00
Sub-event 10 -  2016-03-19 14:02:00   14:01:00   14:03:00
Sub-event 11 -  2016-03-19 14:07:00   14:06:00   14:08:00
Sub-event 12 -  2016-03-19 14:26:00   14:25:00   14:27:00
Sub-event 13 -  2016-03-19 14:38:00   14:37:00   14:39:00
sandy@SandyPC:~/BTP/Code$
```

Time Intervals of sub-events detected in ManU vs ManCity match



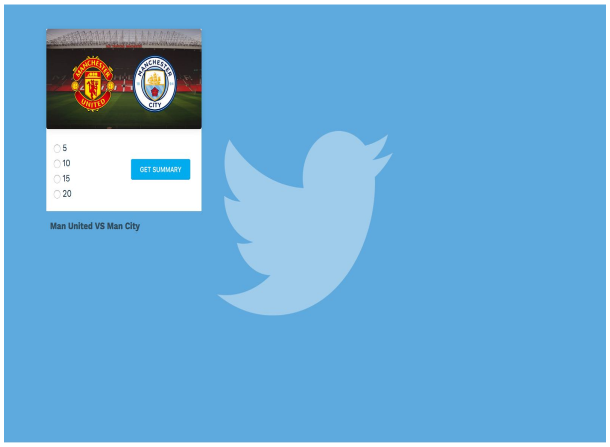
```
sandy@SandyPC:~/BTP/Code$ python event.py

VALUE OF THE THRESHOLD :  72
 -----

                          Peak Time    Start Time   End Time

Sub-event 00 -  2016-03-20 16:05:00   16:04:00   16:06:00
Sub-event 01 -  2016-03-20 16:15:00   16:14:00   16:16:00
Sub-event 02 -  2016-03-20 16:44:00   16:43:00   16:45:00
Sub-event 03 -  2016-03-20 16:47:00   16:46:00   16:48:00
Sub-event 04 -  2016-03-20 16:48:00   16:47:00   16:49:00
sandy@SandyPC:~/BTP/Code$
```

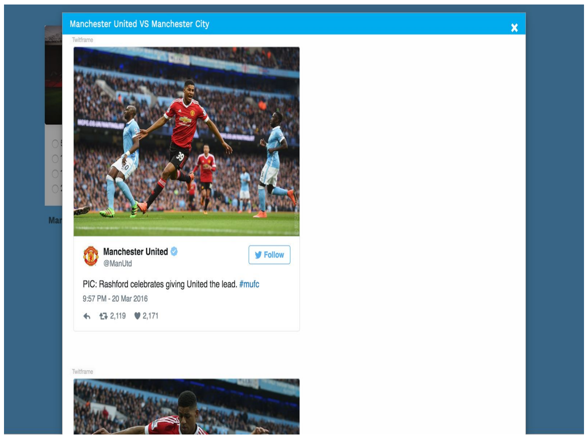
Arsenal vs Everton (Best Tweets from a set of tweets on Tf-Idf)



```
sandy@SandyPC:~/BTP/Code$ python tfidf.py ./Datasets/arsenalvseverton.json
The Score of each tweet

[0.0, 0.0, 0.1369386631187064, 0.2191843675107441, 0.24383464928723386, 0.25972057345242006, 0.26373510315172727, 0.2913054205771068, 0.34273195303143
1, 0.35453342017034795, 0.36011613729364406, 0.45505161831430124, 0.4771141907250938, 0.4809250089583208, 0.5698057805491303, 0.6353355088969441, 0.73
38470798077473, 1.0126792266109623, 1.0179865973001867, 1.2234807397355747]


RT @guardian_sport: Goal! Everton 0-1 Arsenal (Welbeck) https://t.co/ywS2YwNmxI #efc #afc

RT @BBCSport: GOAL!  What a goal this is. Great passing and Danny Welbeck skips past the keeper.  #EFC 0-1 #AFC.  https://t.co/tkzoFklSrn

RT @Football__Tweet: GOAL: Everton 0 - 1 Arsenal. Classic Gunners goal. Danny Welbeck rounds the keeper and opens the scoring. Class! #EFC

RT @BBCMOTD: We make it 13 passes in the build-up to the #AFC goal.  #EFC sliced open. 0-1.  https://t.co/ZHQZH1luug  #EFCvAFC https://t.c

RT @Football__Tweet: WOODWORK: Everton hit the beans on toast early doors. #EFC #AFC
```

UI and the display of various matches and their summaries



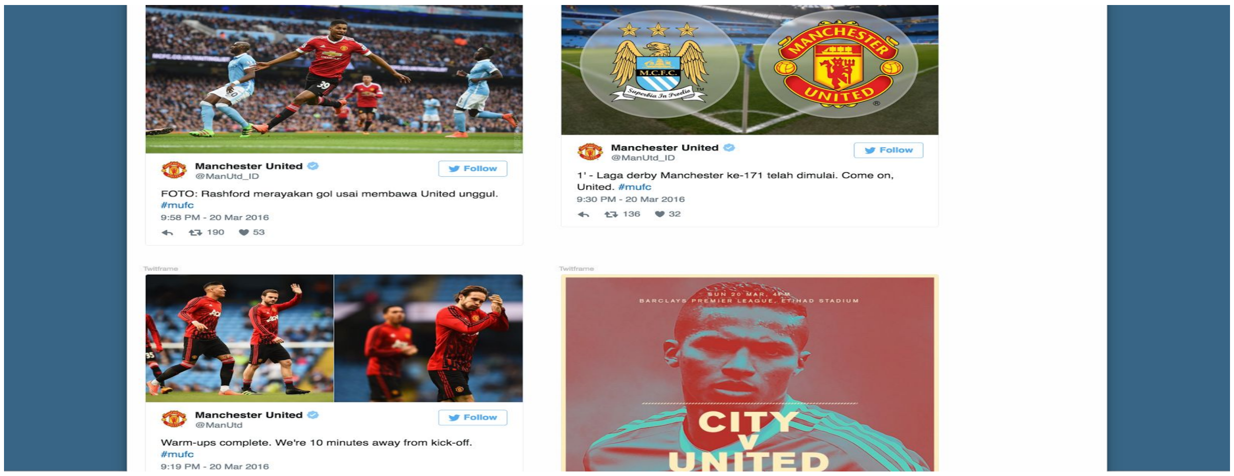


Manchester United @ManUtd

PIC: Rashford celebrates giving United the lead. #mufc

9:57 PM - 20 Mar 2016

2,119 2,171



Manchester United @ManUtd_ID

FOTO: Rashford merayakan gol usai membawa United unggul. #mufc

9:58 PM - 20 Mar 2016

190 53

Manchester United @ManUtd_ID

1' - Laga derby Manchester ke-171 telah dimulai. Come on, United. #mufc

9:30 PM - 20 Mar 2016

136 32

Manchester United @ManUtd

Warm-ups complete. We're 10 minutes away from kick-off. #mufc

9:19 PM - 20 Mar 2016

VIII. FUTURE WORK

We plan to examine this heuristic(of selecting the right threshold) in detecting sub events to see how it applies to other types of sporting events not in our data set. Finally , generate on single system by linking the work done.

We may also try an online version of our algorithm of sub event detection by computing the threshold from the tweets in a moving window with start time and end time, where the size of the moving window is a parameter.

IX. CONCLUSIONS

We have shown that it is possible to use the publicly available tweets about a sports event to aid in event detection and summary generation for associated media. Using a filtered stream of tweets we can identify and tag the most significant events within 2 different field sports with a high degree of accuracy and success. We believe that the power of our approach lies within its seeming simplicity, allowing for equivalent results to far more computationally complex,
and time consuming approaches.

Plotting of the absolute number of tweets over the duration of the match is done. This would help in detecting the intervals of various sub-events over the course of the match. Segmenting a timeline of an event with generating the occurrences of all the sub events is done.Extraction of relevant tweets from a set of tweets is done. Finding the N sentences(on the basis of scores generated using phrase graph) contained in our set of tweets that will best summarize the set to be done as a future work.

X. REFERENCES

- Reference Paper 1 : Event summarization using Tweets
- Reference Paper 2 : Summarizing Sporting Events Using Twitter
- http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/
- http://www.slaw.ca/wp-content/uploads/2011/11/map-of-a-tweet-copy.pdf
- https://marcobonzanini.com/2015/03/02/mining-twitter-data-with-python-part-1/
- https://apps.twitter.com/
- https://marcobonzanini.com/2015/04/01/mining-twitter-data-with-python-part-5-data-visualisation-basics/

XI. Learning of various Twitter problems

Discovering Similar Users and Grouping them on Twitter

This work studies the problem of discovering “similar” users at Twitter, where we define two users to be similar if they produce content similar to each other. The discovery of top similar accounts for each Twitter user has a variety of applications at Twitter including user recommendations and advertiser targeting.
The discovery of top similar accounts for each Twitter user has a variety of applications at Twitter. First, similar accounts can help in providing link recommendations which can drive the discovery of connections on Twitter. For example, if a user follows Kobe Bryant at Twitter, the user might also be interested in LeBron James. In fact, this benefits the recommended user as well, since users with more follows are more likely to be engaged with Twitter. Second, similar user accounts are also useful for accurate advertisement targeting since Nike might be interested in showing its advertisements to followers of accounts similar to itself, such as those of Adidas. Third, the discovered similar users could help enhance user experience by delivering more relevant content to users by helping target content recommendations better. Finally, since the social graph itself can be enriched by similar-to semantics, it can enhance the precision of various analytic tasks such as community detection.

Example 1 @KingJames (LeBron James) is a similar-to user account to @kobebryant (Kobe Bryant), because both LeBron James and Kobe Bryant are top basketball players at NBA.
Example 2 @katyperry (Katy Perry) is a similar-to user account to @ladygaga (Lady Gaga), because both Katy Perry and Lady Gaga are famous singers and songwriters.

Click-through prediction for advertising in Twitter timeline

We present the problem of click-through prediction for advertising in Twitter timeline, which displays a stream of Tweets from accounts a user choose to follow. Traditional computational advertising usually appears in two forms: sponsored search that places ads onto the search result page when a query is issued to a search engine, and contextual advertising that places ads onto a regular, usually static Web page. Compared with these two paradigms, placing ads into a Tweet stream is particularly challenging given the nature of the data stream: the context into which an ad can be placed updates dynamically and never replicates. Every ad is therefore placed into a unique context. This makes the information available for training a machine learning model extremely sparse.

We give a summary of various methods for future references.
1. Pointwise learning, which considers each ad as an individual training instance. This is the production model currently deployed at Twitter.
2. Pairwise learning. This method essentially refers to pairwise learning with calibration. Since the performance of pairwise approach before calibration is extremely poor in our empirical studies, we only report results on the calibrated version, which preserves the original ranking performance.
3. Combined learning, which takes into account classification and ranking under one framework.
4. Pseudo. Based on combined learning, we artificially create more pairwise training instances called pseudo-pairs.

Localized twitter opinion mining using sentiment analysis

Analysis of public information from social media could yield interesting results and insights into the world of public opinions about almost any product, service or personality. Social network data is one of the most effective and accurate indicators of public sentiment. In this study, a methodology which allows utilization and interpretation of twitter data to determine public opinions is discussed . Analysis was done on tweets about a product , say iPhone 6. Feature specific popularities and male–female specific analysis has been included. Mixed opinions were found but general consistency with outside reviews and comments was observed.
Sentiment analysis technique is an effective means of discovering public opinions. Various companies often use online or paper based surveys to collect customer comments. Due to the emergence of social networking sites and applications, people tend to comment on their facebook or tweet profile. Therefore the paper based approach is not an efficient approach. Only a very small customer base can be reached and there is no guarantee that their answers in the survey are honest or not. Here social media comes into play. Facebook, Twitter and all other social media sites are full of people's opinions about products/services they use, comments about popular personalities and much more. Hence mining opinions about various subject matters from social media is a much more innovative approach for market analysis. A lot of research has been done on opinion mining from social media, most of which focuses on people's sentiment towards various topics. But analyzing social media data in this manner gives a much generalized idea. To make it more specific, sentiment analysis can be performed on social media data from explicit locations. Our approach is to find the sentiments in specific locations. This will allow companies to focus their marketing expenditures on areas where sentiment is low, while maintaining minimum advertisement in areas of high popularity.
Initially the tweets were filtered using Natural Language Processing tools. Only tweets which contained selected grammatical relations involving previously chosen keywords were selected. Each of the tweets in this filtered set were then given parts of speech tags. Each individual word in every tweet was assigned its own part of speech tag. After POS tagging the tweets were then processed using SentiWord which gave each tweet a sentiment score. Finally the data was presented graphically and several comparisons to real world scenarios were made to justify the accuracy of the methodology.

XII. Important Links

Presentation :
https://docs.google.com/presentation/d/1we-7kp6slHox6EVjyMOKUY-35pwWpRzDO1xFc_OMvRc/edit?usp=sharing

Report :
https://docs.google.com/document/d/1ef2MlhUY1uvlwt2X5Pi0Fbcb6adAvEJUwpaBzXYkrUc/edit?usp=sharing

Github (Code) : https://github.com/SandeepKasa/Summarization-Tweets-of-a-Football-Event