Event Detection in Live Twitter Streams Using Tf-Idf and Clustering Algorithms

Tavishi Jain, Bhavya Singh, and Rupesh Kumar Dewang

Abstract Twitter is a fast emerging form of social media. People use Twitter as a platform to report real-life events. The present paper focuses on extracting those events by analyzing the text stream on Twitter. This paper presents methods that take the tweets in real time as input and generate clusters of tweets denoting different communities as output. The tweets are collected using spark streaming and then pre-processed, and a key graph of keywords is constructed using the tf-idf method. Further community detection is applied on this key graph to generate clusters of tweets as a result.

Keywords Event detection · Live tweets · Tf-Idf · K-mean clustering algorithms · Key graph

1 Introduction

Twitter has been the most promising news delivery and social media platform for the internet user. As time is passing, the number of users on Twitter is also increasing exponentially. Since the requirements are too high, so we need some technology to satisfy those requirements. A large amount of data in the form of tweets, images, videos and animation is being generated very frequently. There are social media platforms like Facebook, Instagram and YouTube which have changed the way of satisfying needs by using technology. All these social media giants have changed the traditional way of connecting people over the internet. Since Twitter is one of the most popular platforms, that's why we are using Twitter for Event Detection purposes. Twitter is a platform where tweets related to any local or global events are written by users in a few seconds. Natural events like earthquakes, disease outbreaks, etc. are significantly detected by performing event detection on Twitter data [2]. Twitter easily figure out what is happening right now. Tweets are real time data for analysis covering different subjects from various sources. Event detection about emerging

T. Jain · B. Singh · R. K. Dewang (✉)
Department of Computer Science & Engineering, Motilal Nehru National Institute of Technology Allahabad, Prayagraj, India
e-mail: rupeshdewang@mnnit.ac.in

S. Sharma et al. (eds.), *Data, Engineering and Applications*, Lecture Notes in Electrical Engineering 907, https://doi.org/10.1007/978-981-19-4687-5_36

events is highly important if it is performed on real time data [9]. Different people are interested in different types of news and events. Some people want to know about local events [18]. Business oriented organizations are interested in sponsoring their product and services to their favourite customers [4, 5, 13]. Event detection can fulfill the requirements of these people. Event detection can also help in the detection of natural disasters and warning people faster than any other media [7]. Event detection is also helpful in crime detection for example bomb blast or terrorist attack. Event detection in Twitter is effected due to a large amount of meaningless data [15]. Data which is being generated on Twitter is not of high quality, So various high scalable and efficient techniques are being for the pre-processing of the data to make it usable for event detection. Users prefer to write short tweets and they generally use distorted words [12]. Different events consist of different numbers of participants, different time periods and relationships [14]. Real time collection is a big challenge in real time event detection because every time a large amount of data is created on Twitter. Real time event detection has different applications and with technology support. In the present paper, we have proposed the detection of events in live Twitter streams. For construction, the key graph tf-idf scheme is used. The key graph is further used for detecting the communities using the Betweenness Centrality score and then finally clusters are formed using k-mean with cosine similarity. The rest of this paper is structured as follows: Sect. 2 introduces the most relevant related works. Section 3 provides the used dataset description. Section 4 provides a detailed description of the proposed approach. Experimental Setup and Results are discussed in Sect. 5. Section 6 finally concludes the paper with future work.

2 Related Work

New Event Detection models do a single step increscent clustering algorithm. When a new document is collected similarity is found betweenthe document along with computation on known events and selection is done on the basis of maximum similarity. A threshold is set, if similarity is more than threshold then the document belongs to a known event else considered as a new event. A Key Graph based approach is used by Hassan Sayyadi, Matthew Hurst and Alexey Markov [6]. They built a key graph based on co-occurrence and used betweenness centrality algorithm for clustering keywords. In this algorithm, they count all keywords in a single community, though a subgroup of the keyword may be better. Modified version of TF/IDF was developed by Allen et al. [1] also penalized the threshold in which the threshold is amerced by the time distance between the event and the document. Calculation of IDF using online clustering algorithm is required as to know future document features. The same method is used by Sayyadi et al. in [14]. A supplementary data set is used by Allen et al. [1] in order to find IDF while Yang et al. [16] proposed an increscent IDF factor. Time difference was the basis to find the similarity between documents and events along with consideration of time window and a decay factor that Yang et al. used. To take out past event Yang et. al [16] put forwarded an agglomerative

clustering, GAC (augmented Group Average Clustering). In order to handle cluster quality, they used looping bucketing and re-clustering. A probabilistic model was put forwarded by (Li et al. 2005) for RED and in order to maximize log-likelihood of the distributions they used the Expectation Maximization (EM) algorithm. A significant number of events is required by such algorithm which is not feasible in practice. Li et al. found an approximation of event counts from the article count- time distribution [11]. Event detection models usually use similar algorithms, many different document representation, distance or similarity metrics, and clustering algorithms mentioned in the theory [3, 8, 10, 17].

3 Dataset Used

In this paper, we have collected the tweets from Twitter based on a search keyword (hashtag) of "Facebook" and then, this data is given to the spark client where the data is processed in batches. The batch interval used is 1 min in which 80–90 tweets are collected in one batch of spark. So, our one iteration of algorithm runs on approximately 80–90 tweets.

4 Proposed Approach

We have collected the data set from Twitter in form of batches in real time using Spark streaming API. Prepossessing is performed on each batch of data which includes Tokenization, removal of Stop words and Stemming. Our approach comprises the following phases: (1) Construction of Key Graph; (2) Identifying Communities in Key Graph; (3) Document Clustering using K-Mean with Cosine similarity.

4.1 Construction of Key Graph

Key graph is made using pre-processed data. We first calculate term frequency (TF), document frequency (DF) and inverse document frequency (IDF). Remove keywords with low DF because these keywords are not useful. Key graph is constructed by taking the remaining keyword as a node of the graph. Co-occurrence of keyword represents the relationship between keywords. Make an edge between keywords which co-occur in the same document. Some edges are present as noise in the key graph. We remove edges which are not satisfying following two conditions-

(1) If the co-occurrence of two keywords is below some specified threshold value then the edge between these two keywords is removed.
(2) For the edge of the node 'X' and 'Y', if there is a potential probability of viewing.

'X' in the documents if 'Y' is present in the document and if 'X' exists then the condition of 'Y' in the documents are calculated and if both are smaller than the threshold, then the edge will be removed.

Algorithm 1 Construction of Key graph from tweets	
Input: Keyword (hashtag) of tweets	
Output: Key graph (kgi) of keywords ki	
1	BEGIN
2	FOR Extracted all tweets ti (specified time limit) in real time for corresponding hashtag
3	Pull out words wi, noun ni, noun phrases npi, adverb ai adjective adi and named entities nei from ti as keywords ki
4	END FOR
5	FOR all keywords ki extracted from tweets ti do
6	Calculate Document frequency df of all tweets ti
7	END FOR
8	Remove keywords ki with low Document frequency df
9	Construct the keyword ki co-occurrence graph cgi as follows:
	– Generate single vertex vk for each keyword ki
	– Generate single link li;j for every set of co-occurring keywords ki and kj
10	END

4.2 Identifying Communities in Key Graph

Edges of the graph represent the relationship of keywords and the whole key graph presents a social network of keywords. Communities of keywords are represented by highly dense graphs. Communities have a large number of edges because there exist more relationships between keywords in the community. Between different communities, there are few links. Betweenness Centrality score is used to find the edges between communities. Betweenness centrality score for any node is the number of shortest paths between all pairs of graphs which pass through that node. Edges between communities come across many shortest paths so these edges have a high betweenness centrality score. These edges are connecting the different communities so after removing edges with high betweenness centrality score we will get different clusters of keywords and each cluster represent different communities. Each community is the hypothesis of an event. This is an iterative process. In each iteration, we remove edges with low BC scores. For this, we first calculate the BC score by finding the shortest path between all node pairs of graphs. We use BFS to find the shortest path. If two edges have the same BC then the edge with low conditional probability is removed. If the edges which are removed belong to two different communities and have high conditional probability then edge is duplicated in both communities and

then removed. We again calculate BC score for the next iteration and perform the same operation until all edges with a high score are not removed. Finally, we get a cluster of keywords representing events.

Algorithm 2 Identifying Topic Features	
Input: Set of key graphs (kgi)	
Output: Topic (Ti = t1;t2;t3:::tn) feature sets (Fi = f1; f2; f3::: fm)	
1	Separate Key Graph (kgi) in communities coi of strongly related (kiecoi) keywords ki as topics Ti
2	FOR all topic tieT do
3	Place all keywords ki in the related feature vector (FVi = f v1; f v2; f v3::: f vn) of related keyword partition fp
4	END FOR

4.3 Document Clustering Using K-Mean

Each group of keywords make a synthetic document called as Key Document. All documents can be grouped in the original collection, similar to this synthetic document, thus a group of periodical documents is obtained. To discover document clusters, k-mean with cosine similarity is used. Sometimes keywords are common in an important document, or the main document has few keywords. It makes a general class of documents. These important documents make a set of documents related to higher levels. For example, a key document containing only “Modi”, “Rahul”, “Votes”, and “Election” indicate the Indian general elections which is not a separate event. We can find such important documents with help of the similarity of documents related to an important document. Documents are data points, and the variance of documents related to such key documents will be very large. So, we calculate the variance of documents for every main document and then filter those key documents that have a large variance. This makes it easier to find those key documents which truly represent events. However, documents allocated to key clusters are based on the cosine similarity of documents to the key documents, some of the documents are filtered later to reduce noise. Apart from these, the similarity between the documents and the centroid of each cluster are also calculated and filter those documents which have low cosine similarity to the centroid.

Algorithm 3 Document Clustering (K-mean with cosine similarity) using Topics	
Input: Topic (Ti = t1;t2;t3:::tn) feature sets (Fi = f1; f2; f3::: fm)	
Output: Topics (Tl = t1;t2;t3:::tm) under the cluster documents (cd1;cd2;cd3:::cdn)	
1	FOR all topic ti
2	Initialize the k value and select as the initial centroid value ci&cj
3	Calculate the similarity (s) of topic ti for document di: s(ti = di) = = cos(di;FVi) $\sum$(tieT)cos(di;FVi)
4	Recalculate the centroid of each cluster until centroid value not changed
5	END FOR
6	FOR complete topic ti and tl do
7	IF the topic T overlap of ti and tl
8	Combined ti and tl into a new topic NT where FVi = f vti + f vtl
9	END IF
10	END FOR

4.4 Example of Construction Keygrpah, Identifying Topic Features and Document Clustering (K-means with Cosine Similarities) Using Topics

Consider the graph in the above Fig. 1, the nodes represent the keywords and the edges represent the co-occurrence of the two keywords between which it is present. Consider all the edge weights to be equal to 1. The following table gives the values of betweenness centrality scores calculated for every edge using the Breadth First Search algorithm.

Cosine Similarity

Let us say we have multiple documents and we need to determine the similarity between those documents. Let us name the two documents as document1 and document2 respectively. A document can be represented by a bag of terms or a long vector, with each attribute recording the frequency of a particular term (such as word, keyword or phrase) in the document. We will be having two term frequency vectors (d1 and d2). Table 2 shows that d1 and d2 denote term frequency in doc1 and doc2 respectively.

$$cos\theta = \frac{d1.d2}{|d1||d2|}$$

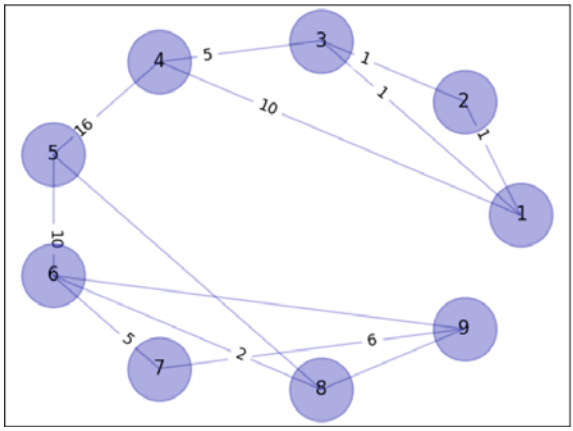

Fig. 1 Initial keyword graph

Finding similarity between document d1 and d2:

d1 = 5,0,3,0,2,0 and d2 = 3,0,2,0,1,1

1. First, calculate the dot product of these two documents

$$d1.d2 = 5 * 3 + 0 * 0 + 3 * 2 + 0 * 0 + 2 * 1 + 0 * 1 = 23$$

2. Then calculate kd1kandkd2k

$$||d1|| = \sqrt{5 * 5 + 0 * 0 + 3 * 3 + 0 * 0 + 2 * 2 + 0 * 0} = 6.164$$

$$||d2|| = \sqrt{3 * 3 + 0 * 0 + 2 * 2 + 0 * 0 + 1 * 1 + 1 * 1} = 3.873$$

3. Calculate cosine similarity:

$$\cos(d1, d2) = \frac{23}{6.164 * 3.873} = 0.96$$

5 Experimental Setup and Results

We have used the above presented algorithms on batches of tweets formed by passing the live tweets coming from the spark streaming API by using a search keyword. Batch interval used for the experiment is 1 min in which 80–90 tweets are collected in one batch of spark. So, our one iteration of the algorithm runs on approximately 80–90 tweets. In the first step of the algorithm, a key graph is formed which contains nodes and edges between any two nodes if they co-occur. Here we obtained approximately

40 keywords(nodes). After that, clusters of keywords are formed from the key graph and then tweets are categorized into these clusters of keywords using cosine similarity (Tables 1 and 2).

Table 1 Betweenness centrality score

Node 1	Node 2	Betweenness centrality
1	2	1
1	3	1
1	4	10
1	5	0
1	6	0
1	7	0
1	8	0
2	3	1
2	4	0
2	5	0
2	6	0
2	7	0
2	8	0
3	4	0
3	5	0
3	6	0
3	7	0
3	8	0
4	5	16
4	6	0
4	7	0
4	8	0
5	6	10
5	7	0
5	8	5
6	7	6
6	8	2
7	8	0

Table 2 Calculation example

Doc	Team	Coach	Hockey	Baseball	Soccer	Penalty
d 1	5	0	3	0	2	0
d 2	3	0	2	0	1	1

5.1 Batch wise Key graphs And Categorization of Tweets

In this section, we have shown the results obtained by running the above approach in batches. The algorithm is applied on each batch and the results are hence depicted batch-wise.

Batch 1 Results In Fig. 2, the initial key graph and final graph of batch 1 results. The figure shows the graph formed initially after pre-processing of tweets and also filtering of keywords based on document frequency. Here, the keywords represent the node of the graph and the edges between them show their co-occurrence. The final key graph is the formed after clustering of keywords phase of the algorithm. It contains edges only between those nodes which belong to a particular cluster. Table 3 shows the tweets belonging to a particular category. For example, number 2 represents the tweet belonging to this category.

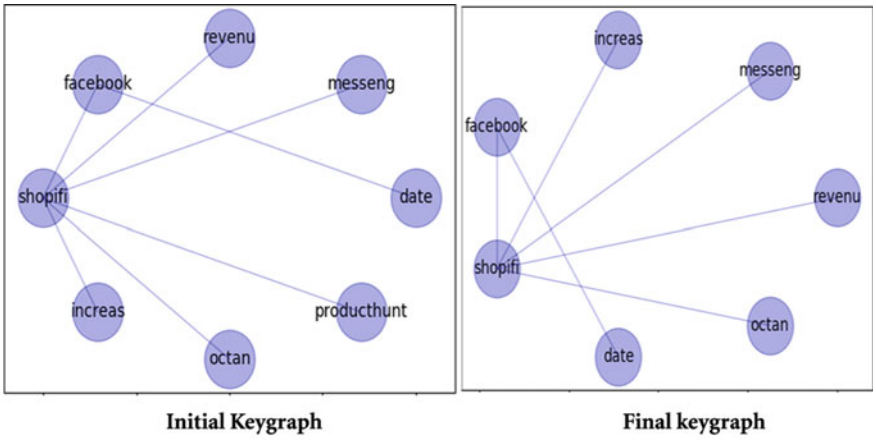

Fig. 2 Initial and Final Key Graph of Batch 1 Results

Table 3 Categorization of tweets btach-1

Category No	Tweets Belonging to this category
2	['WhatsApp co-founder flees Facebook just as fb gets underway http://t.co/yF6wsKEnOi']
6	['TR @Mdproductionsmd: HELLO MAY- What a month it is going to be with some exciting things happening. Check out what we have on at MD HQ th']
16	['Facebook partners with RED to develop a high-end, professional VR camera https://t.coMSVSdADKVg]
35	['Dont worry Son day of judgement is for a reason.']
37	['#facebook #facebookmarketing #zuck #page #b']
41	['RI @blackmirror: 1000 simulations completed. https://t.co/mpxcDitBVQ']

Batch 2 Results Similarly, for batch 2, Fig. 3 represents the initial key graph and final key graph. Table 4 shows the categorization of tweets.

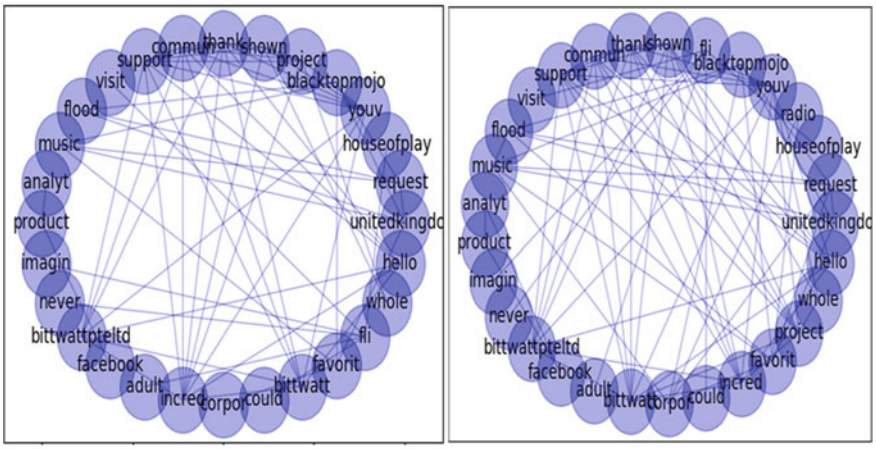

Fig. 3 Initial and Final Key Graph of Batch 2 Results

Table 4 Categorization of tweets btach-2

Category No	Tweets Belonging to this category
12	['And that is why heaven and hell are for ETERNITY! https://t.co/k2RQZA74KT']
17	['Worth a quick refresher, for all of us parents, https://t.co/ptGLaWp5kZ']
22	['Facebook. Into twitter. http']
23	['Facebook is trying to close the book on Cambridge Analytica: https://t.co/SYcxIfCRcY']
24	['Seven year old girl owns riding in the rain']
25	['Reminder- in 1 h Online Marketing Facebook Series, click on the link an d register! https://t.co/XgoeDMuNJO']
33	[1- red coloured vinyl I https://t.co/rRiGAUatt6']
40	['RT @blackmirror: 1000 simulations completed. https://t.co/mpxcDitBVQ']
71	[Spotify, Soundcloud, and GoPro can now let their users to share to facebook and Instagram stories https://t.co/mqkISPEHmv]
72	['I want this!! https://t.co/fa1Xz8YF8P']
82	Not new product on Product Hunt: Facebook Analytics Facebook's new analytics app on iOS and Android https://t.co/cRFz82Akw8']

6 Conclusion and Future Work

In this paper, we have used a key graph based approach for clustering keywords to find out events from live tweets. Presently, the batch interval used is small due to the amount of memory spark needed. Therefore, further work can be done to run the above algorithm on a distributed system so that algorithm can be run on a large dataset and can produce better results. Apart from that, the complexity of the algorithm becomes too high when used for live processing therefore, another approach needs to be formulated to reduce the complexity.

References

1. Allan J, Papka R, Lavrenko V (1998) On-line new event detection and tracking. In: Sigir, vol 98, pp 37–45. Citeseer
2. Boyd DM, Ellison NB (2007) Social network sites: definition, history, and scholarship. J Comput-Mediat Commun 13(1):210–230
3. Brants T, Chen F, Farahat A (2003) A system for new event detection. In: Proceedings of the 26th annual international ACM SIGIR conference on research and development in informaion retrieval. ACM, pp 330–337
4. Dewang RK, Singh AK (2018) State-of-art approaches for review spammer detection: a survey. J Intell Inf Syst 50(2):231–264
5. Farzindar A, Inkpen D (2012) Proceedings of the workshop on semantic analysis in social media. In: Proceedings of the workshop on semantic analysis in social media
6. Hasan M, Orgun MA, Schwitter R (2018). Real-time event detection from the twitter data stream using the twitternews+ framework. Inf Process Manag
7. Java A, Song X, Finin T, Tseng B (2007) Why we twitter: understanding microblogging usage and communities. In Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on web mining and social network analysis. ACM, pp 56–65
8. Kumaran G, Allan J (2004) Text classification and named entities for new event detection. In: Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 297–304
9. Kwak H, Lee C, Park H, Moon S (2010) What is twitter, a social network or a news media? In: Proceedings of the 19th international conference on world wide web. ACM, pp 591–600
10. Lam W, Meng H, Wong K, Yen J (2001) Using contextual analysis for news event detection. Int J Intell Syst 16(4):525–546
11. Li Z, Wang B, Li M, Ma W-Y (2005) A probabilistic model for retrospective news event detection. In Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 106–113
12. Nallapati R, Feng A, Peng F, Allan J (2004) Event threading within news topics. In: Proceedings of the thirteenth ACM international conference on information and knowledge management. ACM, pp 446–453
13. Sabeeh A, Dewang RK (2018) Comparison, classification and survey of aspect based sentiment analysis. In: International conference on advanced informatics for computing research. Springer, pp 612–629
14. Sayyadi H, Hurst M, Markov A (2009) Event detection and tracking in social streams. In: Third international AAAI conference on weblogs and social media
15. Wang X, Gerber MS, Brown DE (2012) Automatic crime prediction using events extracted from twitter posts. In: International conference on social computing, behavioral-cultural modeling, and prediction. Springer, pp 231–238

16. Yang Y, Pierce T, Carbonell JG (1998) A study on retrospective and on-line event detection
17. Yang Y, Zhang J, Carbonell J, Jin C (2002) Topic-conditioned novelty detection. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 688–693
18. Yuan Q, Cong G, Ma Z, Sun A, Thalmann NM (2013) Who, where, when and what: discover spatio-temporal topics for twitter users. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 605–613