Bursty Event Detection Model for Twitter

Anuradha Goswami[1(✉)], Ajey Kumar[2], and Dhanya Pramod[2]

[1] Symbiosis Institute of Business Management, Symbiosis International (Deemed University), Bengaluru, India
anuradha@sibm.edu.in

[2] Symbiosis Centre for Information Technology, Symbiosis International (Deemed University), Pune, India
{ajeykumar,dhanya}@scit.edu

Abstract. Huge amount of diversified information in the form of multimedia data gets uploaded to Online Social Network platform every second. This eventually gets a sudden burst during high impact events. Twitter platform plays a very important role during these events in the process of diffusion of this information across the entire social network of users. The real challenge is in the analysis of tweet during these bursty events when data gets generated in large volume with high arrival rate. Under this circumstances, near real-time detection of bursty event should be implemented to match up the speed of the information diffusion which demands efficient algorithms. In this paper a bursty event detection algorithm is proposed which considers a dynamic set of tweets in every time window and generates optimal k topics per window of a bursty event. This research has also studied the goodness of the topics produced across the different time windows. Our proposed model is successful in creating better semantically coherent and contextual topics for bursty event as compared to the other state of the art techniques such as Latent Dirichlet Allocation Model, Gibbs Sampling Dirichlet Mixture Model and Gamma-Poisson Mixture Topic Model.

Keywords: Event · Burst · Topic Modelling · LDA · GSDMM · GPM · Coherence Measures

1 Introduction

The inherent dynamism of Online Social Network (OSN) lies in the huge amount of varied information getting uploaded to OSN platform every second in the form of multimedia data from different events [1]. Any latest happening or prolonged event occurring around the globe has its footprint in OSN in some way or the other [2]. Any event 'E' is defined as a happening which is probable to occur in the next time span or duration [3]. On Twitter, to describe an event, the users use #tag (or hashtag) or @ symbol, which further facilitates in coupling different events with each other directly or indirectly [4]. According to [5], both unplanned events like natural disasters and planned events such as ICC World Cup Twenty20 on Twitter, which either can be trendy or non-trendy, can be bursty. The bursty behavior of an event is directly proportional to the rate of information

S. Devismes et al. (Eds.): ICDCIT 2024, LNCS 14501, pp. 338–355, 2024.
https://doi.org/10.1007/978-3-031-50583-6_23

diffusion over Twitter or any other OSN [5]. These bursty events which can be called as 'trends' have the capability to catch the attention of huge number of users almost instantly [6].

Real-time stream of data from Twitter help the researchers to analyze real-world bursty events within a specific timeline. Every tweet is accompanied by a timestamp of its creation, username, and biographical sketch making it easier for the researchers to take up the challenge of automatically detect and analyze the bursty events. The real challenge is in the analysis of tweet text during bursty events when data comes in large volume with high arrival rate [7]. Under this circumstance, close to real-time detection of bursty event should be implemented to match up the speed of the information diffusion which demands efficient algorithms. Prediction of bursty events has got important implication in the field of social, political, several planned or unplanned cases of events.

There are few algorithms proposed for detection of bursty events in literature. ~~Am~~Most of the approaches use fix term of vocabulary, requires a set of query words, needs number of topics to discover, and also have a set threshold value in order to define the bursty event cluster [8]. Additionally, most of the techniques use a vector-space model to represent the tweets, given the dimension of the vector same as the word vocabulary [9]. Researches who have considered streaming of data, assumed a dynamic word vocabulary for bursty event detection which changes with time [2] Some recent literatures have used deep learning techniques, attention mechanisms and network structures too to detect bursty events [10–12]. To the best of our knowledge, none of them have studied the goodness of the topics produced across the different time windows. In this paper, a bursty event detection algorithm is proposed which considers a dynamic set of tweets in every time window and generates optimal topics per window of a bursty event.

The rest of the chapter is organized as follows: Sect. 2 elaborates the review of literature. Section 3 details the proposed burst detection framework and the corresponding algorithm. The implementation, evaluation results and analysis along with The experimental setup details, datasets description, the preparation of the datasets is illustrated in Sect. 4. Discussion on the results is performed in Sect. 5, followed by conclusion in Sect. 6.

2 Review of Literature

A very traditional work used statistical techniques and tests on data distribution to extract bursty keywords topics in an event [13]. In online mode, Twitter Monitor tool was designed by [6] which detects emerging topic trends in Twitter stream. Individual keywords buzz was used to identify trends in two steps. The occurrence of individual keywords in tweets is measured to identify the bursts. This was modified by a study by [7] through an algorithm named '*Window Variation Keyword Burst Detection'* where a scalable and fast online procedure was proposed for detecting online bursty keywords. A study by [14], proposed a different approach for detection of online bursty keywords named as EDCoW (Event Detection with Clustering of Wavelet-based signals) model. EDCoW considers individual word as signals through an application of analysis of wavelet to frequencies of words. Emerging temporal trends were interpreted in a study

by [15]. Firstly, a taxonomy of trends was found in the large dataset of Twitter. Secondly, the study found out primary features through which categorization of trends can be accomplished using each trend category features.

A segment-based system for detection of bursty events was introduced by [16] called *'Twevent'*. Twevent first maps and detects event segments from bursty tweet segments, followed by clustering/grouping the segments of events into events by using their distribution of frequency and similarity in content. An interesting study by [17] researched on identification of bursty topics early in the timeline with a large-scale real-time data from Twitter. Tool named TopicSketch proposed by the study, was an integrated solution consisting of two stages. In the first stage the model maintains three measures viz. total count of the tweets, each word occurrence and the respective word pairs. These measures were used as an indicator of a sudden burst in the attention of users towards the tweet, which further facilitates in the bursty topic detection. In the second stage, a topic model based on sketch was used to depict the bursty topics and their surge based on the statistics monitored in the sketch of the data. Incremental clustering methodology was used by studies [18, 19] to detect burst events where evolution of events was also experimented and solved [20]. The new arrival of tweets results in updating of the bursty topics for incremental clustering technique. Study [21] proposed a topic model, which is incremental in nature and includes the temporal features of texts, named as 'Bursty Event dEtection (BEE)' to detect the bursty events.

EventRadar was proposed by [22] which deals with activity burst in a localized area. A geo burst algorithm was proposed which was implemented using geo-tagged tweets containing information on location, time and text of the tweets. The topic clusters/ groups which are geographically tagged are created as candidate events per query window. A statistical approach was followed by a study by [23] on the Twitter platform. The study showed that a sudden spike in the tweet frequency follows a log-normal distribution with respect to the arrival of data. The data or tweet burstiness of any event was mapped with the z-score of the rate of tweet arrival. Real-time streaming text was used by study [5] to understand the bursty attitude of events. This study explored various event features and used clustering to classify the features as per their similarity index.

A study by [24] considered cross social media influence and unsupervised clustering for burst detection model. In this work, the time series social media data were divided into time slices and for each slice the burst word features in that time window were also calculated. The burst degree of words was calculated by fusing the three burst features in the time window, post which burst word set got generated. Finally, agglomerative hierarchical clustering technique was applied to cluster the word set to convert it into event. A novel graph based technique called KEvent was proposed by [25] where tweets were divided into separate bins to extract bursty keyphrases. The word2vec model was used to create a weighted keyphrase graph from the keyphrases. Final event detection was performed using Markov clustering.

Lately, deep learning algorithms [10, 11] coupled with attention networks is used by the researchers to handle the temporal dynamics of emerging keywords to detect events from tweets.

3 Proposed Burst Detection Model

Keywords/terms/words/tokens are synonymous for our research work and are interchangeably used throughout. A stepwise burst detection framework is detailed in Fig. 1.

The proposed burst detection algorithm is an extension of the Window Variation Keyword Burst detection algorithm given by [7]. The extended features are:

a) *Threshold:* A threshold in included for: *first* in selecting the most frequent words per window in ***Algorithm 3*** and then in ***Algorithm 4,*** for selecting the bursty keywords across two consecutive windows. This approach helps in the detection of *appropriate the bursty topics.*
b) *Topic Creation:* After the list of bursty keywords is obtained in ***Algorithm 4***, in the end we generate optimal *k* topics out of the bursty keywords per window. This approach helps in *identifying the trending topics.*
c) *Coherence Scores of Topics:* ***Algorithm 5*** generates coherence scores of optimal *k* topics in each time window across the bursty event. This approach helps in *identifying bursty topics of similar context per window.*

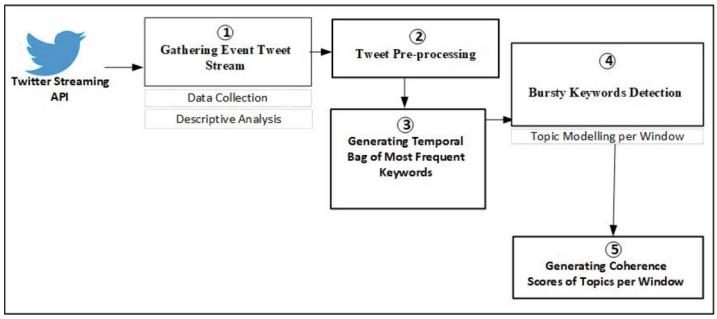

Fig. 1. Burst Detection Framework

The detailed explanation of each algorithm is given in piecewise manner according to the modules maintained in the framework, with their respective input, output, and the corresponding pseudocode. Table 1 provides a description of the important variables used in the pseudocode for a better understanding. The algorithms should be read keeping the framework, variable description and the pseudocode synced with one another.

i) ***Gathering Event Tweet Stream:*** The process starts with collection of tweets (*G_Stream1*) using Twitter streaming API and converting into non-duplicated tweets (*G_Stream2*) as shown in Table 2. Tweet can be regular tweet or a retweet. Each tweet is of 140 or 280 characters. The events selected for our research are *natural disaster events.*

ii) ***Tweet Pre-processing:*** The next module in the pipeline is data pre-processing, shown in Table 3 which involves preparation of the dataset to make it appropriate to feed for generating the most frequent bag of keywords.

iii) ***Generating Temporal Bag of Most Frequent Keywords:*** The aim in this module is to output the most frequent words/tokens appeared in the respective bag of tweets within a particular time window. Time window size, *window_size* is decided on understanding the dataset from the descriptive analysis. We check on the total number of days' data available and the *burst_datasize.* Final count of number of time windows, *window_num* is dependent on the *window_size* considered and the *burst_datasize.* Collection of pre-processed tweets is divided into bag of tweets *tweet_bag* as per the *window_size.* Every window starts with an initial timestamp *init_time.* For the first window, the timestamp is zero. Following this, every time window will have a duration according to the *window_size.* The finishing timestamp of a window *end_time* is calculated by adding *window_size* to *init_time* of that window. All the *init_time* values for all the windows are stored in *window_init_time* for future use. A snapshot of the windowing system referred in our algorithm is shown in Fig. 2. Here *Tw, Tw + 1,* refers to the incoming sequential stream of tweets. For every window, the bag of tweets is created, where tweets are further tokenized to get the bag of words total_win_words. For each word in the bag, word frequency word_freq per window-wise is calculated.

The proposed algorithm has applied a threshold for considering the most frequent words per window (most_frequent_words). A threshold of 20% of the total number of tokens per window is considered for selecting the most_frequent_words for a particular window. The threshold value is based on the state-of-the-art study by [26]. In this module, we recorded the set of most_frequent_words along with their respective frequencies of occurrence per window, window-wise total number of tokens/words, total number of tweets (no_of_tweets) per window number for further use in the rest of the modules. The pseudocode of the stated process of the algorithm is given in Table 4.

iv) ***Bursty Keywords Detection:*** The input to this module is *G_Queue2,* consisting of most frequent keywords per time window. The purpose is to find out the how many most frequent keywords are eligible of becoming bursty keywords per time window and model these bursty keyword into window-wise topics as given in

Table 1. Variable Names and Its Description

Variable Name	Description
G_Stream1	Global Stream of Raw tweets (in JSON)
G_Stream2	Global Stream of unique Raw tweets (in CSV) with selected features
G_Queue1	Collection of Pre-processed Tweets and tokenized Tweets
G_Queue2	Collection of Most frequent keyword bags window-wise
no_of_tweets	Total number of tweets per window
burst_datasize	Total number of tweets in the whole dataset
window_size	Size of the time window (in seconds)
window_num	Signifies the number of each time window
window_init_time	Stores the earliest timestamp (*init_time*) for every time window
init_time	Initial timestamp of the tweets in at the starting of every time window (in seconds)
end_time	*init_time* + *window_size* (in seconds) for every time window
word_freq	Frequency of a keyword in a window
tweet_bag	Collection of tweets in a window
most_frequent_words	Collection of most frequent words from a single window
total_win_words	Total number of words in one window
window_total_words	Collection of all list of *total_win_words* for all windows
window1, window2	Two consecutive time windows
Hash_Dict	Dictionary data structure storing the words which are present in two consecutive time windows
sorted_word_rank	List of keywords which have increasing probability of presence across consecutive windows
Sorted_word_rank_ave_cutoff	Selected bursty keywords list having a cut-off frequency more than the average frequency of the total number of words in *sorted_word_rank*
all_optimal_k_topics_per_window	Collection of optimal k topics window-wise for all windows
topics_coherence_score	Coherence scores of the optimal k topics measured by using the coherence frameworks- *UMass*, *UCI*, *NPMI*, C_V and *word2vec*
CoherenceModel()	Function as given in the Genism package of python

Table 2. Algorithm for Gathering Event Tweet Stream

Algorithm 1: Gathering Event Tweet Stream	
Input: *Stream of raw data tweets in .json format (G_Stream1)* ***Output:*** *Raw data tweets with no duplicates in .csv format (G_Stream2)*	
1.	*Tweet ids used to collect raw data tweets G_Stream1 in JSON object*
2.	*Conversion of JSON to CSV raw data tweets G_Stream2*
3.	*Duplicate tweets removal from G_Stream2*
4.	*Perform Descriptive Analysis on G_Stream2*

Table 3. Algorithm for Tweet Pre-processing

Algorithm 2: Tweet Pre-processing	
Input: *Raw data tweets without duplicates in .csv format (G_Stream2)* ***Output:*** *Pre-processed Cleaned data tweets (G_Queue1) in .csv format*	
1.	*Removal of rows having missing values for any attributes*
2.	*Removal of punctuation marks, URL's, extra whitespaces between words, and numbers*
3.	*Convert the tweets into lowercase*
4.	*Removal of stopwords and extended stopwords*
5.	*Tweets← Store all the cleaned tweets*
6.	*tokenized_Tweets ← Tokenize each tweets in Tweets*
7.	*Store Tweets and tokenized_Tweets in G_Queue1*

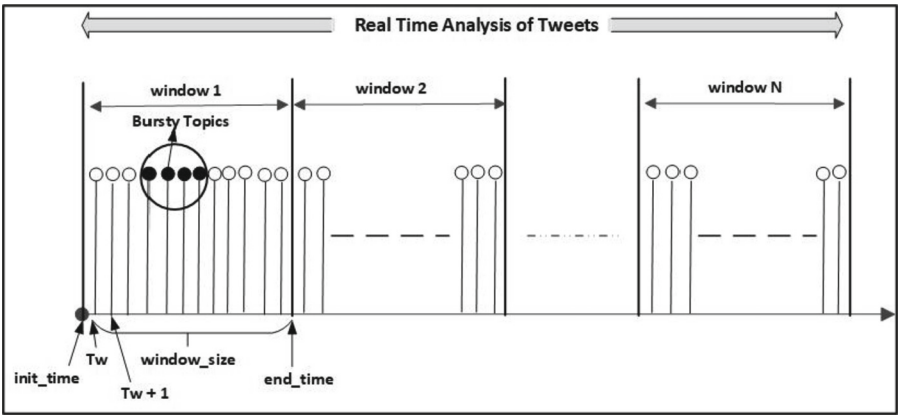

Fig. 2. Time Windowing System of the Proposed Algorithm

Table 5. A dictionary *data structure Hash_Dict* is initialized to store records of *most_frequent_words* has occurred in two consecutive windows- *windows1* and *window2*. Further, it is checked for these words whether the probability of occurrence of the words are increasing across the two consecutive windows. If it is increasing, those words are considered to be eligible for bursty keywords for

Table 4. Algorithm for Generating Temporal Bag of Most Frequent Keywords

Algorithm 3: Generating Temporal Bag of Most Frequent Keywords

Input: *Global Queue (G_Queue1) of pre-processed tweets*
Output: *Global Queue (G_Queue2) of most frequent keyword bags*

```
1.  init_time ← 0
2.  while tw ← get_tweet(G_Queue1) do
3.      if init_time == 0 then
4.          init_time ← timestamp(tw)
5.          end_time ← init_time + window_size
6.          tweet_bag ← null
7.          window_init_time ← init_time
8.          window_num ← 0
9.          total_win_words ← 0
10.     end if
11.     if timestamp(tw) < end_time then
12.         token_tw← get_tokenized_tweet(tw)
13.         tw_len ← length(token_tw)
14.         total_win_words ← total_win_words + tw_len
15.         Insert token_tw into tweet_bag
16.     else
17.         most_frequent_words ← Find unique words and Calculate its frequencies  using tweet_bag.
18.         most_frequent_words_desc ← Sort the most_frequent_words in decreasing order of its fre-
    quencies.
19.         most_frequent_words_cutoff← Store the top 20 % of  words in  most_frequent_words_desc.
20.         Push [window_num, window_init_time, total_win_words,
                most_frequent_words_cutoff ] list to G_Queue2
21.         window_num ← window_num + 1
22.         tweet_bag ← null
23.         total_win_words ← 0
24.         window_init_time ← end_time
25.         init_time ← end_time
26.         end_time ← init_time + window_size
27.     end if
28.  end while
```

the previous window. All these eligible bursty keywords are sorted as per their decreasing probability and frequency window-wise, and stored in *sorted_word_rank* and *sorted_word_freq* respectively. In order to get meaningful topics, a threshold value of average probability/frequency is calculated. All the bursty keywords having probability greater than equal to average probability and frequency greater than equal to average frequency are stored in *sorted_ word_rank_avg_cutoff* and *sorted_word_freq_avg_cutoff* respectively. Finally, *k* topics are created from the list of bursty keywords per window in *sorted_ word_freq_avg_cutoff* where *k* is the user input greater than zero for the number of topics. In order to get meaningful topics, the value assigned to k should be optimal. Based on the coherence score the optimal number of topics can be calculated. We have used *UMass coherence score for determining the optimal value for number of topics per datasets.* All the generated optimal *k* topics per window-wise is stored in *all_optimal_k_topics_per_window* for further processing in the next module.

Table 5. Algorithm for Bursty Keywords Detection & Optimal k-Topics per Window

Algorithm 4: Bursty Keywords Detection & Optimal k-Topics per Window

Input: *Global Queue (G_Queue2) of most frequent bags of keywords*
Output: *Global Data Frame of optimal k topics per window (all_optimal_k_topics_per_window)*

```
all_optimal_k_topics_per_window← []
for i in range length (G_Queue2) do
    most_frequent_words ←get_most_frequent_words( G_Queue2[i])
    for ∀word ∈ most_frequent_words do
      if word ∉ Hash_Dict then
          window_init_time ← get_window_init_time(G_Queue2[i])
          freq ← get_word_freq(word) from most_frequent_words
          total_win_words ← get_total_win_words(G_Queue2[i])
          window_num ← get_window_num(G_Queue2[i])
          window1← [window_init_time, freq, total_win_words]
          window2 ← []
          flag ← 0
          Push {word: [window1, window2, flag, window_num]}in Hash_Dict
      else if flag == 0 and (word present in the next consecutive window) then
            window2 ← get_window1(Hash_Dict[word])
            flag ← 1
            window_init_time ← get_window_init_time(G_Queue2[i])
            freq ← get_word_freq(word) from most_frequent_words
            total_win_words ← get_total_win_words(G_Queue2[i])
            window1← [window_init_time, freq, total_win_words]
            Hash_Dict[word] ← [window1, window2, flag, window_num]
          end if
      end if
    end for
    #Finding the Bursty keywords for each window
    if i >0 then
      for ∀ word in Hash_Dict do
          if flag == 1 then
            prob1 ←  Calculate probability of word in window1
            prob2 ←  Calculate probability of word in window2
            if prob1 – prob2 > 0 then  # if words probability is increasing
              word_rank[word] ← prob1
              word_freq [word] ← get_word_freq(word) from from Hash_Dict
            end if
          end if
      end for
      sorted_word_rank← Sort  word_rank[∀word] in decreasing order of probability
      sorted_word_freq← Sort  word_freq[∀word] in decreasing order of frequency

      #Threshold- bursty keywords having probability/frequency more than its average probability/frequency
      avg_rank ← Average  of sorted_word_rank[∀word]
      avg_freq ← Average  of sorted_word_freq[∀word]
      sorted_ word_rank_avg_cutoff ← sorted_word_rank[∀word] ≥ avg_rank
      sorted_word_freq_avg_cutoff ← sorted_word_freq[∀word] ≥ avg_freq
      Create Word Cloud  using sorted_word_freq_avg_cutoff
      k ← Optimal number of topics by using UMass Coherence score
      optimal_k_topics ← Create k topics using sorted_word_freq_avg_cutoff  #Apply K-Means
      Insert [i, optimal_k_topics] into all_optimal_k_topics_per_window
    end  if  # line 26
end for # line 2
```

v) ***Generating Topic's Coherence Scores per Window:*** Optimal Topics generated per window is fed as an input to this module as shown in Table 6. Coherence scores of the topics is measured by using the coherence frameworks- *UMass*, *UCI*, *NPMI*, C_V and *word2vec*.

Table 6. Algorithm for Generating Topic's Coherence Scores per Window

Algorithm 5: Generating Topic's Coherence Scores per Window

Input: *Global Data Frame of optimal k topics per window (all_optimal_k_topics_per_window)*

Output: *Topic's Coherent Scores per Window*

```
# Build a tweet_dictionary from the tokenized tweets in G_Queue2 using Dictionary()
# Build a tweet_corpus in a doc2bow() format of the tweet_dictionary
# Every coherence measure will take dictionary and corpus as input
#Fine coherence measures used: UCI Coherence, UMass Coherence, NPMI Coherence, Cv
 Coherence and word2vec Coherence

1.  coherence_measures ← {C_UCI, C_UMASS, C_NPMI, C_v, C_w2v }
2.  tweet_dictionary ← Dictionary(get_tokenized_tweet(G_Queue2))
3.  tweet_corpus ← [tweet_dictionary.doc2bow( ∀get_tweet(G_Queue2))]
4.  for i in range length(all_k_topics_per_window) do
5.      tweets_coherence ← []
6.      topics_in_window ← all_k_topics_per_window [i]
7.      topics ← []
8.      for t in range(k) do
9.          topics ← topics_in_window[t]
10.     end for
11.    topics_coherence_score ← CoherenceModel (topics, tweets_corpus, tweets_dictionary, C_UCI)
12. end for
13. Repeat line 4 to line 12 for each coherence_measures
```

Important Aspects of the Framework:

The framework designed is suitable under different real world high impact events like natural disasters, public opinion events or any emerging trends.

- A dynamic threshold determination is utilized which incorporates variability in the model, making it more suitable for the real-world scenario.
- Tweet and word vocabulary per window is not static but dynamically obtained according to the size of the window.
- Optimal k number of topics can be obtained per window using coherence score as per user choice of coherence measures.
- New module, for generation of Coherence Score of Topics, which helps to identify bursty topics of similar context and believed to be highly significant during impactful events.
- The designed approach is implemented on Twitter microblogs. But can be applied universally on any short text messages.

4 Implementation, Results and Analysis

Experimental Set-Up: Anaconda Jupyter Notebook and Google's Colab Pro environment was used as a platform for the study. The PC Configuration used was 4-core Intel Core i7 processor and 16 GB memory. Python version 3.8 was used as a programming language to implement the models. A Python library Twarc which is also a command line tool was used for archiving Twitter JSON data. The same was also used for rehydrating the dehydrated data sets which consist of only the list of tweet ids.

Dataset Description: We have used three natural disaster dataset collected from Kaggle Repository. All these repositories were released through a study by [27] where the data was collected by the author through specific keyword query search. The following datasets were selected with respect to volume of tweets, user engagement, retweet count showing the virility of the event. A brief snapshot on the datasets is elaborated in the Table 7.

The proposed algorithm is run across the three datasets and results are obtained. A baseline comparison is done with the Latent Dirichlet Allocation (LDA) Model [28], Gibbs Sampling Dirichlet Mixture Model (GSDMM) [29] and Gamma-Poisson Mixture Topic Model (GPM) [30] to show the perspectives in which the proposed model outperforms the baseline models. The latter two algorithms are proven to be good for short texts topic modelling.

Table 7. Dataset Features

Dataset (Duration)	Data Source	Number of Tweets	Number of Tweets Post Duplicate Removal	Keywords Used for Collection
Hurricane_Harvey (August 18 – 26, 2017)	Twitter	627557	**424782**	'Harvey', 'hurricaneharvey'
Typhoon_Hagupit (December 5 – 11, 2014)	Twitter	104172	**33710**	'typhoon', 'hagupit'
Hurricane_Sandy (October 25 – 28, 2012)	Twitter	568186	**139476**	'hurricane', 'sandy'

4.1 Proposed Algorithm Implementation

The implementation of the proposed algorithm is carried out for all the mentioned datasets post some analysis of the datasets required for the implementation. Table 8 summarizes the corresponding variable values found from all the three datasets post analysis of the datasets.

- *Window Size:* For Hurricane Harvey, variations in topics was much better for window size at 24 h or 86400 s as compared to window size at 6 h or 12 h. The burst in data happened only after the 5^{th} day of the incident. So, it was pointless to go by lesser than 24 h for these 5 days as the incoming stream of tweets is very less. So, window size is taken as 24 h or 86400 s. In case of Typhoon Hagupit, the window size is considered as 12 h or 43200 secs. The variability in topics is better here for this window size as compared to lesser or more than 12 h. Also, this dataset shows a good burst in incoming tweets from the very beginning, so expected dynamic topics to be present at every 12 h of time window. Hurricane Sandy is a 3-day dataset with a burst of tweets within a very short period of time. Owing to the lesser number of days and huge tweets streaming in, the window-size is kept 12 h.

Table 8. Important Findings from the Dataset

Parameters	Hurricane Harvey Values	Typhoon Hagupit Values	Hurricane Sandy Values
Total Number of Tweets	424782	33710	139476
Unique Words	68948	19354	204746
Window Size (in seconds)	86400	43200	43200
Number of Windows	6	11	4
Number of Optimal Topics found per Window	3	5	3

- *Number of Windows:* The detection of bursty keywords was considered comparing two consecutive windows. So, to determine a set of bursty keywords for a current window, the current and the next immediate window is considered. So, the total number of windows for which bursty keywords is detected is calculated as 6, 11 and 4 respectively for Harvey, Typhoon and Sandy datasets, which is one less than the actual number of windows in the main data frame as in Table 8.
- *Optimal Number of Topics:* While deciding on the number of topics for the events of the three datasets, overall coherence scores were calculated using *UMass Coherence measure* and plotted with varying number of topics per window. The aim is to choose the number of topics for which the coherence score is optimized. For most of the windows, the coherence measure is stable at number of topics as 3, 5 and 3 for Harvey,

Typhoon and Sandy datasets as in Table 8. Ideally, Hurricane Sandy could have been the best dataset with respect to the burstiness of data. But, actual implementation of the proposed algorithm on this dataset showed worst performance with respect to the topics generated with no variation at all. At the same time, as the topics are same across all the windows, there is no change in the coherence score with the change in topics. This clearly shows a disparity in the distribution of frequencies across the unique words across all the time windows.

4.2 Evaluation Results and Analysis

The list of coherence or confirmation measures [31] considered in this research to evaluate the bursty topics generated through the proposed and the baseline models are: UCI Coherence (C_{UCI}), UMass Coherence (C_{UMASS}), NPMI Coherence(C_{NPMI}), CV Coherence(C_V) and Word2vec (C_{W2V}). Following the coherence framework, the aggregated score of the measures is obtained by calculating the arithmetic mean of all the coherence or confirmation scores. The performance of the proposed model is compared with three baseline models in this research based on these scores. During the process of evaluation, we experimented different settings of parameter to achieve the best result possible. The sliding window sizes were tweaked in the range of (10, 150), and the context window was varied between (10, 100) for both the proposed algorithm and the LDA model. For GSDMM and GPM, the tweaking was done with the number of iterations (iters), top words of the cluster (nTopWords) considered and the size of the document (N). The number of topics (K) in all the models for every dataset were determined with respect to the average coherence score.

For influence of hyper parameters, the dirichlet priors and the gamma priors' values were tweaked for both GSDMM and GPM. For GSDMM, the dirichlet priors a and b are tried for a = 0.01,0.25,0.5,0.75,0.05 and b = 0.1,0.5,1.0,2.5 respectively. Finally, with respect to quality of the topics getting created for each of these, we settle on a = 0.25 and b = 0.15. Similar things were repeated for GPM model for the gamma and the dirichlet priors. The evaluation results of coherence scores obtained by implementing all the models, including the proposed algorithm are depicted in the following tables for respective datasets. The highlighted rows in bold are measures where our model has outperformed as compared to the baselines.

Table 9. Proposed Bursty Model Evaluation in Hurricane Harvey

		Window Number					
Topic Model	Coherence Measures	1	2	3	4	5	6
Proposed Burst Detection Model	**C_{UMASS}**	-0.0789	−0.071	−0.062	−0.014	−0.873	−0.596

(*continued*)

Table 9. (*continued*)

		Window Number					
Topic Model	Coherence Measures	1	2	3	4	5	6
	C_{UCI}	−0.770	−0.801	−0.098	−0.715	−0.323	−0.157
	C_{NPMI}	−0.051	0.011	0.019	−0.024	−0.032	−0.012
	Cv	**0.381**	**0.476**	**0.427**	**0.336**	**0.323**	**0.362**
	C_{W2V}	**0.996**	**0.995**	**0.995**	**0.996**	**0.998**	**0.998**
	Aggregate Score (AM)	−0.104	0.122	0.256	0.115	0.175	0.226
LDA	**C_{UMASS}**	−0.136	−0.181	−0.137	−0.147	−0.169	−0.153
	C_{UCI}	0.037	0.043	0.029	0.025	0.042	0.027
	C_{NPMI}	0.014	0.015	0.017	0.016	0.015	0.015
	Cv	0.261	0.290	0.279	0.285	0.281	0.256
	C_{W2V}	0.989	0.988	0.989	0.990	0.989	0.989
	Aggregate Score (AM)	0.233	0.231	0.235	0.234	0.231	0.227
GSDMM	**Average Coherence Score**	−20.133	−24.623	−25.688	−27.456	−30.814	−32.161
GPM	**Average Coherence Score**	−27.023	−23.595	−23.229	−27.024	−27.464	−31.488

- Coherence evaluation measures for Hurricane Harvey and the baseline comparison is shown in Table 9. The proposed model has generated competitive scores in case of C_V and Word2vec.
- Coherence evaluation measures for Typhoon Hagupit and the baseline comparison using the coherence measures in shown in Table 10. For this dataset, the C_V score measure is better for the proposed model as compared to the other three models.
- Coherence evaluation measures for Hurricane Sandy and the baseline comparison is shown in Table 11. The proposed model has resulted in better results for all the coherence measure as compared to LDA, GSDMM AND GPM for this dataset.

Table 10. Proposed Bursty Model Evaluation in Typhoon Hagupit

		Window Number										
Topic Model	**Coherence Measure**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Proposed Burst Detection Model	C_{UMASS}	-0.042	-0.034	-0.036	-0.056	-0.041	-0.065	-0.038	-0.037	-0.027	-0.063	-0.051
	C_{UCI}	-0.123	-0.149	-0.151	-0.166	-0.159	-0.156	-0.162	-0.155	-0.196	-0.282	-0.171
	C_{NPMI}	-0.014	-0.019	-0.019	-0.020	-0.019	-0.019	-0.020	-0.020	-0.027	-0.027	-0.023
	C_V	**0.356**	**0.345**	**0.345**	**0.342**	**0.352**	**0.351**	**0.344**	**0.344**	**0.336**	**0.336**	**0.342**
	C_{W2V}	0.805	0.812	0.812	0.814	0.803	0.807	0.812	0.812	0.819	0.811	0.814
	Aggregate Score (AM)	0.196	0.190	0.190	0.182	0.187	0.183	0.187	0.188	0.180	0.154	0.181
LDA	C_{UMASS}	-0.015	-0.019	-0.014	-0.018	-0.026	-0.018	-0.018	-0.018	-0.025	-0.013	-0.022
	C_{UCI}	0.030	0.041	0.025	0.037	0.029	0.043	0.038	0.017	0.062	0.038	0.029
	C_{NPMI}	0.009	0.006	0.013	0.021	0.012	0.015	0.010	0.013	0.017	0.007	0.009
	C_V	0.255	0.261	0.269	0.271	0.280	0.282	0.255	0.289	0.263	0.267	0.278
	C_{W2V}	0.833	0.836	0.837	0.835	0.834	0.837	0.836	0.833	0.832	0.831	0.830
	Aggregate Score (AM)	0.222	0.225	0.226	0.229	0.225	0.232	0.224	0.227	0.230	0.226	0.224
GSDMM	**Average Coherence Score**	-22.081	-23.860	-24.548	-28.025	-25.105	-25.651	-26.625	-27.496	-23.863	-22.822	-23.596
GPM	**Average Coherence Score**	-27.110	-26.913	-24.337	-29.851	-26.730	-27.276	-23.569	-25.159	-25.707	-21.602	-23.683

Table 11. Proposed Bursty Model Evaluation in Hurricane Sandy

		Window Number			
Topic Model	Coherence Measure	1	2	3	4
Proposed Burst Detection Model	$\mathbf{C_{UMASS}}$	**.0001**	**−0.028**	**−0.029**	**−0.028**
	$\mathbf{C_{UCI}}$	**0.024**	**0.003**	**0.003**	**0.003**
	$\mathbf{C_{NPMI}}$	**0.014**	**0.012**	**0.012**	**0.012**
	CV	**0.356**	**0.361**	**0.361**	**0.361**
	$\mathbf{C_{W2V}}$	**0.998**	**0.998**	**0.998**	**0.998**
	Aggregate Score (AM)	**0.278**	**0.269**	**0.269**	**0.269**
LDA	$\mathbf{C_{UMASS}}$	0.0001	0.0001	0.0001	0.0001
	$\mathbf{C_{UCI}}$	0.004	0.006	0.006	0.007
	$\mathbf{C_{NPMI}}$	0.003	0.005	0.004	0.005
	CV	0.236	0.237	0.233	0.239
	$\mathbf{C_{W2V}}$	0.979	0.980	0.984	0.984
	Aggregate Score (AM)	0.244	0.246	0.245	0.247

(*continued*)

Table 11. (*continued*)

		Window Number			
Topic Model	Coherence Measure	1	2	3	4
GSDMM	**Average Coherence Score**	−37.781	−40.937	−44.726	−41.445
GPM	**Average Coherence Score**	−38.436	−38.776	−45.458	−42.886

5 Discussions

The coherence score measures the quality of the topics getting generated per window. According to [31], higher or closer the coherence score towards '1', more coherent the topics are. Also, the range of UMass coherence is −14 to + 14, UCI and NPMI Coherence is between −1 to + 1, for CV and Cw2v both are between 0 and 1. C_V is proven to be the best measure in baseline paper [31]. This is a combination measure, found by combining indirect cosine confirmation measure with NPMI and the concept of Boolean sliding window. C_V and C_{w2v} which are semantic and contextual measures of the topics, have given the best scores across all the datasets. For all the datasets, in general the NPMI coherence measure has given better coherence values than non-normalized UCI coherence version of it. Overall, C_{w2v} measure has performed well as compared to the other measures. The fact can be for the length of the input text. In the baseline paper [31], the goodness of the coherence measures was proved with long texts or articles. So, the scores the proposed algorithm achieved is proved to be competitive. In this case, we are trying the apply coherence measures for short texts. This shows the direction towards an improvement to the algorithm required which will take the length of the document also into consideration, and is an immediate future work. Apart from that in all the datasets, better performance of our model as compared to the other with respect to C_V and C_{w2v} is a contribution of this study. As both these measures signifies the semantic and contextual features of topics, our model is successful in creating better semantically coherent and contextual topics as compared to the other state of the art techniques available in the field of topic modelling.

Practical Implications of this Research: The proposed model can be used for modelling topics for any event based on Twitter. Additionally, the researchers can also measure the goodness of the topics through coherence measures, inferencing on the coherent topics at different time window across the events. This information can be further leveraged to understand the trends per time window. In case of disaster events or any high impact events, knowledge on coherent topics per window can facilitate in making several decisions in support for the disaster at that point of time.

6 Conclusion

This paper detailed the complete work regarding the proposed burst model of a high impact event. The proposed algorithm detects bursty optimal topics during high impact events comparing the bursty words across consecutive time windows. The algorithm further measures the coherence scores of the bursty optimal topics window-wise using a coherence framework. The coherence scores of the topics generated from the proposed algorithm is compared with the state-of-the-art baseline topic modelling techniques. Through proper experimentation and analysis, our proposed model is successful in creating better topics than the baseline models with respect to the contextual coherence features.

References

1. Comito, C., Forestiero, A., Pizzuti, C.: Bursty event detection in Twitter streams. ACM Trans. Knowl. Disc. Data (TKDD) **13**(4), 1–28 (2019)
2. Imran, M., Castillo, C., Diaz, F., Vieweg, S.: Processing social media messages in mass emergency: a survey. ACM Comput. Surv. (CSUR) **47**(4), 1–38 (2015)
3. Fedoryszak, M., Frederick, B., Rajaram, V., Zhong, C.: Real-time event detection on social data streams. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2774—2782 (2019)
4. Lee, P., Lakshmanan, L.V., Milios, E.E.: Incremental cluster evolution tracking from highly dynamic network data. In: 2014 IEEE 30th International Conference on Data Engineering, pp. 3–14. IEEE (2014)
5. Singh, T., Kumari, M.: Burst: real-time events burst detection in social text stream. J. Supercomput. **77**, 1–29 (2021)
6. Mathioudakis, M., Koudas, N.: Twittermonitor: trend detection over the twitter stream. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 1155–1158 (2010)
7. Guzman, J., Poblete, B.: On-line relevant anomaly detection in the Twitter stream: an efficient bursty keyword detection model. In: Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, pp. 31–39 (2013)
8. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proceedings of the 19th International Conference on World Wide Web, pp. 851–860 (2010)
9. Zhao, W.X., Chen, R., Fan, K., Yan, H., Li, X.: A novel burst-based text representation model for scalable event detection. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 43–47 (2012)
10. Rezaei, Z., Eslami, B., Amini, M.A., Eslami, M.: Event detection in Twitter by deep learning classification and multi label clustering virtual backbone formation. Evol. Intell. **16**(3), 833–847 (2023)
11. Singh, J., Pandey, D., Singh, A.K.: Event detection from real-time twitter streaming data using community detection algorithm. Multimed. Tools Appl., 1–28 (2023)
12. Yang, J., Wu, Y.: An approach of bursty event detection in social networks based on topological features. Appl. Intell., 1–19 (2022)
13. Kleinberg, J.: Bursty and hierarchical structure in streams. Data Min. Knowl. Disc. **7**(4), 373–397 (2003)
14. Weng, J., Lee, B.S.: Event detection in Twitter. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 5, no. 1 (2011)

15. Naaman, M., Becker, H., Gravano, L.: Hip and trendy: characterizing emerging trends on Twitter. J. Am. Soc. Inform. Sci. Technol. **62**(5), 902–918 (2011)
16. Li, C., Sun, A., Datta, A.: Twevent: segment-based event detection from tweets. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 155–164 (2012)
17. Xie, W., Zhu, F., Jiang, J., Lim, E.P., Wang, K.: Topicsketch: real-time bursty topic detection from Twitter. IEEE Trans. Knowl. Data Eng. **28**(8), 2216–2229 (2016)
18. Becker, H., Naaman, M., Gravano, L.: Beyond trending topics: real-world event identification on Twitter. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 5, no. 1 (2011)
19. Osborne, M., et al.: Real-time detection, tracking, and monitoring of automatically discovered events in social media. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 37–42 (2014)
20. Hasan, M., Orgun, M.A., Schwitter, R.: TwitterNews: real time event detection from the Twitter data stream. PeerJ PrePrints **4**, e2297v1 (2016)
21. Li, J., Tai, Z., Zhang, R., Yu, W., Liu, L.: Online bursty event detection from microblog. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 865–870. IEEE (2014)
22. Zhang, C., et al.: Geoburst: real-time local event detection in geo-tagged tweet streams. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 513–522 (2016)
23. Bhuvaneswari, A., Valliyammai, C.: Identifying event bursts using log-normal distribution of tweet arrival rate in twitter stream. In: 2018 Tenth International Conference on Advanced Computing (ICoAC), pp. 339–343. IEEE (2018)
24. Ban, A., Zhang, Z., Gao, D., Zhou, Y., Gupta, B.B.: A novel burst event detection model based on cross social media influence (2022)
25. Sharma, S., Abulaish, M., Ahmad, T.: KEvent–A semantic-enriched graph-based approach capitalizing bursty keyphrases for event detection in OSN. In: 2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), pp. 588–595. IEEE (2022)
26. Mimno, D., Wallach, H., Talley, E., Leenders, M., McCallum, A.: Optimizing semantic coherence in topic models. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language (2011)
27. Zubiaga, A.: A longitudinal assessment of the persistence of Twitter datasets. J. Am. Soc. Inf. Sci. **69**(8), 974–984 (2018)
28. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J, Mach. Learn. Res. **3**, 993–1022 (2003)
29. Yin, J., Wang, J.: A dirichlet multinomial mixture model-based approach for short text clustering. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 233–242 (2014)
30. Mazarura, J., De Waal, A., de Villiers, P.: A gamma-poisson mixture topic model for short text. Math. Prob. Eng. 2020 (2020)
31. Röder, M., Both, A., Hinneburg, A.: Exploring the space of topic coherence measures. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, pp. 399–408 (2015)