Automatic creation of event timelines

P. Akshay Kumar
MT19094

Ritesh Singh
MT19044

Gaurav Lodhi
MT19063

1 Problem Definition

Automatic generation of event timelines for a specific event hashtag whose corresponding tweets are extracted from Twitter for a specific time range. The generated timeline should be informative and interesting. Two models were developed – one capturing the result related information and the other capturing the interesting stats/events that took place on matchday.

2 Background

Over the past few years, there has been a sonic boom with petabytes of data being generated every day on social media. Twitter is one of the widely used social media platforms across the globe. It has evolved with its fast and timely reporting of various events within seconds of its occurrence. Due to the limited word limit of a tweet, it helps to capture the gist of the event that has occurred, very quickly. There was a need for a mechanism or model to put the various pieces of information related to a major event together on a timeline in chronological order.

3 Dataset used

3.1 Train dataset for model 1 and model 2

The data is extracted from Twitter using "GetOldTweets3" and "Tweepy" python modules which internally uses Twitter API corresponding to "#ipl". The criteria used to fetch tweets were as follows:

- Authoritative sources such as @IPL, @ESPNcricinfo, @cricbuzz and so on.
- Top 20 tweets of each day from each of the above sources.
- Date range is specified by the user.

3.2 Test Dataset

Model 1: The timeline generated by model 1 is expected to show the results of IPL teams qualification/elimination related information on the timeline. This was validated using the result data scrapped from `https://www.hindustantimes.com/ipl/results/`.

Model 2: The timeline generated by model 2 is expected to capture the interesting stats/events that took place on the match day. This was validated using the information with respect on every match posted on Wikipedia along with the stats. The data was scrapped from `https://en.wikipedia.org/wiki/2019_Indian_Premier_League#Matches`.

4 Proposed Solution

4.1 Model 1 (Done before mid evaluation)

Preprocessing

- Removal of URL links from tweets.
- Expanding contractions in tweet text.
- Tokenize using word tokenizer.
- Replace twitter handles in tweets with their original Twitter screen names.
- Removal of numbers after tokenization.
- Replace team abbreviation with complete team names.
- Removal of punctuations.
- Stopwords removal.

Creation of Contextual Vector: Contextual vector comprises of a set of phrases which is the ranked list of N-grams related to a hashtag. This captures the key phrases from the most important topics of the day. TF (term frequency), which is the number of occurrences of a token in a tweet and DF (document frequency) which is the occurrence of tokens across tweets for a given day is calculated. The product of TF and inverse DF is assumed to be the contextual score for a given token. The tokens are sorted based on contextual score and the ones which are less than the threshold are removed from the contextual vector.

Creation of Inverted Index: An inverted index is a dictionary of terms where dictionary key is the term and value comprises of occurrence of the term in various documents. The tweets are analogous to documents and tweetID to document ID. Here, the zeroth index contains the document frequency and the rest of the list starting from index 1 contains the term frequency hashed based on document ID. Once all the terms are parsed, each tf is multiplied with IDF computed from the zeroth place of inverted index.

idf = math.log10(number_of_tweets/df)

*tf-idf_score = (math.log10(1+tf))*idf*

Cosine Score calculations: The zeroth index of inverted index is replaced with contextual score for each term which was calculated in 4.2. A document vector is constructed using the 1..N (N is number of tweets per day) columns of inverted index and contextual vector using column 0. This gives a 1Xvocabulary size vectors corresponding to each tweet and a contextual vector. Cosine similarity is computed between these two vectors and the corresponding cosine score is saved in a dictionary.

cosine_score=sim(contextual_vector, document_vector)

Calculation of user engagement score The user engagement score gives weightage to number of likes and retweets for a given tweet. For a given source, the normalized retweet and likes count is computed and then its weighted sum is taken as user engagement score.

user_engagement_score = lambda $\times$ *(retweet_score / max_retweet_score) + (1-lambda)(likes_score / max_likes_score)*

where lambda is a tunable parameter. Here, lambda is taken as 0.7.

Calculations of final score: The final score is computed as follows:

*final_score = cosine_score * log(user_engagement_score + 2)*

Construction of timeline: The tweet with the highest final score is added to the timeline.

4.2 Model 2 (Done after mid evaluation)

Preprocessing

- Removal of URL links from tweets.
- Expanding contractions in tweet text.
- Tokenize using word tokenizer.
- Replace twitter handles in tweets with their original Twitter screen names.
- Removal of numbers after tokenization.
- Replace team abbreviation with complete team names.
- Removal of punctuations.
- Stopwords removal.

Creation of Contextual Vector: Contextual vector comprises of a set of phrases which is the ranked list of N-grams related to a hashtag. This captures the key phrases from the most important topics of the day. TF (term frequency), which is the number of occurrences of a token in a tweet and DF (document frequency) which is the occurrence of tokens across tweets for a given day is calculated. The product of TF and inverse DF is assumed to be the contextual score for a given token. The tokens are sorted based on contextual score and the ones which are less than the threshold are removed from the contextual vector.

Calculation of user engagement score: The user engagement score gives weightage to number of likes and retweets for a given tweet. For a given source, the normalized retweet and likes count is computed and then its weighted sum is taken as user engagement score.

user_engagement_score = lambda $\times$ *(retweet_score / max_retweet_score) + (1-lambda)(likes_score / max_likes_score)*

where lambda is a tunable parameter. Here, lambda is taken as 0.7.

Construction of term document matrix: Here, term refers to a tweet and document refers to the list of terms in tweets corpus vocabulary in the given time range. The term document dataframe comprises of vocabulary terms as columns corresponding to tweet in each row. The value of each cell in the dataframe corresponds to tf-idf score of a token with respect to a tweet. The TF and IDF scores are computed using the following formulae.

$$idf = math.log10(number\_of\_tweets/df)$$

$$tf\text{-}idf\_score = (math.log10(1+tf))*idf$$

Clustering of tweets: The representative point (tweet with highest user engagement score in each cluster) of each of the cluster formed is added to a tweet list. For each tweet in the tweet list, if the intersection is greater than x i.e. x=2 then the cosine score is computed between the contextual vector and the vector corresponding to the tweet. A binary vector is created for every tweet with the size of 1 x contextual vector. If the term is present then 1 is added to vector, else 0. The tweet with the highest cosine score with respect to the contextual vector is added to the timeline and it is removed from the list of tweets. This processed is continued till all the contextual vectors are exhausted.

Construction of timeline The representative point (tweet with highest user engagement score in each cluster) of each of the cluster formed is added to a tweet list. For each tweet in the tweet list, if the intersection is greater than x i.e. x=2 then the cosine score is computed between the contextual vector and the vector corresponding to the tweet. A binary vector is created for every tweet with the size of 1 x contextual vector. If the term is present then 1 is added to vector, else 0. The tweet with the highest cosine score with respect to the contextual vector is added to the timeline and it is removed from the list of tweets. This processed is continued till all the contextual vectors are exhausted.

5 Results

5.1 Timeline generated by Model 1

The timeline generated by model 1 is shown in Table 1.

5.2 Timeline generated by Model 2

The timeline generated by model 2 is shown in Table 2.

Timeline
A win to start the season for CSK. @ChennaiIPL #IPL
Match 3. it is all over! Delhi Capitals won by 37 runs #MIvDC #VIVOIPL
Match 4. it is all over! Kings XI Punjab won by 14 runs #RRvKXIP #VIVOIPL
Match 5. it is all over! Chennai Super Kings won by 6 wickets #DCvCSK #VIVOIPL
Match 6. it is all over! Kolkata Knight Riders won by 28 runs #KKRvKXIP #VIVOIPL
Match 7. it is all over! Mumbai Indians won by 6 runs #RCBvMI #VIVOIPL
Match 8. it is all over! Sunrisers Hyderabad won by 5 wickets #SRHvRR #VIVOIPL
#DC win! Kagiso Rabada concedes only 7 runs in the Super Over and closes the game for @DelhiCapitals. DCvKKR IPL2019
Match 12. it is all over! Chennai Super Kings won by 8 runs #CSKvRR #VIVOIPL
KXIP have done the unbelievable! DC came down from 144/3 to 152/10! They lost 7 wickets for just 9 runs! KXIP beat DC by 14 runs! #KXIPvDC #IPL2019

Table 1: Output example for model 1 on #IPL,

Timeline
ChinnaThala @ImRaina wins the race against Virat Kohli to become the first batsman to scale 5000 #VIVOIPL runs #CSKvRCB
Youngest to take five-fers in IPL: SRHvsMI IPL2019 21y 204d J Unadkat RCB v DD Delhi 2013 22y 137d A JOSEPH MI v SRH Hyderabad 2019
BIG. BIG. CONTROVERSY. Jos Buttler mankaded by Ashwin #RRvKXIP #VIVOIPL
Mishra forces the set Raina to edge to Pant, CSK 98/3. MSD walks out to a big roar #DCvCSK #IPL2019
Yorked! Russell is bowled, but wait... Kings XI penalised for not having the required number of fielders inside the 30-yard circle #KKRvKXIP #IPL2019
When Bumrah bowled to de Villiers and Kohli, everything took a backseat as the fans watched in awe. It was not bullying. It was never meant to be. It was mesmerising #RCBvMI
Sanju Samson takes Bhuvi to the cleaners in the 18th over #SRHvRR

Table 2: Output example for model 2 on #IPL,

parameter	precision	recall	f1-score
Match	1.00	0.81	0.90
No Match	0.00	0.00	0.00
Macro Avg	0.50	0.41	0.45
Weighted Avg	1.00	0.81	0.90

Table 3: Classification Report for Model 1.

parameter	precision	recall	f1-score
Match	1.00	0.76	0.86
No Match	0.00	0.00	0.00
Macro Avg	0.50	0.38	0.43
Weighted Avg	1.00	0.76	0.86

Table 4: Classification Report for Model 2.

6 Evaluation

6.1 Model 1

Model 1 was evaluated by the % match with the results data scrapped from https://www.hindustantimes.com/ipl/results/.The following criteria was used to check the number of matches between the tweet timeline tokens and scrapped data tokens:

- If the number of intersections is greater than 3, then it is considered to be a match. The reason behind 3 as a threshold is that the maximum team name size is 3 and the tweet comprising of the result is likely to contain only the winning team.
- If the number of intersections in less than equal to 3, then we check if the intersection contains any of the team name token. If it does then is taken as a match. Else no match.
- If the length of intersection is 0 then it was considered no match.

Based on the above criteria and assumption, the evaluation metrics for model 1 turned out as follows:

Accuracy: 81.25%

Confusion Matrix: [[39 9] [0 0]]

Classification report: Classification Report for model 1 is shown in Table 3. **Support** remains 48 for all the parameters except **No Match** where support is 0.

6.2 Model 2

Model 2 was evaluated by the % match with the results data scrapped from https://en.wikipedia.org/wiki/2019_Indian_Premier_League#Matches. The following criteria was used to check the number of matches between the tweet timeline tokens and scrapped data tokens:

- If there are only team names in the intersection then it is considered as no match.
- If the length of intersection is 0 then it is considered no match.
- All other cases are considered as a match.

Based on the above criteria and assumption, the evaluation metrics for model 2 turned out as follows:

Accuracy: 76.0%

Confusion Matrix: [[19 6] [0 0]]

Classification report: Classification Report for model 2 is shown in Table 4. **Support** remains 25 for all the parameters except **No Match** where support is 0.

7 Literature Survey

Omar Alonso in journal (Omar Alonso, 2017) propose a model which use contextual vectors to capture the context of the tweet and weighted sum of retweets likes and shares for the user engagement score. Concept of shingles and Jaccard similarity was used to remove duplicate tweets. Omar Alonso in journal (Omar Alonso, 2015) used the TF-IDF scoring metric to compute contextual score. They sorted the tweets using the sum of the contextual score and user engagement score for a given day and added the tweet with the highest combined score to timeline.

References

Fernando Diaz Omar Alonso, Serge-Eric Tremblay. 2017. Automatic generation of event timelines from socialdata. *WebSci '17: Proceedings of the 2017 ACM on Web Science Conference*.

Kartikay Khandelwal Shankar Kalyanaraman Omar Alonso, Sushma Bannur. 2015. The world conversation: Web page metadata generation from social sources. *Conference: the 24th International Conference*.