Bursty Event Detection in Twitter Streams

CARMELA COMITO, AGOSTINO FORESTIERO, and CLARA PIZZUTI, National Research Council of Italy (CNR), Institute for High Performance Computing and Networking (ICAR)

Social media, in recent years, have become an invaluable source of information for both public and private organizations to enhance the comprehension of people interests and the onset of new events. Twitter, especially, allows a fast spread of news and events happening real time that can contribute to situation awareness during emergency situations, but also to understand trending topics of a period. The article proposes an online algorithm that incrementally groups tweet streams into clusters. The approach summarizes the examined tweets into the cluster centroid by maintaining a number of textual and temporal features that allow the method to effectively discover groups of interest on particular themes. Experiments on messages posted by users addressing different issues, and a comparison with state-of-the-art approaches show that the method is capable to detect discussions regarding topics of interest, but also to distinguish bursty events revealed by a sudden spreading of attention on messages published by users.

CCS Concepts: • **Information systems → Clustering**; **Data stream mining**; *Online analytical processing*; *Blogs*;

Additional Key Words and Phrases: Twitter, online clustering, event detection, bursty event

ACM Reference format:
Carmela Comito, Agostino Forestiero, and Clara Pizzuti. 2019. Bursty Event Detection in Twitter Streams. *ACM Trans. Knowl. Discov. Data* 13, 4, Article 41 (August 2019), 28 pages.
https://doi.org/10.1145/3332185

1 INTRODUCTION

The fast expansion of social media in the last years is making available an enormous and continuous stream of user-generated contents containing invaluable information that can be used to understand, in near real time, what is happening in the world. Twitter is a social media microblogging system that is gaining increased popularity because it allows users to broadcast messages that can rapidly be reached by millions of people. It constitutes an important source of communication and information diffusion. Every day users post personal opinions regarding discussion topics of interest and report information on events happening in real time. Analyzing this data can be fundamental for discovering the arise of events, such as cultural exhibition, sport, rallies, but also emergency situations due to natural disasters, such as flooding, hurricane, or earthquake (Imran et al. 2015). By leveraging the knowledge derived from these events, it is possible to understand

Authors' addresses: C. Comito, A. Forestiero, and C. Pizzuti, National Research Council of Italy (CNR), Institute for High Performance Computing and Networking (ICAR), Via P. Bucci 8-9 C, 87036 Rende (CS), Italy; emails: {carmela.comito, agostino.forestiero, clara.pizzuti}@icar.cnr.it.

1556-4681/2019/08-ART41 $15.00
https://doi.org/10.1145/3332185

what people are interested in a time period and what they are doing, by giving the opportunity to organizations to make informed decisions.

Xie et al. (2007) define events as "real-world occurrences that unfold time and space" that can be described with the so-called *5W1H*, six interrogatives *who? when? where? what? why?* and *how?* i.e., metadata information expressing their characteristics. The automatic detection of unknown events in streams of news stories has been faced since the nineties in the Topic Detection and Tracking research program of Allan et al. (Allan et al. 1998; Allan 2002), and by Yang et al. (1998). As discussed in (Allan et al. 1998; Yang et al. 1998), the detection of new patterns can be obtained by employing a clustering method that aggregates similar data in the same group. The detection can be either retrospective, in this case the aim is to find all the events contained in a corpus of stories, or on-line, that is, the stream of documents is processed sequentially, and a new document is assigned to the most similar cluster already generated, if the similarity is above a fixed threshold, otherwise it is considered a new cluster. Each document is represented through the vector space model (Salton and McGill 1986) with the *bag-of-terms* representation, which is a vector of words weighted using the *Term Frequency (TF)* and the *Inverse Document Frequency (IDF)* appropriately normalized.

This basic online single pass approach to cluster documents has been recently adopted also for event detection (Becker et al. 2011; Yin et al. 2012; Alsaedi et al. 2017) from Twitter streams.

In this article, our aim is to analyze Twitter streams to understand what people are interested in a time period, and to detect interesting events, eventually related to emergent situations. This twofold objective is obtained by an online and incremental clustering method, named *BEaTS* (Bursty Event detection in Twitter Streams), able to discover not only trending topics, but also to detect bursty events generated by an unusually high number of messages on the same argument in a short time period.

The method is able to effectively face two main challenges typical of classical approaches for clustering data streams continuously generated over time (Aggarwal and Yu 2010; Silva et al. 2013). The first is that tweets can be examined only once. The other is that, since user's interests regarding particular topics change with time, some clusters can become obsolete and should be discarded if no new tweet is added after a time period. In order to deal with the former problem, each tweet is examined one at a time and its main features extracted. The information gathered by all the similar tweets, that is those dealing with a common subject, is then summarized into the centroid of the cluster they have been assigned to. Each centroid is characterized by a number of features, such as the timestamps of the cluster creation and that of the last update, along with the set of terms appearing in the tweets examined so far, belonging to that cluster, and their frequencies. The main contributions of the article can be summarized as follows:

— a new representation for tweets has been defined storing both textual information through unigram and bigram words/hashtags/mentions, as well temporal and geographical information;
— the definition of centroid we introduce allows to gather statistics useful for the detection of topics of interest in a time period, as well the sudden spreading of an unusual event; differently from approaches using the *tf-idf* weight vector representation, where the most frequent terms must be chosen in advance, the terms stored in the centroid are incrementally added to it every time a new tweet is assigned to that centroid. Thus, which terms maintain must not be fixed a priory. The new terms continue to be included in a centroid, but there is no problem of too many terms, because many clusters become inactive after a time period; thus, our space requirements do not increase forever. The definition of centroid takes into account the temporal evolution of term frequencies to allow the detection

of bursty events. In such a way our approach becomes a general method to discover both topics of interest and bursty topics;

— the concept of time-fading model for data streams has been exploited to define a new fading function that takes into account the characteristics of the tweets and allows the introduction of the concept of *active/inactive* cluster. The function combines the frequency of tweet streams with the concept of interesting emerging topic by discarding (i.e., becoming inactive) clusters of tweets that do not grow in size for a time period, while retaining those receiving new tweets;
— a similarity measure that takes into account the importance of terms according to their frequency, weighted for accounting historical data, is defined. Moreover, the use of the fading function also in the similarity measure allows the method to effectively consider more important the most recent tweets;
— the concept of time window is exploited to periodically check if active clusters become bursty, i.e., they present an unusual size growth in a short time period;
— a thorough experimentation on datasets of tweets addressing different issues, and a comparison with state-of-the-art approaches show that the method is capable to detect discussions regarding topics of interest, but also to distinguish bursty events revealed by a sudden spreading of attention on messages published by users.

The article is organized as follows. In the next section, an overview of the most recent proposals for discovering events in social data is given. Section 3 introduces the concepts used to model the clustering problem of tweet streams. Section 4 describes the online algorithm and defines the similarity measure between a tweet and a centroid. Section 5 reports the experimental results obtained by executing *BEaTS* and the contestant methods on the considered datasets. Finally, Section 6 concludes the article and discusses future developments.

2 RELATED WORK

In the last years, there has been a significant research effort on detecting topics and events in social media. Several surveys describing state-of-the-art event detection techniques have been presented (Atefeh and Khreich 2015; Cordeiro and Gama 2016; Goswami and Kumar 2016; Weiler et al. 2017; Hasan et al. 2018). Techniques defined for traditional media, and often adapted for Twitter data, as reported in (Atefeh and Khreich 2015), are classified into *document-pivot* and *feature-pivot* techniques. In the former approach, documents showing textual similarity are clustered. In the latter, clustering is performed on relevant keywords extracted from the documents. Representative methods in these two classes, often used as baseline approaches to compare with to assess the performances of methods are *Latent Dirichlet Allocation (LDA)* (Blei et al. 2003) and *Document-Pivot Topic Detection (Doc-p)* (Petrović et al. 2010), respectively.

LDA is a topic model that relates words and documents through latent topics. It associates with each document a probability distribution over topics, which are distributions over words. A document is represented with a set of terms, which constitute the observed variables of the model. The topic and term distributions have to be estimated by using Bayesian Inference. One of the main drawback of *LDA* is that it requires the expected number of topics as input parameter. An extension to *LDA* for a generative biterm topic model to enhance the topic learning has been proposed by Yan et al. (2013). The approach models the word co-occurrence patterns instead of the documents. Wang et al. (2012) consider the temporal information of text streams, such as sequences of messages posted by the same author, and model the topic transitions of such streams. However, the aim of this approach, fixed the number of topics, is to predict the topic distribution in future posts, not clustering consecutive posts.

Doc-p is an on-line clustering method that for each document considers the list of words, and computes the cosine similarity of the *tf-idf* (Salton and McGill 1986) representation of a new text with respect to all those already examined. If the best similarity is above a threshold θ, then the story is assigned to the cluster with the best match, otherwise a new cluster is generated.

As pointed out in (Atefeh and Khreich 2015), direct application of these techniques to Twitter streams may give low quality results; thus, many different methods have been proposed. Generally, the common approach is to extract different data features from social media streams and then summarize such features for event detection tasks by exploiting the information over content, temporal, and social dimensions. The different approaches are classified on the base of the task they accomplish, namely *retrospective event detection* (that is patterns in the data are identified after an event has occurred) vs. *new event detection* (i.e., a decision about a new or an old event has to be taken as documents are examined), *unspecified* (i.e., no prior information about the events is available) vs. *specified* event detection (i.e., the user is provided with some features known about the event).

With respect to this classification, our approach can be considered as an unspecified and new event detection approach. In the following, some of the most popular methods are described, paying particular attention to the approaches detecting events online and those focusing on bursty events. For a comprehensive overview, refer to the review papers cited above.

Burst-keyword-based methods. A relevant number of methods proposed in the literature discover events by first extracting bursty terms from text documents, such as news articles, or from social media, and then clustering such terms. Fung et al. (2005) proposed a parameter free probabilistic approach to find a minimal set of bursty features than can represent a bursty event. Their feature-pivot clustering approach groups the obtained bursty features and then identifies the hot periods of the bursty events. The probabilistic approach of Mei and Zhai (2005) discovers theme patterns from text streams by first generating word clusters for each time period and then using the Kullback–Leibler divergence measure to discover coherent themes over time. The method builds an evolutionary graph that shows how themes change over time. Wang et al. (2007) introduced another probabilistic algorithm to discover correlated bursty patterns and the periods in which they occur from multiple text streams, eventually having different language vocabulary. A bursty topic in a stream is defined as a probability distribution of words covered for a relatively long, but intense, consecutive period. When multiple text streams cover the same event in the same period they are said correlated. Note that the approach is retrospective, though it differs from (Allan et al. 1998; Allan 2002; Yang et al. 1998) since it considers simultaneously more streams. *TwitterMonitor* is a system (Mathioudakis and Koudas 2010) that discovers and analyzes trends over Twitter streams in real time. The system first finds bursty keywords and then groups them by considering their co-occurrences. A keyword is defined bursty if it appears in the stream of recent tweets at an unusually high rate. The authors do not formally explain neither what "unusually high rate" means, nor the length of the time period corresponding to their concept of "recent." *EDCoW* is a method proposed by Weng and Lee (2011) for event detection that computes the energy of words by using wavelet analysis on word frequency signals. The approach filters away trivial words and then clusters the remaining ones with a modularity-based graph partitioning technique. A method that finds temporal patterns by proposing a time series clustering method using a time series shape similarity metric has been presented by Yang and Leskovec (2011). The proposed K-Spectral Centroid (K-SC) clustering algorithm allows the efficient computation of cluster centroids under the defined distance metric. *Twevent* (Li et al. 2012) is an event detection system based on the concept of *tweet segment*. A tweet segment is defined as one or more consecutive words appearing in a tweet. Thus, the approach first segments the tweets, and then identifies bursty segments in a time window by considering both the frequency of segments in tweets and the number of users who post them.

A k-nearest neighbor clustering method is then applied to segments by using cosine similarity to provide the *candidate events*. Filtering is applied to these candidates and the top ranked events are returned. Zhao et al. (2012b) aimed to identify event-related bursts from correlated social media activities, such as post publication, click on shared photos, connections between users. Fixed a sequence of terms, an event-related burst is defined as a period in which the amount of activities related to these terms is higher than the average. To this end the authors introduced the concept of smoothness of a sequence of activities to discriminate noisy events. Zhao et al. (2012a) proposed a burst-based text representation model that enriches frequent terms with the time interval during which the burst occurs. To find bursty terms, the two-state automaton of (Kleinberg 2003) is used. Events are then identified by clustering the bursty features. Yao et al. (2012) aimed to detect bursty tagging events, that is small subsets of tags co-occurring on social media to annotate items of interest. The approach first builds a graph of correlated tags, and then extracts subgraphs to detect bursty tagging events. Lee et al. (2012) defined a measure for discriminating between bursty terms in a stream of documents by combining different scores related to the frequency of selected terms. Sakaki et al. (2013) proposed an approach to detect a target event identified by a set of terms, by considering only the relevant tweets, i.e., those containing the query word. Each tweet is represented by three features as follows: the number of words in the tweet and the position of the query word in it, the words appearing in the tweet, and the words before and after the query word. A target event is discovered by using a Support Vector Machine classifier (Joachims 1998) that categorizes each tweet as belonging to either the positive class, i.e., the query event to search for, or the negative class. The method has been applied to Japanese tweets reporting on earthquake events. It was able to detect an earthquake by monitoring tweets, faster than the official announcements. *EvenTweet* (Abdelhaq et al. 2013) is a framework that detects localized events described by bursty keywords observed during time frames, and appearing in geo-localized tweets in a particular location. These events are then extracted by clustering keywords on the base of their spacial similarity. *ET* (Parikh and Karlapalem 2013) is a system that discovers events from tweets by defining a time span for a set of tweets and then computing frequent keywords in these intervals. These representative keywords are clustered by using a hierarchical clustering method, and each cluster is interpreted as an event. Li et al. (2014) presented an incremental temporal topic model that first generates a set of events and the probability of each to be interesting at the time t_i, and then labels an event as bursty if the number of tweets between two time windows is doubled. Zhang and Qu (2015) obtained bursty events by first extracting bursty words within a temporal window and then applying a hierarchical clustering method. They use the *tf-idf* representation of words to compute the burst weight, which is based on the frequency of the term. Guille and Favre (2015) proposed a statistical method that relies not only on the textual content of a tweet, but also on the mentions they contain to exploit the interaction among users. The authors show that the method using the mentions outperforms a variant without mentions. Stilo and Velardi (2016) discretize temporal series of events to transform words into string of symbols, then, from a set of known events, a regular expression is learned to distinguish between event-like from non-event terms. Finally, a hierarchical clustering method is applied to the former terms to generate clusters of events. *TopicSketch* (Xie et al. 2016) detects bursty topics from Twitter by relying on the concept of word acceleration. This system first maintains as a sketch of the data the arriving rates, called *velocity*, of pairs and triples of words, then the acceleration, i.e., the change of velocity, between two time windows is computed. A monitor tracks the sketch and estimates potential bursty topics if the acceleration is above a fixed threshold. After that the system infers the actual bursty topics by exploiting tensor decomposition.

The main limitation of many of the above approaches is that they often need to fix the term vocabulary and focus on bursty keywords appearing in the streams, covering this way only events related to highly frequent words. Moreover, the methods of Zhao et al. (2012b) and Sakaki et al. (2013) detect only targeted events, since they requires a set of query words, thus they are unable to catch unknown topics. The method of Li et al. (2014) needs as input parameters the number of topics to discover and the the number of words of the vocabulary representing each document, that of Zhang and Qu (2015), besides the term vocabulary, must also set a threshold value to define a cluster as a bursty event cluster.

Differently, our approach is able to detect topics of discussion by grouping similar tweets, identifying in nearly real time the ones that become bursty.

Online event detection. Yin et al. (2012) presented an online clustering method for topic discovery, inspired by Yang et al. (1998). They represent the textual content of tweets by a traditional vector-space model, in which a tweet is a vector of words $(v_1, v_2, \ldots, v_d)$, where d is the size of word vocabulary and v_j is the weight of jth term in the tweet computed through the *tf-idf* concept. The authors use the cosine and Jaccard similarities to compute the closeness of a tweet to a centroid, multiplied by a time factor. Each new incoming tweet is assigned to the cluster whose centroid has minimum distance, then the centroid of a cluster C is updated as the normalized vector sum of all the tweets in it. Only tweets containing bursty words are considered for inclusion in a cluster. Bursty words are detected by comparing the probability that a word appears in a time window against the expected probability of occurring in a random window. Becker et al. (2011) proposed an online clustering approach coupled with a post-processing classification step that distinguishes between events and non-events. The non-events are defined by the authors as trending activities in Twitter, such as conversation topics, which do not represent any real-world occurrences. To identify real-world events, each message is represented as a *tf-idf* weight vector, where terms are all those appearing in all the documents. Since the number of terms may be very high, the authors consider the n most frequent terms, where n is experimentally fixed, to compute features that help in revealing the events. Our approach is more general since our concept of event includes not only real fact occurrences, but also discussion topics people are interested in a particular period, such as an exhibition or a TV show. Pohl et al. (2012) use also the *tf-idf* representation for discovering sub-events, i.e., an event that occurs at a specific time or location during an emergency case. The detection is performed by using a clustering approach based on a Self Organizing Map. Aiello et al. (2013) introduced two methods to detect trending topics: *Soft Frequent Pattern Mining (SFPM)*, and *BNGram*. *SFPM* is a soft version of the well known frequent pattern mining approach that finds frequent patterns in association rules (Agrawal and Srikant 1994) by taking into account the simultaneous co-occurrences between any number of terms. *BNgram* finds emerging topics by considering the co-occurrences of n-grams instead of unigrams, and comparing the frequencies of terms in the current time slot and the preceding ones. A classical group average hierarchical clustering method (Murtagh 1983) is then applied to group n-grams into clusters. Two clusters are joined if their similarity is above a fixed threshold.

Wang et al. (2015) proposed *Sumblr*, a framework for continuous summarization of tweet streams. The framework consists of three main components, namely the Tweet Stream Clustering module, the High-level Summarization module and the Timeline Generation module. Similar to *BEaTS*, *Sumblr* implements an online algorithm that clusters tweets with only one pass over the data. The algorithm employs two data structures. The first one is the tweet cluster vector (TCV) maintained dynamically in memory during stream processing. The second structure is the pyramidal time frame (PTF), which is used to store and organize cluster snapshots at different moments, thus allowing historical data to be retrieved. The high-level summarization module supports, through the TCV-Rank algorithm, the generation of online (using the clusters maintained in memory) and historical summaries (using historical cluster snapshots from the PTF). The core of the timeline generation module is a topic evolution detection algorithm, which exploits online/historical summaries to produce real-time/range timelines.

The main differences between *BEaTS* and the above approaches is in the summarization technique. In (Yin et al. 2012; Becker et al. 2011; Aiello et al. 2013; Wang et al. 2015), the textual content of tweets is represented with a traditional vector-space model, where the vector dimension is fixed in advance as the size of word vocabulary. In streaming scenarios, however, word vocabulary dynamically changes over time. Differently, we deal with content evolving signature structure of cluster centroids by either updating frequencies of already present terms, or including new terms; since we refer to term frequencies as relative values, there is no need of recalibrating the vocabulary size. Consequently, in *BEaTS* the terms stored in the centroid are incrementally added to it every time a new tweet is assigned to that centroid. Thus, which terms maintain must not be fixed a priory. Another distinguished feature of *BEaTS* is that it combines lexical similarity and temporal proximity as a criterion for clustering. Moreover, different from Yin et al. (2012), our approach does not restrict the analysis to only those tweets containing the words classified bursty. Finally, with respect to the work in (Aiello et al. 2013) we are able, like the works in (Yin et al. 2012; Wang et al. 2015), to detect topic evolution and burstiness, and the summaries are produced in an online fashion.

3 PROBLEM FORMULATION AND MODEL

In this section, the event detection problem is first formulated, then the necessary notations and definitions are introduced.

3.1 Problem Formulation

The problem tackled in this article is the design of a (near) real-time system able to collect and analyze streams of short texts generated by users on social media for the detection of unknown events. Events can be trending topics, i.e., discussion issues on a common argument of interest happening in a particular time period, such as, for instance, a sport match or a political debate, but also burtsy facts generated by the occurrence of an unusually high rate of posts on the same subject in a short time period, for example, a terroristic attack, or a natural disaster. Events are obtained by clustering together tweets dealing with the same theme, and those raised in a very short time period, are classified as bursty events. In the following of the article, we refer to event and topic terms interchangeably.

Since the length of the time period during which a new topic should be discovered depends on the application domain, we assume that the user of the system provides three types of information as follows: the life-span a topic is considered interesting, the time slot (called also window) after which the mined events are delivered, and the frequency of checking the arise of bursty events inside the time window. The output of the approach, after each elapsed time slot, is the list of events discovered in the current time window, with the distinction of those classified as bursts.

These assumptions make the system apt for different real-world scenarios. For instance, monitoring natural disasters, such as fire or flooding, of a particular area can be achieved by setting burst detection frequency every few minutes, while finding social and cultural events needs more time, such as hours, or even days.

It is worth to note that, though we used data coming from a specific social network, namely Twitter, the approach is applicable to any stream of short messages produced on social media.

3.2 Data Model

Clustering tweet streams, continuously generated over time, poses several challenges for the development of efficient methods. Analogously to classical data stream approaches (Aggarwal and Yu 2010; Silva et al. 2013), tweets can be examined only once, that is, social data must be processed as it is produced, in one pass, and cannot be maintained in memory. This implies that, because of scalability and performance issues, space-efficient data structures have to be designed to summarize and maintain the data analyzed so far. To obtain this objective, we leverage on the concept of cluster centroid to accumulate statistics relative to the tweets already examined and assigned to the same cluster.

We first introduce the concept of *social object*, to extract relevant textual features from a tweet.

Definition 3.1. A tweet tw posted by a user u at time t from a location l, can be represented as a tuple, denominated *social object*, $so = (o, u, t, l, sgn)$, where o is the object identifier and $sgn = (w_u, w_b, h_u, h_b, m_u, m_b)$ is a feature vector, called *signature*. Each feature is a list of items defined as follows:

— w_u are the words appearing in the tweet;
— w_b are the word bigrams;
— h_u are the hashtags appearing in tw;
— h_b are hashtag bigrams;
— m_u are the mentions contained in the tweet;
— m_b are mention bigrams.

The location l is expressed in terms of the geographic coordinates (latitude and longitude). The signature sgn of a tweet tw is a compact representation of its content by extracting words, hashtags, and mentions from the text, along with the item bigrams, i.e., the adjacent item pairs appearing in the tweet. An example of social object signature built from a tweet is the following.

Example 3.2. The tweet "`Christopher Walken is making me SO NERVOUS as Captain Hook...and I love it #PeterPanLive`," was posted in New York on December 5th, after a television special that was broadcast by NBC on December 4, 2014, regarding a musical adaptation of Peter Pan in which Christopher Walken had the role of Captain Hook. The signature of this tweet is w_u = {Christopher, Walken, making, NERVOUS, Captain, Hook, love}, w_b = {Christopher Walken, Walken making, making nervous, nervous Captain, Captain Hook, Hook love}, $h_u = \{\#PeterPanLive\}$, $h_b = m_u = m_b = \emptyset$.

Definition 3.3. A social data stream is a continuous and temporal sequence of social objects $so_1 \ldots so_r \ldots$, generated by social media users from an initial time t_0.

We formulate the problem of detecting interesting topics, eventually related to emergency events, as an online data stream clustering algorithm. The method sequentially processes the incoming social objects, one at a time, and incrementally updates clusters. A new tweet will join the most similar cluster, previously generated, if the similarity between it and the cluster is above a given threshold; otherwise, the tweet will form a new cluster. Thus, as time progresses, the arrival of a tweet reporting on a subject never appeared before is deemed a new event, and a new cluster is created from this object.

A social data stream, however, is characterized by a high, and often rapid, variation of the contents posted by users. We thus define *bursty* a cluster that receives an unusually high number of tweets in a short time period. To formalize the concept of burstiness it is necessary to introduce the notions of *active/inactive* cluster and of *time window*.

A cluster is said *active* if it keeps a growing size rate over a temporal horizon, that is, it continues to receive social objects. This means that the topic of discussion is judged interesting and attracts new users. However, when no new object is assigned to a cluster C for a time period, that is, users loose interest on that particular topic, the cluster becomes *inactive* and it is removed from the list of clusters obtained so far.

The time period depends on the application domain; thus, it must be determined by the user needs. To this end, we define the *life-span* of a cluster as the time period between the last centroid update and the time the cluster has been generated, weighted by a time dependent factor.

Definition 3.4. The life-span of a cluster C is defined as $lf = \delta + 2^h \times (t_c - t_0)$, where t_0 is the creation time of the cluster and t_c is the time of the last object assigned to C. δ and h are user defined parameters. δ is the minimum time period allowed to a cluster to survive after its creation, even if no new tweet is added to it within a δ time, while h determines the temporal horizon a cluster is considered active. The decay rate of a cluster is defined as $\lambda = 1/lf$.

Analogously to the concepts introduced by Aggarwal and Yu (2010) for clustering data streams, a time-dependent *fading function* that diminishes the importance of a cluster is defined as follows:

$$f(t_c) = 2^{-\lambda(t_{so}-t_c)} \tag{1}$$

where t_{so} is the publication time of the post tw that could be added to C and λ is the *decay rate* of a cluster. λ establishes the importance of the historical data in the social streams. The lower the value of λ, i.e., the higher the h value, the higher the importance of the historical information maintained in the cluster since the longer the life-span of a cluster.

A cluster C is considered *inactive*, and removed from the list of currently active clusters, if no new objects arrive during its life-span, that is, $t_{so} - t_c \geq lf$, thus

$$f(t_c) = 2^{-\frac{(t_{so}-t_c)}{lf}} \leq 0.5 \tag{2}$$

The detection of bursts is meaningful only for active clusters with respect to a *short time period*. To this end, we must formalize what we mean by "short time period," and which type of test should be performed to discover an unusual growth of the cluster size to deem it bursty.

Let $t_p = (\delta + h)/p$ be the length of the time period (also called window or slot) at the end of which the burstiness of C much be checked, and p is a parameter fixed by the user to choose the number of periods in which $\delta + h$ must be divided. p is a granularity parameter: the higher its value the shorter the time of checking the formation of bursts.

To fulfill the introduced concepts, an efficient and compact data structure for storing and maintaining summary information of all the tweets examined so far and assigned to a cluster C is needed. These summaries have to take into account not only the items occurring in the tweets, but also their frequencies and their temporal evolution. Moreover, the detection of bursty clusters is possible only if the length of the time window t_p is fixed. To this end, the centroid CC of a cluster C is defined as follows.

Definition 3.5. The centroid of a cluster C is a tuple $CC = (c, t_0, t_c, \pi, sgn_C, \mathcal{U}, \mathcal{L}, \mathcal{T})$, where c is the cluster label, t_0 is the creation time of the cluster, t_c is the timestamp of the last time a social object was added to C, π is a variable storing how many time slots elapsed between t_c and t_0, $sgn_C = (sgn, \mathit{ff})$ is the textual signature of C, where sgn is the feature vector analogous to the tweet signature, $\mathit{ff} = (f_{w_u}, f_{w_b}, f_{h_u}, f_{h_b}, f_{m_u}, f_{m_b})$ is the list of frequencies corresponding to the signature; $\mathcal{U} = (u, U, \mu(U), var(U))$, where u is the list of distinct users that posted the tweets assigned to C in the current time window, U the total number of users in the current time window, $\mu(U)$ the incremental mean of the number of users U and $var(U)$ the incremental variance of the number of users U, both relative to the current time window; $\mathcal{L} = (l, L, \mu(L), var(L))$ is analogous to $\mathcal{U}$ where instead of users we consider the locations from where the tweets have been posted. Finally, $\mathcal{T} = (T, \mu(T), var(T))$ contains the incremental mean and variance, relative to the current time window, of the total number of tweets of C.

While sgn_C is used to compute the similarity between a new tweet and an existing cluster, the role of $\mathcal{U}, \mathcal{L}, \mathcal{T}$ is important to discover if a cluster is bursty, i.e., its size presents an unusual growth in a time slot t_p. To this end, to evaluate the appearance of a bursty cluster C, we exploit the well known statistic measure of *Z-score* by computing its value for three features characterizing C: the number of users that sent the tweets assigned to C in a time slot t_p, the number of different locations from which these tweets have been posted, and the number of tweets. A high Z-score of either the number of users or locations or tweets is an indication that in the current time slot there has been an atypical growth of messages. Let x denote the value of one of these three features in the current time period, then the *Z-score* is defined as follows:

$$z(x) = \frac{x - \mu(x)}{\sigma(x)} \tag{3}$$

where μ is the mean value of x computed in all the previous time slots, and σ the corresponding standard deviation.

Since our approach can store only summary information because of the streaming characteristic of data, in order to compute mean and standard deviation of a feature x, we exploit the incremental calculation of these formulas, as explained in (Knuth 1997), and reported in the following:

$$\mu_n = \mu_{n-1} + \frac{1}{n}(x_n - \mu_{n-1}) \tag{4}$$

$$S_n = S_{n-1} + (x_n - \mu_{n-1})(x_n - \mu_n) \tag{5}$$

$$\sigma_n = \sqrt{\frac{S_n}{n}} \tag{6}$$

For our purposes, n corresponds to the current time slot, while $n-1$ to the number of time slots already elapsed. Which is the number n_p of current time slot can be easily computed as

$$n_p = \left\lfloor \frac{t_{so_i} - CC.t_0}{t_p} \right\rfloor \tag{7}$$

where the $\lfloor x \rfloor$ denotes the greatest integer less than or equal to the real number x. In order to compute the *Z-score* of the three features, we apply formulas (4) and (5) for the incremental evaluation of mean and variance as follows. As regards the number of users, let $\mu_{t-1}(\mathcal{U}.U)$ be the average number of users in the previous time windows, then the incremental mean $\mu_t(\mathcal{U}.U)$ and the incremental variance $S_t(\mathcal{U}.U)$ of the number of users are computed as follows:

$$\mu_t(\mathcal{U}.U) = \mu_{t-1}(\mathcal{U}.U) + \frac{1}{n_p}(\mathcal{U}.U - \mu_{t-1}(\mathcal{U}.U)) \tag{8}$$

$$S_t(\mathcal{U}.U) = S_{t-1}(\mathcal{U}.U) + (\mathcal{U}.U - \mu_{t-1}(\mathcal{U}.U))(\mathcal{U}.U - \mu_t(\mathcal{U}.U)) \tag{9}$$

Analogously are calculated the incremental mean and variance of the number of locations $\mathcal{L}.L$ and tweets $\mathcal{T}.T$. Then, a cluster is declared bursty if either $z(\mathcal{U}.U)$, or $z(\mathcal{L}.L)$, or $z(\mathcal{T}.T)$ are greater than 2, that is, one out of the number of users, locations, and tweets, in the current time slot is at least two times the standard deviation above the average values of the time slots considered so far.

In the next section, we will describe the online clustering method and how cluster centroids are updated.

4 THE ONLINE CLUSTERING ALGORITHM *BEATS*

In this section, first the method is described, then the new similarity measure used for assigning a tweet to a cluster is defined.

```
Algorithm BEaTS
Input: A continuous stream of tweets $tw_1, \dots, tw_n, \dots$, a similarity threshold $\epsilon$,
       the minimum survival time period $\delta$, the active time period $h$, the granularity $p$ of the time window $t_p$
Output: The set of currently active clusters $\mathcal{C}$, the set of discovered bursts $\mathcal{B}$

begin
1.  $t_p = (\delta + h)/p$;
3.  $\mathcal{C} \leftarrow \emptyset$;
4.  $\mathcal{B} \leftarrow \emptyset$;
5.  i=1;
6.  $so_1 \leftarrow$ GenerateSocialObject($tw_1$);
7.  $C_1 \leftarrow$ CreateCluster($so_1$);
8.  $\mathcal{C} \leftarrow \mathcal{C} \cup C_1$;
9.  while (not end of stream)
10.     $i \leftarrow i + 1$;
11.     Receive the next tweet $tw_i$;
12.     $so_i \leftarrow$ GenerateSocialObject($tw_i$);
13.     let $t_{so_i}$ be the time stamp of $so_i$
14.     for each cluster $C_j \in \mathcal{C} = \{C_1, \dots, C_k\}$
15.         if isActive($C_j$) then
16.             $sim(so_i, C_j) \leftarrow$ ComputeSimilarity($so_i$,$C_j$);
17.         else
18.             $\mathcal{C} \leftarrow \mathcal{C} - C_j$;
19.         end
20.     end
21.     c=argmax$_{j \in \{1,\dots,k\}}$ sim($so_i, C_j$)
22.     if $sim(so_i, C_c) >= \epsilon$ then
23.         if $(t_{so_i} - C_c.t_0) > C_c.\pi \times t_p$
24.             if CheckBurst($C_c$) then
25                  $\mathcal{B} = \mathcal{B} \cup C_c$;
26.             end if
27.             $C_c.\pi = C_c.\pi + 1$
28.         end if
29.         updateCentroid($C_c, so_i$);
30.     else
31.         $C_i \leftarrow$ CreateCluster($so_i$);
32.         $\mathcal{C} \leftarrow \mathcal{C} \cup C_i$;
33.     end if
34. end while
end
```

Fig. 1. Online clustering algorithm.

4.1 Methods

The *BEaTS* method is an incremental algorithm that sequentially processes the incoming social data streams and groups similar tweets into the same cluster, each cluster characterizing an event. The method is based on the algorithm proposed by Yang et al. (1998), sensibly modified to deal with social data streams and to discover, eventually bursty, clusters.

The pseudo-code of the algorithm is reported in Figure 1. The algorithm receives the continuous stream of tweets as input, along with the similarity threshold ϵ, the minimum survival time period δ and the active time period h of a cluster, and the granularity p of the time window to determine the length of the time slot after which clusters are checked to detect bursts. The first time the algorithm receives a tweet tw_1 from the stream, it generates the first cluster C_1 from the social object so_1 representing tw_1, by storing in the centroid the signature of so_1 and setting the corresponding frequencies to 1. Moreover, C_1 receives the timestamp of so_1, and the value of π is set to 1 (steps 6–8).

While a new tweet tw_i arrives at time t_{so_i}, the algorithm builds the representation so_i of tw_i (steps 11–13) and computes the similarity between the tweet and the clusters active at the timestamp t_{so_i}, while the inactive clusters are removed from the set of clusters (steps 14–20). Let C_c be the cluster whose centroid has maximum similarity with so_i (step 21). If this similarity value $sim(so_i, C_c)$ is lower than ϵ a new cluster is generated from so_i and added to the set of clusters

Algorithm UpdateCentroid
Input: a social object $\mathrm{so} = (\mathrm{o}, \mathrm{u}, \mathrm{t_{so}}, \mathrm{l}, \mathrm{sgn})$, he cluster centroid $CC = (c, t_0, t_c, \pi, sgn_C, \mathcal{U}, \mathcal{L}, \mathcal{T})$
Output: an updated cluster centroid CC

```
begin
    $\mathrm{CC.t_c = so.t_{so}}$
    for i=1 to 6 do
        let $\mathrm{I = so.sgn(i) \cap CC.sgn_C.sgn(i)}$
            $\mathrm{D = so.sgn(i) - I}$
        $\mathrm{CC.sgn_C.sgn(i) = CC.sgn_C.sgn(i) \cup D}$
        for each item in I, add 1 to the corresponding $\mathrm{CC.sgn_C.ff(i)}$
        for each item in D, set to 1 to the corresponding $\mathrm{CC.sgn_C.ff(i)}$
    end
    if $\mathrm{so.u} \in \mathcal{U}.\mathrm{u}$
    then
        $\mathcal{U}.\mathrm{u} = \mathcal{U}.\mathrm{u} \cup so.u$
        $\mathcal{U}.\mathrm{U} = \mathcal{U}.\mathrm{U} + 1$
    end
    if $\mathrm{so.l} \in \mathcal{L}.\mathrm{l}$
    then
        $\mathcal{L}.\mathrm{l} = \mathcal{L}.\mathrm{l} \cup so.l$
        $\mathcal{L}.\mathrm{L} = \mathcal{L}.\mathrm{L} + 1$
    end
    return CC;
end
```

Fig. 2. Update centroid procedure.

(steps 31–32). Otherwise, if the time for checking if C_c is bursty has elapsed (steps 23), the function $CheckBurst(C_c)$ is executed (step 24), and if C_c is a burst, it is added to the set $\mathcal{B}$ of discovered bursts (step 25). In either case, the social object is added to C_c by updating the centroid (step 29). These steps are repeated until the algorithm receives new tweets.

Three important steps of the algorithm are the *centroid maintenance*, the *similarity function computation*, and the *bursts detection*.

The pseudo-code of the method to update a centroid $CC = (c, t_0, t_c, \pi, sgn_C, \mathcal{U}, \mathcal{L}, \mathcal{T})$ when a new tweet tw is added to any cluster C is described in Figure 2. Let $so = (o, u, t_{so}, l, sgn)$ be the social object representing tw. First of all, the timestamp of C is updated with the timestamp t_{so} of so. As regards the signature sgn_C, all the items of each feature must be checked if already present in the centroid signature. Thus, the intersection I and the difference D between the signatures of so and CC are computed. Then, for each feature $so.sgn(i)$ of the signature, if an element of this feature already appears in the feature $sgn_C.sgn(i)$ of the centroid, the corresponding frequency must be incremented by 1, otherwise, it will be added to the centroid feature and its frequency is set to 1. Moreover, both the list of users and locations are updated by either adding a user/location not already present, either incrementing by one the number of users $\mathcal{U}.U$ and locations $\mathcal{L}.L$.

The *CheckBurst* procedure, described in Figure 3, computes the *Z-score* of the number of users $z_t(\mathcal{U}.U)$, of locations $z_t(\mathcal{L}.L)$, and of tweets $z_t(\mathcal{T}.T)$, and it declares a cluster bursty if one of these values is above 2.

4.2 Similarity Measure

The similarity measure used by a method to group objects in the same cluster plays a key role for any clustering algorithm. Since our aim is to group social data concerning the same topic, devising the proper similarity function is crucial for the effectiveness of the algorithm. To this purpose, we can make the following observations:

— stream data objects discussing the same topic are usually temporally close, suggesting the use of a combined measure of lexical similarity and temporal proximity as a criterion for clustering;

```
Algorithm CheckBurst
Input: the cluster $C$ with centroid $CC = (c, t_0, t_c, \pi, sgn_C, \mathcal{U}, \mathcal{L}, \mathcal{T})$
Output: Cluster declared either bursty or not

begin
    $z_t(\mathcal{U}.U) = \frac{\mathcal{U}.U - \mu_t(\mathcal{U}.U)}{\sigma_t(\mathcal{U}.U)}$;
    $z_t(\mathcal{L}.L) = \frac{\mathcal{L}.L - \mu_t(\mathcal{L}.L)}{\sigma_t(\mathcal{L}.L)}$;
    $z_t(\mathcal{T}) = \frac{\mathcal{T}.T - \mu_t(\mathcal{T}.T)}{\sigma_t(\mathcal{T}.T)}$;
    if $(z_t(\mathcal{U}.U) \geq 2)$ or $(z_t(\mathcal{L}.L) \geq 2)$ or $(z_t(\mathcal{T}.T) \geq 2)$ then
        C is bursty;
    end
end
```

Fig. 3. CheckBurst procedure.

—words, hashtags, and mentions appearing more frequently should have a higher weight when computing the similarity;
—a significant change in the lexicon and in TF are reliable indicators that the incoming stream reports on a new topic.

According to these observations, we propose a measure based on the Jaccard similarity that takes into account the lexical similarity of terms, their frequency, and the time distance.

Let $so = (o, u, t_{so}, l, sgn)$ be a social object with signature $sgn = (w_u, w_b, h_u, h_b, m_u, m_b)$, and $CC = (c, t_0, t_c, \pi, sgn_C, \mathcal{U}, \mathcal{L}, \mathcal{T})$ a centroid with signature $sgn_C.sgn = (w_u^C, w_b^C, h_u^C, h_b^C, m_u^C, m_b^C)$ and $sgn_C.ff = (f_{w_u}, f_{w_b}, f_{h_u}, f_{h_b}, f_{m_u}, f_{m_b})$ the corresponding list of frequencies. For each word, hashtag, mention, both unigram and bigram of *so*, the intersection with the terms in the centroid signature and their union are computed. Let $I^i = sgn(i) \cap sgn_C.sgn(i)$ be the intersection of a couple of features, and $U^i = sgn(i) \cup sgn_C.sgn(i)$ their union. Then,

$$sim(so, CC) = \frac{\sum_{j=1}^{|I^i|} sgn_C.ff_I(j)}{\sum_{j=1}^{|U^i|} sgn_C.ff_U(j)} \times f(t_c) \tag{10}$$

where $sgn_C.ff_I(j)$ and $sgn_C.ff_U(j)$ are the sum of frequencies of the terms appearing in the intersection and union, respectively, while $f(t_c)$ is the fading function (Equation (1)) introduced in the previous section. $f(t_c)$ biases the similarity function toward clusters temporally closer to the tweet. The presence of this term allows to take into account not only the similarity between the signatures, but also their time distance, because, as observed above, the same events are usually generated in close temporal horizon.

Example 4.1. Consider the social object of Example 3.2, and suppose to have a cluster with the following word unigrams of the centroid signature w_u^C = {peter, pan, young, viral, pirates, love, Christopher, Walken, killed}, and corresponding frequencies $ff_{w_u} = \{10, 10, 3, 3, 3, 5, 8, 8, 1\}$, then $I^1 = I^{w_u}$ = {Christopher, Walken, love}, $U^1 = U^{w_u}$ = {peter, pan, young,viral,pirates,love, Christopher, Walken, killed, making, nervous, Captain, Hook}. Then, $sgn_C.ff_I(1) = 8 + 8 + 5$, while $sgn_C.ff_U(1) = \{10 + 10 + 3 + 3 + 3 + 5 + 8 + 8 + 1 + 1 + 1 + 1 + 1\}$. The other terms of Equation (10) are computed in the same way.

5 EXPERIMENTAL STUDY

In this section, we report our experimental outcomes to demonstrate the effectiveness and the efficacy of the proposed approach. To this end, we first evaluate the ability of the algorithm in detecting events and compare it with four state-of-the-art methods, *LDA* (Blei et al. 2003), *Doc-p* (Petrović et al. 2010), *SFPM* (Aiello et al. 2013), and *BNgram* (Aiello et al. 2013), on three different datasets, described in the next section. After that, we evaluate the efficacy of *BEaTS* in discovering bursty events in nearly real time, and compare it with *TopicSketch* (Xie et al. 2016), *Twevent* (Li et al. 2012), and *EDCoW* (Weng and Lee 2011), by using the dataset and results made available from the authors.[1] The algorithm has been written in *R* (Feinerer et al. 2008).

5.1 Datasets

The first two datasets come from the data collected in the context of the *Social Sensor Project*,[2] for two major events in 2012: the *FA Cup final*, and the *US Elections* that took place in November 2012. Tweets related to these events were collected, as explained in (Aiello et al. 2013), by using a set of filters on keywords and hashtags, chosen by experts and specific to the target event. Both datasets are relative to just one event, and the topics that can be extracted from them are essentially specific to that event, which are the main episodes recurring during the event. The third dataset, instead, is constituted by all the tweets that have been posted within the area of Manhattan during the month of December 2014, obtained without applying any filter on keywords and hashtags. This dataset thus is not related to a specific argument, but contains events of diverse kind happening during the month.

To evaluate the capability of the approaches to discover the actual events occurring in the datasets, i.e., the so called ground-truth, Aiello et al. (2013) relied on mainstream media reports. They reviewed the published information regarding the event, and built the ground-truth by selecting the most significant stories. The ground-truth consists of a set of keywords and a short headline describing the story. To extract the ground-truth from the Manhattan dataset, we adopted the same approach. In the following details about the three datasets and the corresponding ground-truths are provided.

FA Cup Final. The Football Association Challenge Cup, or FA Cup, as described in (Aiello et al. 2013), is the main competition in English football, and the oldest association football competition in the world (being first held in 1871). In 2012, the match between the two finalists Chelsea and Liverpool lasted 90 minutes plus a 15 minute half-time break. Data was extracted using the official event hashtags, and the names of the teams and key players. The ground-truth includes 13 topics, each of the three goals, some key bookings, and the start, middle, and end of the match.

US Elections. This dataset regards the United States presidential election held on Tuesday, November 6, 2012 when President Barack Obama and his running mate, Vice President Joe Biden, were re-elected, defeating the Republican nominee, Mitt Romney, and his running mate, Paul Ryan. There were also elections for US Senate and the House of Representatives, and for several state governors. Moreover, some states held referendums regarding issues such as same-sex marriage and marijuana legalization. The list of keyword used to extract tweets included the names of the candidates and various hashtags, such as Election2012. The ground-truth comprises 64 topics, announced by US television networks regarding the outcomes of the Presidential race in particular states, but also referendum results, senate race results, and Obama's victory speech. For further details about the two above datasets refer to (Aiello et al. 2013).

Manhattan. To collect the tweets, we implemented a multi-threaded crawler to access the Twitter Streaming API. Specifically, the data extracted are tweets tagged with GPS coordinates within the boundaries of the area of Manhattan in New York City. The collected dataset consists of 671,170 tweets issued by 91,356 mobile users, during the month of December 2014. The average number of tweets extracted per day varies from a minimum value of 11,944 tweets on December 3 and a peak

[1] http://topicsketch.appspot.com/.
[2] www.socialsensor.eu.

of 28,565 tweets on December 13. This peak is particularly significant since it reflects the outcomes of the topic detection algorithm. In fact, as will be illustrated in the following, one of the most relevant topics identified is related to an event that took place in New York on December 13, that is, the march Million March NYC in solidarity with the families of those killed by law enforcement officers.

To reduce the amount of noise, the raw data has been preprocessed by applying tokenization and stemming. We used the packages tm and stringr of R (Feinerer et al. 2008) (e.g., removePunctuation, removeStopWord) to extract cleaned terms from the original messages by removing stopwords and punctuation. We also used the package SnowballC of R for the stemming process to reduce inflected words to their root.

The ground-truth consists of 41 events, extracted by checking the headlines on the web of 10 of the main newswire sources in New York: The New York Times, The Daily News, The Huffington Post, CCN, The Guardian, Wikipedia, NBC News, Fox, New York Post, CBS New York. Analogously to (Aiello et al. 2013), each topic is characterized by a set of keywords, namely the most frequent hashtags and terms used to describe the event.

In Table 1, a sample of the top ground-truth events are listed. For each event, we report the story obtained from different newswire sources and some representative tweets. Differently from the previous dataset, topics vary greatly, ranging from arts and music to sports, from crimes and racism to cinema and finally, due to the period, converging toward Christmas-related argumentation. As can be noted, the topics are well aligned with the textual description of the real-word story.

The top 10 topics include two on December 1: one, Banksy Documentary, refers to the art documentary about the street artist exhibition of Bansksy; the other, Knicks Match, to the NBA basketball match among New York Knicks and Miami Heat in the Madison Square Garden. On December 5, we detected other two topics: one concerns the musical Peter Pan Live! a television special that was broadcast by NBC on December 4–8, 2014; the other topic is the Foo Fighter Concert at Irving Plaza. Finally, we noted that as Christmas was approaching the predominant topics were indeed Christmas related. Accordingly, we obtained a number of clusters about Christmas. We reported in Table 1 only the cluster of December 21 as it was the largest among the Christmas-related ones. Among the topics, particularly relevant are the ones about the Death of Erik Garner. Specifically, we have the following three topics related to this real-life story: (1) one topic, on the December 3, is associated with the grand jury decision to not indict NYPD officers involved in the chokehold case; (2) a second topic concerns the consequent protests that erupted in the city the day after, on December 4; (3) the third topic is related to the march Million March NYC of December 13 organized in solidarity with the families of those killed by law enforcement officers, streamed through the city streets in Washington, New York, Boston, Chicago, and Oakland.

5.2 Evaluation Metrics

To evaluate and compare the results of the methods, we use the metrics adopted in (Aiello et al. 2013). As suggested in this article, the Levenshtein similarity can be used to determine when a topic is considered detected and to overcome problems due to spelling when matching terms. A term in a detected topic matches a ground-truth term if their Levenshtein similarity is greater than 0.8. In the following, G denotes a ground-truth event and P the predicted event that matches all its keywords.

- *Event recall*: $E\text{-}Rec = \frac{TP_E}{TP_E + TN_E}$ is the percentage of ground-truth events successfully detected by a method, where TP_E (true positive) is the number of successfully detected ground-truth events, and $TP_E + TN_E$ (TN_E :True Negative) is the total number of ground-truth events. A ground-truth event is considered successfully detected if there exists a predicted event that matches at least one of its terms.

Table 1. List of the Top 10 Ground-Truth Topics in the Manhattan Dataset

Topic	Day	Real-word stories	Sample tweets
Banksy Documentary	1	31 Days of Mystery: The "Banksy Does New York" Documentary (from Huffington Post)	In honor of taking this picture before
		Banksy Does NY: HBO's Newest Doc Takes Us Back To Banksy's Residency in NYC (from Arts & Culture, New York, News)	I knew what it was and watching the #banksy
Knicks Match	1	New York Knicks vs. Miami Heat: Betting Odds, Tips (from tipsterlabs.com)	#Knicks vs. #Heat @ #MadisonSquareGarden #MSG #NYC @ Madison Square Garden
		New York Knicks - Madison Square Garden, New York City (from The Garden)	Knicks vs. Heat #melo @ Madison Square Garden
Next Wave Festival	2	"Howie the Rookie," in BAM's Next Wave Festival (from The New York Times)	#BAM announces 2014 #BAMNextWave Festival
		BAM 2014 Next Wave Festival, the largest experiment in live performance (from Explore Brooklyn)	
Death of Eric Garner	3	Eric Garner's son wants cop indicted for father's death, assures no Ferguson-like riots in Staten Island (from DailyNews)	Praying for no riots tonight in #NYC after #EricGarner grand jury decision
		Wave of protests after Grand Jury doesn't indict officer in Eric Garner chokehold case (from the New York Times)	Nyc March heading up 5th ave #EricGarner #BlackLivesMatter
Death of Eric Garner	4	Protests erupt in New York City after a grand jury decides not to indict NYPD officers Daniel Pantaleo and Justin Damico in the death of Garner (from Wikipedia)	Chelsea, #NYC. Cops in undercover cars. Protesters unite with #reasonablecause. #stopracism #stophate #enoughisenough #justice #EricGarner
		Eric Garner grand jury decision sparks protests which shut down West Side Highway, Brooklyn Bridge and Lincoln Tunnel (from DailyNews)	Several scattered groups of protesters around Manhattan. #EricGarner
		Eric Garner protests erupt in New York (from The Guardian)	
Peter Pan Live	5	Peter Pan Live! is a television special that was broadcast by NBC on December 4, 2014, starring Allison Williams in the title role and Christopher Walken as Captain Hook (from Wikipedia)	Christopher Walken is making me SO NERVOUS as Captain Hook...and I love it. #PeterPanLive.
		Peter Pan Live! with Allison Williams and Christopher Walken on NBC (from the New York Times)	Christopher Walken tapping is all I need to see in life. #PeterPanLive
Foo Fighters Concert	5	Foo Fighters End Cross-Country Sonic Journey With Marathon New York Show (from The RolllingStones)	@foofighters Please let Irving Plaza fans who've been standing in the cold for hours keep their place in line! We just wanna ROCK
		Foo Fighters Setlist at Irving Plaza, New York, NY, USA (from NewYork.com)	I guess that was it for @foofighters - BS on not waiting in line before 3pm at @IrvingPlaza - tickets were sold out at 1pm #IrvingPlaza #Foo.
Million March NYC	13	Justice for All and Million March NYC police brutality protests ? how the day unfolded (from The Guardian)	Amidst the #HandsUpDontShoot chant at #MillionsMarchNYC, hearing a few yell Fight back! All white, in case you were wondering
		25,000 March in New York to Protest Police Violence (from the New York Times)	
		Two NYPD cops assaulted as Manhattan march over Eric Garner case turns ugly (from Daily News)	Given that cops kill 500+ people a year, this is an indictment rate of 1%. #MillionsMarchNYC
Bocelli Concert	17	Andrea Bocelli Concert at Madison Square Garden, New York City. (from Youtube.com)	#Bocelli Concert @ #MSG
		Andrea Bocelli USA December Tour Concert in New York Concert at the Madison Square Garden MSG NYC on December 17, 2014 (from http://manhattan.backpage.com/)	
Christmas Time	21	Christmas Time	Merry Christmas; Happy Holidays to all! #Rockefeller #christmas in #nyc @ Rockefeller Center http
			#Rockefeller #Christmas tree- pretty even in the dreary, drizzly weather! #NYC @ Rockefeller Center

— *Keyword precision*: Given a predicted event P, *K-Prec* $= \frac{TP_K}{TP_K+FP_K}$ is the percentage of correctly detected keywords TP_K of that event out of the total number $(TP_K + FP_K)$ of keywords identifying that event (FP_K : false positive). A keyword is said correctly detected if it matches with a term contained in some ground-truth event. The total precision of a method is computed by averaging the individual precision scores over all the detected events.
— *Keyword recall*: Given a predicted event P, *K-Rec* $= \frac{TP_K}{TP_K+TN_K}$ is the percentage of correctly detected keywords TP_K over the total number of keywords of the ground-truth event G. The total recall is similarly computed by averaging over all the detected events.

These scores have been computed by using an evaluation script provided by Aiello et al. (2013), at the top N events generated by the contestant algorithms, for a range of values of N.

5.3 Evaluation of Discovered Events

In this section, we evaluate the ability of our approach to detect events from the Twitter datasets, described in Section 5.1, and compare them with the four contestant methods *LDA* (Blei et al. 2003), *Doc-p* (Petrović et al. 2010), *SFPM* (Aiello et al. 2013), and *BNgram* (Aiello et al. 2013). Note that these methods do not distinguish bursts among the obtained events, thus the comparison can be made only with respect to the detected events. The software implementing these methods has been provided by Aiello et al. (2013). We executed them by using the same parameter setting adopted in (Aiello et al. 2013). More in detail, *LDA* needs the expected number of clusters and keywords, which have been fixed to 200 and 15, respectively. The input for *Doc-p* is the similarity threshold, set to 0.8. As regards *SFPM* and *BNgram*, we maintained the parameters set by the authors.

BEaTS has as input the parameters δ, the minimum survival time period, h, the active time period, and the similarity threshold ϵ. Since for this experiment we do not search for bursts, the granularity p of the time window has been fixed such that the *CheckBursts* procedure is never executed. The optimal choice of these parameters must take into account the nature of the datasets, in particular the following two key features are crucial for this choice: (1) single or multiple events dataset; (2) time duration. Thus, it is not possible to fix equal values for all the datasets. We executed a set of experiments to determine the best parameter configuration of δ, h, and ϵ, for each dataset by considering the characteristics of each dataset. Thus, we considered different values of δ, h, and ϵ, and computed the event recall for increasing number of events to search for. In particular, for the FA Cup dataset, Figure 4(a) shows the recall obtained by our approach for $\delta = 10$, $h = 5$, and $\epsilon = \{0.5, 0.7, 0.9\}$, and $\delta = 20$, $h = 10$, and $\epsilon = \{0.5, 0.7, 0.9\}$, where the time is expressed in minutes.

The figure points out that the best event recall is achieved when $\delta = 10, h = 5, \epsilon = 0.5$. In fact, from the graph it is evident that for higher value of δ and similarity threshold ϵ the recall gets worse. After that, fixed $\delta = 10$ and $h = 5$ we computed keyword precision and recall for increasing values of the similarity threshold ϵ. Figures 4(b) and 4(c) show similar trend for both keyword precision and recall. Specifically, their values decrease with increasing similarity threshold, thus we chose $\epsilon = 0.5$. This configuration has been used throughout the experiments on the *FA Cup* dataset. In the same way we determined the best parameter configuration for the other two datasets. Figures (5(a–c) and 6(a–c)) show how we determined the optimal values of the parameters for the *US Elections* and *Manhattan* datasets, i.e., $\delta = 20, h = 10, \epsilon = 0.5$ for the *US Elections* dataset and $\delta = 60, h = 15, \epsilon = 0.5$ for the *Manhattan* dataset.

We start the comparison with related approaches by showing in Table 2 the precision and recall metrics in the case of fixed number N of events. Specifically, for the FA Cup dataset the number of events is set to $N = 2$ and for the US Election dataset $N = 10$. These values are those chosen in (Aiello et al. 2013). For the Manhattan dataset we set $N = 20$.

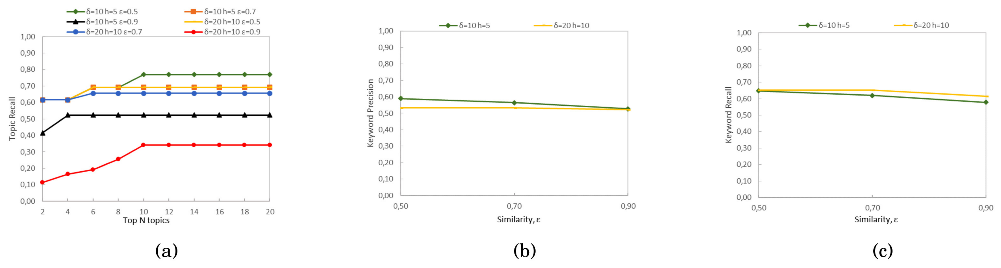


Fig. 4. FA CUP—Event recall (a), Keyword Precision (b), and Keyword Recall (c) of *BEaTS* algorithm for different values of setting parameters.

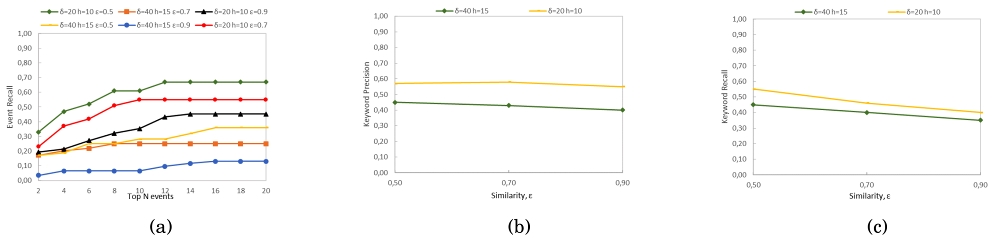


Fig. 5. US Elections—Event recall (a), Keyword Precision (b), and Keyword Recall (c) of *BEaTS* algorithm for different values of setting parameters.

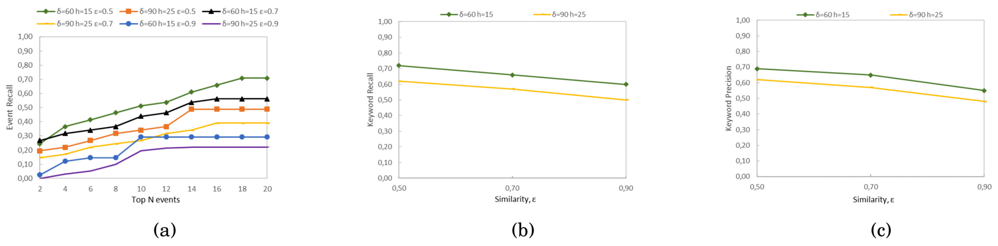


Fig. 6. Manhattan—Event recall (a), Keyword Precision (b), and Keyword Recall (c) of *BEaTS* algorithm for different values of setting parameters.

Table 2. Comparison of Topic Detection Algorithms, in Terms of Event Recall, Keyword Precision, and Keyword Recall for the FA Cup, US Elections, and Manhattan Datasets

	FA Cup			US Election			Manhattan		
Method	E-REC@2	K-PREC@2	K-REC@2	E-REC@10	K-PREC@10	K-REC@10	E-REC@20	K-PREC@20	K-REC@20
LDA	0	0	0	0.45	0.31	0.53	0.39	0.60	0.50
Doc-p	0.54	0.31	0.53	0.06	0.28	0.43	0.24	0.30	0.30
SFPM	0.54	0.40	0.61	0.28	0.37	0.52	0.22	0.42	0.14
BNgram	0.36	0.32	0.62	0.25	0.42	0.52	0.32	0.55	0.12
BEaTS	**0.61**	**0.59**	**0.65**	**0.67**	**0.57**	**0.55**	**0.70**	**0.69**	**0.72**

The highest values are highlighted in bold.

We observe that for the FA Cup dataset at $N = 2$ the proposed approach *BEaTS* achieved the best event recall, though *Doc-p* and *SFPM* obtain a value of 0.54, close to that of *BEaTS*. Furthermore, *BEaTS* exhibits the highest keyword precision and recall. This means that the keywords appearing in the topics detected by *BEaTS* are pretty clean and well describe the event. This is confirmed by

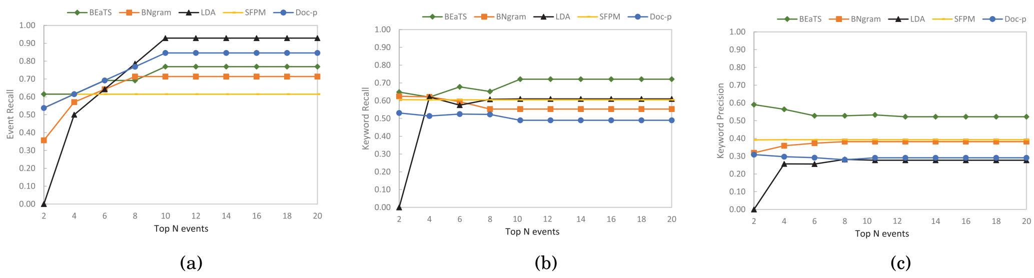


Fig. 7. Event recall (a), Keyword Recall (b), and Keyword Precision (c) @ N of the different algorithms for the FA Cup dataset.

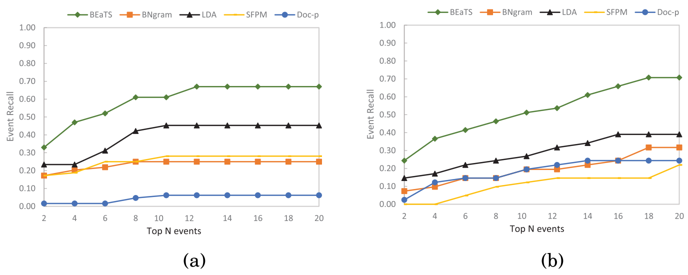
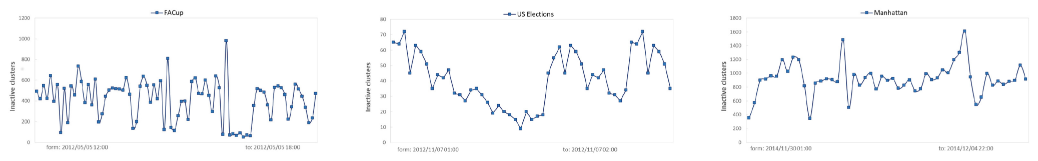


Fig. 8. Event recall @ N of the different algorithms for the US Elections dataset (a) and for the Manhattan dataset (b).

the very good keyword recall (0.65). For the US Election dataset the event recall at $N = 10$ achieved by *BEaTS* is considerable higher than that obtained by the other methods, especially *Doc-p* that gets the worst performance. Also the keyword precision is significantly better than the contestant methods. The keyword recall, instead, has values very close for all the methods. Same trend is obtained for the Manhattan dataset. *BEaTS* achieves an event recall of 0.70, while *LDA* gives the best value 0.39 out of the other methods. Analogously for keyword precision and recall, 0.69 and 0.72 compared to 0.60 and 0.50 of *LDA*, respectively. To explore the variation of the performance when more events are produced, we studied the performance metrics as the number of top results considered varies.

Figure 7(a) shows that for the FA Cup dataset *BEaTS* achieves the higher event recall for smaller values of N, then its recall curve gets flat and reaches the value of 0.769. Although *LDA* for small values of N exhibits the worst performance, it then improves with the number of detected topics till reaching the highest recall of about 0.93. For this dataset dealing with a focused subject, i.e., the a football match, standard topic model approaches like *LDA* and *Doc-p* achieved the best event recall when the number of detected topics increases. However, as highlighted in Figures 7(b) and 7(c) our method reaches the best keyword recall and precision. Note that these values remain rather stable for all the methods, when the number of topics increases.

Figure 8(a) displays the event recall obtained for the US Election dataset. The results obtained show that *BEaTS* achieved the highest event recall, followed by *LDA*. Note the very poor performance obtained by *Doc-p*. For all the methods, the event recall remains quite stable when the number of produced topics is greater than 8. The keyword recall is almost the same for varying number of topics and assumes around the value of about 0.52 for all the methods except for *Doc-p* that reached 0.42, thus it has not been visualized. Also the keyword precision remains constant with the topics and ranges from 0.57 for *BEaTS* to 0.28 for *Doc-p*.



(a) (b) (c)

Fig. 9. Number of clusters that become inactive after a bunch of 1,000 tweets is examined. (a) The FACup, (b) the US Elections, and (c) the Manhattan dataset, respectively.

The results obtained for the Manhattan dataset, displayed in Figure 8(b), confirm the competitive results in terms of performance of *BEaTS* with respect to the other methods. It is worth to point out that this dataset is more difficult because the events can be on many different topics. *BEaTS* achieved the highest event recall, followed by *LDA* and BNgram. SFPM had the worst performance. Also in this case we do not show keyword precision and recall since for all the methods they remain stable with the number of topics, with values very close to that reported in Table 2 for $N = 20$.

The performances achieved by the methods with the three datasets reflect the different nature of the target events. For the FA Cup dataset users comment on a focused subject for a short time period. The stories about the primaries in US are more difficult to capture. This is even more evident for the Manhattan dataset that has been obtained without any filter, and thus contains information referring multiple events spanning rather different arguments.

From these results we can observe that standard topic detection methods, like *LDA* and *Doc-p*, can perform reasonably well on very focused events, while their performance can be low when considering more general events. On the contrary *BEaTS* improves its performance on more complicated data plenty of very different and heterogeneous events, like the US Elections dataset and even more the Manhattan dataset.

Therefore, we can state that *BEaTS* is able to retrieve more topics than the other methods and the keywords appearing in such topics are pretty clean as our method presents the highest precision.[3]

To conclude this section, we want to show the benefits of the fading function on the number of clusters that become inactive as new tweets are examined. Figure 9 reports the number of clusters that are discarded after a bunch of 1,000 tweets is considered for all the datasets. As the figure highlights, the fading function sensibly influences the results of our approach. In fact, it allows to discard a relevant number of clusters that do not grow in size, thus of no interest. Maintaining them would only slow down the method.

5.4 Computational Complexity and Execution Time

BEaTS is an incremental method that examines one tweet at a time and assigns it to the cluster whose centroid is the most similar. Thus, its capability to deal with a large number of tweets depends on the current number of active clusters. Let us assume that the number of active clusters at time t be k. The time requirements of *BEaTS* are dominated by the similarity function, because the similarity of each new tweet and all the current centroids must be computed. Thus, if at time t the number of active clusters so far is k and the total number of tweets to examine is n, the algorithm has a time complexity of $O(nk)$, since the other steps can be executed in time linear with respect to either k or n.

[3]It is worth to point out that the results reported in this article for the contestant methods are different from those published in (Aiello et al. 2013), since the tweets of FA Cup Final and US Election datasets are not exactly the same of those used in (Aiello et al. 2013). Thus, we executed the methods on the updated datasets and not on the original ones tested by Aiello et al.

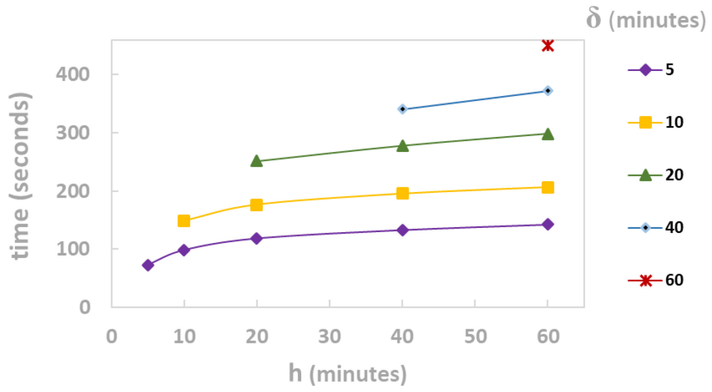


Fig. 10. Execution time of *BEaTS* on dataset composed of 21,504 tweets posted within the area of Manhattan on December 22, 2014.

It is worth pointing out that the number of active clusters depends on the two parameters δ and h, the former being the minimum time period a cluster is allowed to survive until it receives new tweets, the latter determining the life-span of a cluster. In order to show the running time required by the method to examine a set of tweets, *BEaTS* has been executed on a computer with Intel(R) Xeon(R) CPU E5-2670 2.60 GHz and 128GB RAM for the set of 21,504 tweets posted within the area of Manhattan on December 22nd, 2014 with different values of δ and h, and the similarity threshold ϵ fixed to 0.5.

Figure 10 shows the execution times required by *BEaTS* to assign the 21,504 tweets to the proper cluster when h and δ range from 5 to 60 minutes. The figure points out that the execution time increases almost linearly when these parameters increase. Note that the value of h, i.e., the temporal horizon a cluster is considered active, must be less than or equal to δ, the minimum time period allowed to a cluster to survive after its creation. Thus, for instance, if δ is fixed to 5 minutes, and h is also 5 minutes, the method needs around 60 seconds to process the 21,504 tweets. The time augments to almost 140 seconds if $h = 60$, thus the increase is almost linear. Considering that the similarity computation can be executed in parallel, a parallel implementation of the method could sensibly reduce the execution times, allowing to process streams of tweets arriving at a very fast rate.

5.5 Evaluation of Bursty Events

In this section, we first analyze the results obtained by our algorithm in terms of bursty events on the Manhattan dataset since, as already observed, the other two datasets are event-specific and thus report only topics about single events. Then, we compare *BEaTS* with *TopicSketch* (Xie et al. 2016), *Twevent* (Li et al. 2012), and *EDCoW* (Weng and Lee 2011) on the same Singapore collection of tweets, originally used by *EDCoW* for evaluation. We show that our approach is able to discover the bursty events detected by these methods, by providing a much richer description of the events.

As described in Section 4, the algorithm *BEaTS* after creating a cluster, triggers its monitoring to identify if any of the cluster features becomes bursty. Specifically, the centroid features $\mathcal{U} = (u, U, \mu(U), var(U))$, $\mathcal{L} = (l, L, \mu(L), var(L))$, and $\mathcal{T} = (T, \mu(T), var(T))$ are updated each time a tweet is added to the cluster and aggregated into temporal slots. At the end of each time slot t the algorithm verifies whether any of the *Z-score* $z_t(\mathcal{U}.U)$ of the number of users, $z_t(\mathcal{L}.L)$ of the number of locations, $z_t(\mathcal{T}.T)$ of the number of tweets is above 2, that is, the cluster is bursty.

Some of these bursty clusters do no correspond to any real-life events, and some of them become bursty several times during the day. This is the case of the cluster associated with the death of

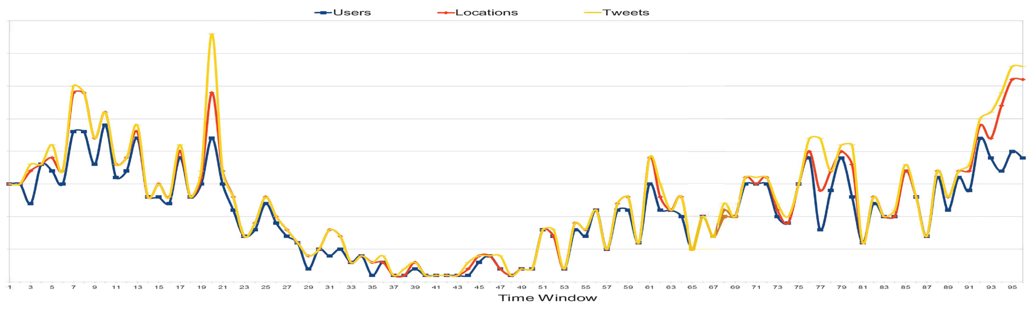


Fig. 11. Features evolution w.r.t time window for the event Erik Garner on December 5th.

Erik Garner, due to the high resonance of the event, and, in general it is the case of the most relevant events. Focusing on this main event occurred during December 2014, Figure 11 shows the evolution of this cluster in terms of the bursty features. We can observe a fluctuating trend of values that are discontinuously concentrated in a narrow range. The cluster becomes bursty in terms of locations and tweets at time window 7 and at time window 20. After that the *Z*-score values are decreasing considerably and start to be consistent again at time window 76 and 95, where the cluster becomes again bursty. This fluctuating trend reflects the effective events of the day when the first news about the judge decision started to be disseminated; then, it maintains a constant interest as it is proved by the almost constant values of the features. After that when the news related to the event are widespread and people start to organize protests, the topic becomes again bursty. To analyze the bursts obtained we introduce the two following metrics, and evaluate them for different configurations of the input parameters.

— *Burst Rate (BR)*. It is the percentage of unique bursty events out of the total number of events detected: $BR = \frac{Distinct\ Bursty\ Events}{Total\ Events}$.
— *Distinct Burst Percentage (DBP)*. It is the percentage of unique bursty events out to the total number of bursty events detected: $DBP = \frac{Distinct\ Bursty\ Events}{Total\ Bursts}$.

We tested different configurations of the algorithm parameters for identifying the setting giving the highest rate of detected burst events. To this end, we considered day 4th of December for two combinations of values for δ and h, i.e., $\{180, 50\}$ and $\{60, 15\}$, and varied the value of the parameter p that determines the granularity of the time window, i.e., the number of time slots in which $\delta + h$ is divided, and, consequently, the frequency of checking burst occurrences. We considered p for the range of values $\{5, 15, 30\}$. This implies that, for example, if $\delta = 60$, $h = 15$, and $p = 15$, $t_p = (60 + 15)/15 = 5$, that is, the burstiness of clusters is checked every 5 minutes.

Figures 12(a) and 12(b) show that the *BR* and the *DBP* are higher when the granularity parameter is set to 15. This trend is similar for both the combinations of values of δ and h, and for all the three features. The highest value, however, is obtained for $\delta = 60, h = 15$ when considering the number of locations $\mathcal{L}.l$. Figure 12(b) shows that the percentage of distinct bursts is higher also for $p = 15$ and location feature, and that higher values of δ and h allow to detect a higher number of clusters declared bursty more times. This is expected, since the algorithm checks if a cluster is bursty after a longer time period, thus it is more likely that it will not be bursty. This setting of parameters is confirmed in Figure 13(a), where the Bursty Rate is reported for days 5th, 13th, and 23rd of December. We thus compared the performance of the three features with respect to different *Z-score* values when $\delta = 60, h = 15, p = 15$. Specifically, Figure 13(b) shows the percentage of distinct bursts over the days by using different *Z-score* threshold values. As expected, the percentage of bursts decreases with increasing values of the the *Z-score* for all the features. When

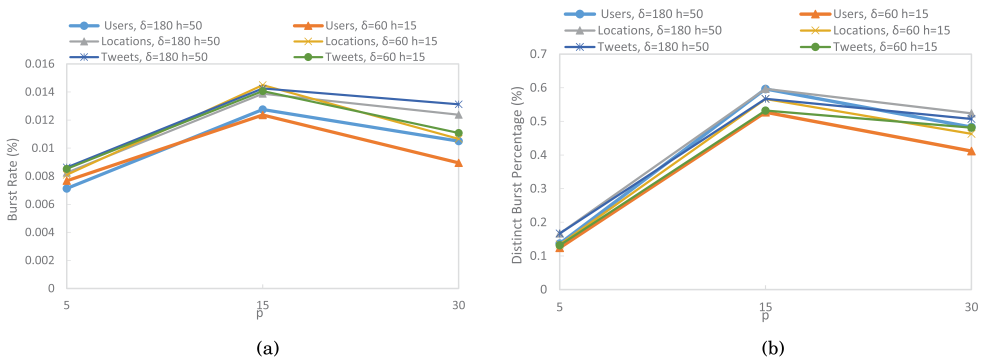


Fig. 12. Burst Rate (a) and Distinct Burst Percentage (b) w.r.t p for different values of δ and different features.

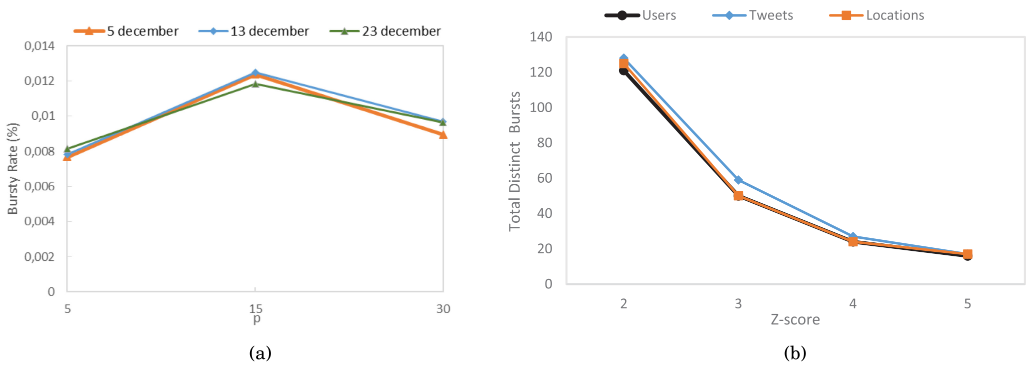


Fig. 13. (a) Bursty rate with $\delta = 60$, $h = 15$, and $p = \{5, 15, 30\}$. (b) Percentage of Distinct Bursts with increasing *Z-score* for different features when $\delta = 60$, $h = 15$, and $p = 15$.

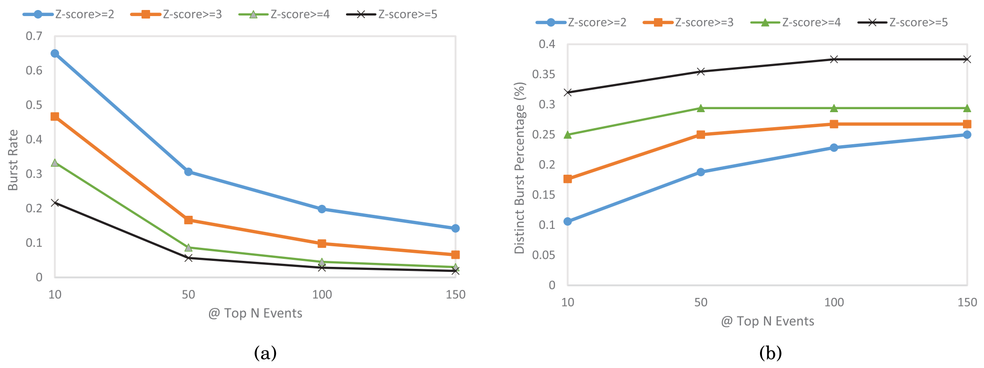


Fig. 14. Bursty Rate (a) and Distinct Bursty Percentage (b) w.r.t top N events for different values of Z-score.

considering the number of tweets, a slightly higher number of distinct bursts are detected. With the last set of experiments, we study the performance metrics as the number of top N events detected varies, where N is the size of the cluster. In this experiment, we report only the results when considering the tweet feature, since for the others we obtain similar values. Figure 14(a) shows how the BR diminishes as the number N of top detected events increases, for all the considered days and for different *Z-score* threshold values, and Figure 14(b) shows that the percentage of unique

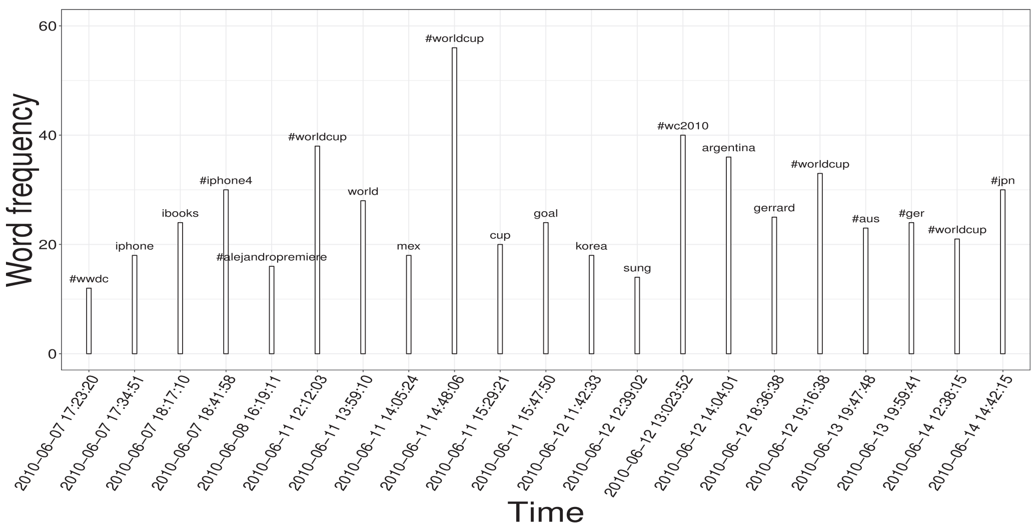


Fig. 15. Bursty topics detected by *BEaTS* in the period June 7, 2010 to June 14, 2010.

bursts increases with the number of top events, for all the *Z-score* values. This suggests that bursts are identified within the top events. It is worth to note the differences for increasing values of the *Z-score*. For instance, when considering the top $N = 10$ events, with a *Z-score* greater than 2 we find that almost the 65% of clusters are considered bursty, while for *Z-score* ≥ 5 this percentage reduces to 20%. Thus, the chosen *Z-score* value plays an important role when it is necessary to distinguish between highly unusual and rarely occurring events, like a disaster, that catch the attention of a lot of people, and less breaking events, like information spreading about a traffic jam.

Comparison with related works. To evaluate the effectiveness of the proposed solution to catch bursty events, we compare *BEaTS* against representative state-of-the-art Twitter bursty event detection systems like *EDCoW* (Weng and Lee 2011), *Twevent* (Li et al. 2012), and *TopicSketch* (Xie et al. 2016).

To perform the comparison, we used a collection of tweets published by users in Singapore in June 2010, which is a collection of 4,331,937 tweets posted by 19,256 unique Singapore based users. This dataset was built by Weng and Lee for evaluating the *EDCoW* event detection method in (Weng and Lee 2011). The same dataset has been used for the evaluation of *Twevent* in (Li et al. 2012) and of *TopicSketch* in (Xie et al. 2016). Specifically, we downloaded the set of tweets made available by Xie et al. (2016)[4] posted between June 7 and June 14, 2010. This period has been selected since three main events occurred, i.e., the *Apple WWDC 2010*, the *MTV Movie Awards*, and the *FIFA World Cup 2010*.

Since, as outlined by Li et al. (2012) and Xie et al. (2016), there is no ground-truth along with this dataset, analogously to them, we conduct a qualitative analysis by showing the differences between the bursty events discovered by *BEaTS* and the contestant systems. *TopicSketch* detected 24 bursty topics in the considered period. *BEaTS* found 21 bursty events out of these 24, at almost the same time. *Twevent* detected 22 topics while *EDCoW* only 10 events; these two latter systems do not specify the hour at which the topics become bursty but only the day. Figure 15 shows the bursty topics detected by *BEaTS*, each represented with the keyword having the highest frequency, while Table 3 reports four of the most significant topics, detected by the different algorithms, i.e.,

[4]http://topicsketch.appspot.com/.

Table 3. *BEaTS* vs. *Twevent*, *EDCoW*, and *TopicSketch*

Day	Topic	*BEaTS*	Twevent	EDCoW	TopicSketch
07	Release of Zynga's Farmville social game for iPhone	#wwdc, iphone, called, hd, multiple, coming, buttugly, custom, talking, zynga, farmville, facebook, ipad;	steve jobs, imovie, wwdc, iphone, wifi;	#iphone4, ios4, iphone;	iphone, farmville, #wwdc, talking, facebook, comes, zynga, coming, goes, day
08	Alejandro by Lady GaGa was premiered officially on June 8, 2010	3rd, amazing, 1st, im, alejandro, video, private, #alejandropremiere, twitter, gaga, music, lady, premier, gaga's, earlier;	lady gaga, music video, gaga, mv, alejandro;	–	alejandro, video, gaga, lady, music, #alejandropremiere, gaga's, amazing, love, im
11	South Africa vs. Mexico, the opening match of the World Cup 2010	mex, world, cup, opening, ceremony, match, vs., south, africa, start, mexico, watch, wrong, fifa, starts, starting, 2010, watching;	south africa, vs. mexico, mexico, goal, first goal;	–	world, match, watching, south, cup, mexico, kick, 2010, africa, vs.
12	South Korea vs. Greece in World Cup 2010	#worldcup, korea, greece, going, half, time, soon, scored, yay, south, jungsoo, 1, 0, nice, fast, goal, give, vs.;	south korea, greece, korea vs. greece, korea won, korea;	#kor, greec, #gre;	korea, goal, south, greece, 1–0, scored, yay, vs., fast

the release of Zynga's Farmville, the release of the music video Alejandro by Lady GaGa, the inaugural ceremony of the World Cup 2010 and the opening match South Africa vs. Mexico. The table shows the differences among the algorithms in terms of words associated with the topics, reproduced from the respective original papers.

Due to lack of space, the complete list of topics detected by the different algorithms, is reported in the Appendix. All events detected by *EDCoW* are reported in Table 1, page 407 in (Weng and Lee 2011), the ones detected by *Twevent* in Table 2, page 163 in (Li et al. 2012).

From Table 3, the following two general observations can be made: (1) the keywords detected by *EDCoW* are less informative compared to the other systems; (2) *BEaTS* and *TopicSketch* detect more semantically meaningful keywords than *Twevent*.

By looking at each single event reported in Table 3, the following comments hold. On June 7, 2010, Steve Jobs revealed the release of iPhone 4, offering new interesting features, such as Farmville client, retina display, and iMovie. This announcement gave rise to a surge of tweets. *BEaTS* and *TopicSketch* detected four bursty events regarding this topic, while *Twevent* and *EdCow* only one. In Table 3, the event occurred at 17:23 on June 7, and the list of keywords obtained by *BEaTS*, *Twevent*, *EDCoW*, and *TopicSketch*, respectively, characterizing the event are reported. From the table, we can observe that *BEaTS* provides a much richer description of the event. In fact, the cluster declared bursty by *BEaTS* at this time contains many more significant words of those detected by all the other systems, including *TopicSketch*. Regarding the *MTV Movie Awards*, *BEaTS* detected the event about the premiere of the music video "Alejandro" by Lady Gaga on June 8, however it missed, compared to *TopicSketch*, the win of the SS501 band (June 11) and the welcome to Twitter of the Korean singer Taec-yeon (June 10). Anyway, we highlight that those two events are not so relevant and actually are not really bursty and, in fact, are detected neither by *EDCoW* nor by *Twevent*. In particular, *EDCoW* did not detect any topic related to the *MTV Movie Awards* event while *Twevent* detected the premiere of the music video "Alejandro" on June 8 and the same topic (with exactly the same words) on June 10. Also for such an event *BEaTS* and *TopicSketch* detect more semantically meaningful keywords/phrases than *Twevent*.

On June 11, 2010, the opening ceremony of the FIFA World Cup 2010 took place with the match South Africa vs. Mexico. A huge number of tweets have been posted worldwide on such event with plenty of bursty topics represented with the hashtag *#worldcup* and several keywords associated with the name of the football teams (e.g., *#ger* standing for Germany, *#aus* standing for Austria, *#ned* standing for Netherlands) or the football players (e.g., *gerrard*). In general, both *BEaTS* and *TopicSketch* detect temporally ordered topics that are more descriptive of the corresponding single events detected by *EDCoW* and *Twevent*. For example, by focusing on the FIFA World Cup 2010 inaugural match, both *BEaTS* and *TopicSketch* detect six topics detailing the key moments of the match, while *Twevent* detects only one event when the first goal has been scored; surprisingly, *EDCoW* does not discover a so significant event. Specifically, as can be argued from the Appendix, on June 11 the following bursty topics about the South Africa vs. Mexico match have been detected. At 12:12 the world cup opening ceremony topic has been detected; at 13:59 the inaugural match South Africa vs. Mexico is going to start. The match is already started at 14:05. The first goal is scored by South Africa at 14:48; at 15:29 Mexico scored a goal by drawing the match. Finally, the match ends at 15:47 in a draw.

By concluding, in general, *Twevent* and *EDCow* are better at discovering events discussed over a long period of time, but may miss events with shorter bursts that instead *BEaTS* and *TopicSketch* are able to detect. Regardless may common features, it is worth pointing out that *BEaTS* and *TopicSketch* are rather dissimilar. In fact, as described in Section 2, *TopicSketch* discovers bursty events by leveraging on the maintenance of data sketch relative to every pair and triple of words appearing in the tweet stream, by fixing both the number of distinct words in the vocabulary and the maximum number of latent topics the method can discover. This latter assumption, as stated by the authors, makes *TopicSketch* not suitable for streams of documents with multiple topics. Moreover, the method is able to detect only bursty events, but not topics discussed for a long period of time. *BEaTS* has been designed to detect trending topics in an eventually long period, as well bursty events rising in a short time period by clustering similar tweets. In fact, as showed in Section 5.3, *BEaTS* performs very well on data plenty of very different and heterogeneous events, like the US Elections dataset and the Manhattan dataset. Our method does not limit either the vocabulary size, nor the number of topics to find, without the problem of space complexity since only the active clusters continue to be present.

6 CONCLUSIONS

The article presented an online algorithm that incrementally groups streams of similar tweets, continuously generated over time, into clusters. Data analyzed so far is summarized and maintained into the cluster centroid by taking into account both textual and temporal features extracted from tweets. The concepts of active/inactive cluster and time window have been introduced to take into account the importance of historical data. A comparison with state-of-the-art approaches on datasets of tweets addressing different issues shows that the method is capable to detect discussions regarding topics of interest, as well unusual and bursty events, revealed by a sudden spreading of attention on messages published by users.

ACKNOWLEDGMENTS

We wish to thank Georgios Petkos for having given us the datasets and the code of the algorithms we compared with, and for his support in executing these methods. We also wish to thank Nicola Procopio for having implemented and experimentally evaluated the first version of the algorithm.

REFERENCES

Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. EvenTweet: Online localized event detection from Twitter. *Proceedings of the VLDB Endowment* 6, 12 (Aug. 2013), 1326–1329.

Charu C. Aggarwal and Philip S. Yu. 2010. On clustering massive text and categorical data streams. *Knowledge and Information Systems* 24, 2 (2010), 171–196.

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94).* 487–499.

Luca Maria Aiello, Georgios Petkos, Carlos Martin, David Corney, Symeon Papadopoulos, Ryan Skraba, Ayse Goker, Ioannis Kompatsiaris, and Alejandro Jaimes. 2013. Sensing trending topics in Twitter. *IEEE Transactions on Multimedia* 15, 6 (2013), 1268–1282.

James Allan. 2002. *Topic Detection and Tracking: Event-based Information Organization.* Kluwer Academic Publisher, Norwel, MA.

James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, Yiming Yang, James Allan Umass, Brian Archibald Cmu, Doug Beeferman Cmu, Adam Berger Cmu, Ralf Brown Cmu, Ira Carp Dragon, George Doddington Darpa, Alex Hauptmann Cmu, John Lafferty Cmu, Victor Lavrenko Umass, Xin Liu Cmu, Steve Lowe Dragon, Paul Van Mulbregt Dragon, Ron Papka Umass, Thomas Pierce Cmu, Jay Ponte Umass, and Mike Scudder Umass. 1998. Topic detection and tracking pilot study final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop.* 194–218.

Nasser Alsaedi, Pete Burnap, and Omer Rana. 2017. Can we predict a riot? Disruptive event detection using Twitter. *ACM Transactions on Internet Technology* 17, 2 (Mar. 2017), 18:1–18:26.

Farzindar Atefeh and Wael Khreich. 2015. A survey of techniques for event detection in Twitter. *Computational Intelligence* 31, 1 (Feb. 2015), 132–164.

Hila Becker, Mor Naaman, and Luis Gravano. 2011. Beyond trending topics: Real-world event identification on Twitter. In *Proceedings of the 5th International Conference on Weblogs and Social Media.*

David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 2003.

Mário Cordeiro and João Gama. 2016. *Online Social Networks Event Detection: A Survey.* Springer International Publishing, Cham, 1–41.

Ingo Feinerer, Kurt Hornik, and David Meyer. 2008. Text mining infrastructure in R. *Journal of Statistical Software* 25, 5 (31 Mar. 2008), 1–54.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S. Yu, and Hongjun Lu. 2005. Parameter free bursty events detection in text streams. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05).* 181–192.

Anuradha Goswami and Ajey Kumar. 2016. A survey of event detection techniques in online social networks. *Social Network Analysis and Mining* 6, 1 (17 Nov. 2016), 107.

Adrien Guille and Cécile Favre. 2015. Event detection, tracking, and visualization in Twitter: A mention-anomaly-based approach. *Social Network Analysis and Mining* 5, 1 (2015), 18:1–18:18.

Mahmud Hasan, Mehmet A. Orgun, and Rolf Schwitter. 2018. A survey on real-time event detection from Twitter data stream. *Journal of Information Science* 44, 4 (2018), 443–463.

Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2015. Processing social media messages in mass emergency: A survey. *ACM Computing Survey* 47, 4, Article 67 (Jun. 2015), 38 pages.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML'98).* 137–142.

Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery* 7, 4 (2003), 373–397.

Donald E. Knuth. 1997. *The Art of Computer Programming, Volume 2 (3rd ed.): Seminumerical Algorithms.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Sungjun Lee, Sangjin Lee, Kwanho Kim, and Jonghun Park. 2012. Bursty event detection from text streams for disaster management. In *Proceedings of the 21st International Conference on World Wide Web (WWW'12 Companion).* ACM, New York, NY, 679–682.

Chenliang Li, Aixin Sun, and Anwitaman Datta. 2012. Twevent: Segment-based event detection from tweets. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12), Maui, HI, October 29–November 02, 2012.* 155–164.

Jianxin Li, Zhenying Tai, Richong Zhang, Weiren Yu, and Lu Liu. 2014. Online bursty event detection from microblog. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC'14).* IEEE Computer Society, 865–870.

Michael Mathioudakis and Nick Koudas. 2010. TwitterMonitor: Trend detection over the Twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD'10).* 1155–1158.

Qiaozhu Mei and ChengXiang Zhai. 2005. Discovering evolutionary theme patterns from text: An exploration of temporal text mining. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD'05).* ACM, New York, NY, 198–207.

Fionn Murtagh. 1983. A survey of recent advances in hierarchical clustering algorithms. *Computer Journal* 26, 4 (1983), 354–359.

Ruchi Parikh and Kamalakar Karlapalem. 2013. ET: Events from tweets. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'13 Companion)*. ACM, New York, NY, 613–620.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to Twitter. In *Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 181–189.

Daniela Pohl, Abdelhamid Bouchachia, and Hermann Hellwagner. 2012. Automatic sub-event detection in emergency management using social media. In *Proceedings of the 21st International Conference on World Wide Web (WWW'12 Companion)*. ACM, New York, NY, 683–686.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2013. Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Transactions on Knowledge and Data Engineering* 25, 4 (April 2013), 919–931.

Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY.

Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. 2013. Data stream clustering: A survey. *ACM Computing Surveys* 46, 1, Article 13 (Jul. 2013), 13:1–13:31 pages.

Giovanni Stilo and Paola Velardi. 2016. Efficient temporal mining of micro-blog texts and its application to event discovery. *Data Mining and Knowledge Discovery* 30, 2 (Mar. 2016), 372–402.

Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. 2007. Mining correlated bursty topic patterns from coordinated text streams. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*. 784–793.

Yu Wang, Eugene Agichtein, and Michele Benzi. 2012. TM-LDA: Efficient online modeling of latent topic transitions in social media. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*. 123–131.

Z. Wang, L. Shou, K. Chen, G. Chen, and S. Mehrotra. 2015. On summarization and timeline generation for evolutionary tweet streams. *IEEE Transactions on Knowledge and Data Engineering* 27, 5 (2015), 1301–1315.

Andreas Weiler, Michael Grossniklaus, and Marc H. Scholl. 2017. Survey and experimental analysis of event detection techniques for Twitter. *Computer Journal* 60, 3 (2017), 329–346.

Jianshu Weng and Bu-Sung Lee. 2011. Event detection in Twitter. In *Proceedings of the 5th International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17–21, 2011*. 401–408.

Lexing Xie, Hari Sunaram, and Murray Campbell. 2007. Event mining in multimedia streams. *Proceedings of the IEEE* 96, 4 (2007), 623–647.

Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim, and Ke Wang. 2016. TopicSketch: Real-time bursty topic detection from Twitter. *IEEE Transactions on Knowledge and Data Engineering* 28, 8 (2016), 2216–2229.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'13)*. ACM, New York, NY, 1445–1456.

Jaewon Yang and Jure Leskovec. 2011. Patterns of temporal variation in online media. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM'11)*. ACM, New York, NY, 177–186.

Yiming Yang, Tom Pierce, and Jaime Carbonell. 1998. A study of retrospective and on-line event detection. In *Proceedings of the 21st ACM SIGIR (SIGIR'98)*. ACM, 28–36.

Junjie Yao, Bin Cui, Yuxin Huang, and Yanhong Zhou. 2012. Bursty event detection from collaborative tags. *World Wide Web* 15, 2 (2012), 171–195.

Jie Yin, Andrew Lampert, Mark A. Cameron, Bella Robinson, and Robert Power. 2012. Using social media to enhance emergency situation awareness. *IEEE Intelligent Systems* 27, 6 (2012), 52–59.

Yu Zhang and Zhiyi Qu. 2015. A novel method for online bursty event detection on Twitter. In *Proceedings of the 6th IEEE International Conference on Software Engineering and Service Science (ICSESS'15)*. IEEE, 284–288.

Wayne Xin Zhao, Rishan Chen, Kai Fan, Hongfei Yan, and Xiaoming Li. 2012a. A novel burst-based text representation model for scalable event detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers – Volume 2 (ACL'12)*. Association for Computational Linguistics, Stroudsburg, PA, 43–47.

Xin Zhao, Baihan Shu, Jing Jiang, Yang Song, Hongfei Yan, and Xiaoming Li. 2012b. Identifying event-related bursts via social media activities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'12)*. 1466–1477.

Received August 2018; revised March 2019; accepted May 2019