Detecting Events in Online Social Networks: Definitions, Trends and Challenges

Nikolaos Panagiotou, Ioannis Katakis(✉), and Dimitrios Gunopulos

Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens,
Panepistimioupolis, Ilisia, 15784 Athens, Greece
{n.panagiotou,katak,dg}@di.uoa.gr

Abstract. Event detection is a research area that attracted attention during the last years due to the widespread availability of social media data. The problem of event detection has been examined in multiple social media sources like Twitter, Flickr, YouTube and Facebook. The task comprises many challenges including the processing of large volumes of data and high levels of noise. In this article, we present a wide range of event detection algorithms, architectures and evaluation methodologies. In addition, we extensively discuss on available datasets, potential applications and open research issues. The main objective is to provide a compact representation of the recent developments in the field and aid the reader in understanding the main challenges tackled so far as well as identifying interesting future research directions.

Keywords: Event detection · Social media · Stream processing

1 Introduction

The Web 2.0 era brought a lot of revolutionary changes in the way World Wide Web content is generated and utilized. Social media and online Social Networks are nowadays the most widely used services along with search engines. Data generated from Web 2.0 activity are of great *value* since they reflect aspects of real-world societies. Moreover, data are *easily accessible* since they can be collected through web-crawlers or public APIs. These two qualities constitute the main motivation for researchers studying online social networks.

The range of novel data analysis applications is impressive. A prominent technique, known as 'sentiment analysis', analyzes user opinions in order to extract the expressed emotion about products [14,35,75], services, or even political figures [80]. Marketing in particular found a perfect fit since now businesses are able to analyse a large volume of public data and identify trends [55], influential profiles [18], experts [33] or to provide personalized advertisements and documents [20]. From another perspective, social scientists study knowledge cascades [32], information propagation [32] or community dynamics [49]. In health care, researchers have been able to track and predict diseases like influenza [76]

S. Michaelis et al. (Eds.): Morik Festschrift, LNAI 9580, pp. 42–84, 2016.
DOI: 10.1007/978-3-319-41706-6_2

and identify disorders such as depression [28]. The list is incomplete and expands rapidly.

From all the above applications, the task of event detection stands out due to its complexity and social impact. Broadly speaking event detection is the problem of automatically identifying *significant incidents* by analysing social media data. Such events can be a concert, an earthquake or a strike.

Most approaches tackle event detection similarly to a clustering problem. Clustering can be performed on the textual features of users' messages (Topic Clustering) or on their spatio-temporal attributes (Spatio-Temporal clustering). Some of the identified clusters correspond to real events while others are just groups of similar messages. The identification of the *event clusters* is often tackled with scoring functions or machine learning classifiers [12]. Some approaches utilize novelty tests [66] while others focus on sentiment peaks [83] and keyword bursts [1]. A common element in many methods is a change detection component necessary to identify that 'something happened out of the ordinary'. Change is detected through statistical analysis of the messages' content or the network's structure (e.g. an increasing number of new connections in the social graph). There are many more lines of research in event detection. The most prominent ones are organized and discussed in the following sections.

The purpose of the article is to provide a categorization of existing approaches in order to let the reader easily grasp the motivation, basic steps and issues of each group of algorithms. In brief, the contribution of this article can be summarized into the following points:

- It provides definitions of numerous concepts related to event detection from Web 2.0 data. These definitions aim at formalizing the problem by disambiguating fuzzy concepts. Additionally, they allow a common terminology that will aid in presenting the state of the art under the same framework.
- The state of the art is identified, organized and discussed.
- Open issues and potential future research directions are presented.

The remainder of this article is organized as follows. In Sect. 2 recently introduced definitions of event detection are presented. In Sect. 3, an overview and a taxonomy of event detection approaches is given. Section 4 presents the algorithms in more detail emphasizing on intuition, main advantages and disadvantages. Section 4 outlines architectures utilized in relevant systems for efficient event detection. After that, in Sect. 5, we review a large number of Event Detection applications. Next, in Sect. 6 we summarize the evaluation procedures (protocols, datasets, metrics) that are utilized in evaluating the algorithms and we comment on the obtained results. Finally, the paper concludes with a discussion on related problems and open issues.

2 Research Challenges and Requirements

There are numerous research challenges inherent in event detection. In this section we discuss the ones that differentiate this task from other well known

problems. Hence, we justify why off-the shelf data and text mining approaches are not suitable for tackling event detection.

Volume and Velocity. Data from social media come in great volume and velocity. Therefore, algorithms should be online and scalable in memory and computational resources. High data volume makes batch processing computationally infeasible. Data structures like Count-Sketch [19], randomized data structures such as Bloom [38] and Bloomier [50] filters, sampling methods [54] and streaming algorithms are often used in real-time streaming applications. The authors in [2] use the Count-Min Sketch [24] data structure to improve the efficiency of the Content Summary clustering algorithm they propose. Osborne et al. in [66] use a hashing function to calculate neighbours in constant time and [43] uses simple *inverse document frequency* (IDF) scores in order to avoid document-to-document comparisons and to reduce the number of computations. Most of the related work aim in building online systems capable of processing high rate streams such as the Twitter Sample stream (1 %) or even the Firehose stream (100 %) [56].

Real-Time Event Detection. Events should be identified as soon as possible, especially when the approach is intended to be used in critical applications like emergency response. In this case, methods for event detection should be evaluated not only in terms of Precision and Recall but also in terms of how fast they can identify a specific type of event. In [56], the authors offer a detailed description of the real-time elements of their approach and comment on advantages and disadvantages of making the process parallel.

Noise and Veracity. It is only natural that user generated information is characterized by noise. Social media are filled with spam messages, advertisements, bot accounts that publish large volumes of messages, hoaxes, as well as internet memes [45]. Another obstacle is that textual information in social media is very limited. Users usually publish very short messages a fact that makes off-the-shelf Text Mining and NLP methods unsuitable.

Feature Engineering. Selecting the most suitable features to utilize in supervised or unsupervised learning components is not a trivial task. Textual representations such as Term-Document matrices are not sufficient. As many researchers have observed, there are specific characteristics that appear in event related messages. These features could be content-based attributes such as TF-IDF scores, number of tags and emoticons or structural features like the number of followers (Twitter) or friends (Facebook). Supervised approaches mostly focus on content features in order to train classifiers such as Naive Bayes or Support Vector Machines. Many researchers have concluded that the utilization of the correct feature-set is very crucial for the event detection process. As an example, Becker et al. [11] presents a comparison between structural features and Term-Document matrices. They conclude that the combination of textual and non-textual features lead to a statistically significant gain in Precision.

Evaluation. Algorithm evaluation sets difficult to overcome obstacles. Unfortunately, availability of event detection datasets is very limited [57]. The TDT5[1] dataset is used by many researchers such as [43,66] in order to evaluate Precision. However, the nature of this data is quite different (topic detection and tracking) and hence it serves only as last resort. Results obtained from TDT5 could significantly vary from those obtained from Twitter or Facebook. TDT5 comes from news-wire articles and contains well formed high quality text. On the other hand, social media text has unique textual characteristics including abbreviations, use of slang language and misspellings. A public dataset gathered from social media sources is very important since it could be used to train supervised classifiers and also evaluate the algorithms in terms of Precision or Recall. Since such dataset is not available most research teams create their own corpora that are manually annotated [12,57,66] with a small number of events. This fact makes the results subjective to sample bias and also hinders comparative experiments.

3 Definitions and Context

The lack of a formal definition for the problem of event detection initiates a lot of issues since the problem is multi-dimensional and many aspects are not obvious. Up until now there were some individual efforts towards defining specific sub-problems. We begin this section by presenting such definitions of tasks that relate to event detection or similar problems. Next, we propose definitions that extend and unify the ones that appear in the literature.

According to the Topic Detection and Tracking (TDT)[2] project [3], an event is "something that happens at specific time and place with consequences". The consequences may motivate people to act in social media and hence the events will be reflected in network activity (e.g. large number of tweets on Twitter, new groups on Facebook and new videos on YouTube). Aggarwal et al. [2] provides a definition of *News Event* as "something that happens at specific time and place but is also of interest to the news media". That is, apart from making an impact to the Web 2.0 world, an event should also affect conventional news media. In [12] the authors state that "an event is a real-world occurrence e with a time period T_e and a stream of Twitter messages discussing the event during the period T_e". Their definition has a Twitter scope and is related to an increased amount of messages in a time window. However, it could be applied to other platforms that operate as a stream of documents.

McMin et al. [57] defines *event* as "something *significant* that happens at specific time and place". The authors state that something is significant when it is discussed by the news media. This is quite similar to the definition of [2]. Weng et al. in [89] state that an event is "a set of posts sharing the same topic and words within a short time". Abedelhaq et al. [1] state that events stimulate people to post messages but in a substantial geographic space. This is connected

[1] http://www.itl.nist.gov/iad/mig/tests/tdt/resources.html.
[2] http://www.itl.nist.gov/iad/mig/tests/tdt/tasks/fsd.html.

to *localized event detection* (local events versus global events). In [1] the authors define *localized events* as "events with a small spatial extent". Boettcher et al. in [15] state that an event is "an occurrence or happening restricted on time". This definition differentiates *Real World* events from *Virtual* events. Virtual events are restricted within the limits of the online world. Examples of such cases are memes, trends or popular discussions. Discussions are considered events since people are active in social media because of them. Nevertheless, they do not correspond to a real world incident. Wang et al. [87] define *Social Events* as events among people when the one is an acquaintance of the other.

A slightly different definition that also includes the concept of *social* event is given in [26]. The authors identify four event categories: *local non-social*, *local social*, *global social* and *global*. A social event is an event that involves participants that have been together again in another situation. An example of such a *social* event is a conference where the participants have attended the same conference in the past. Finally, Popescu et al. [69] describe the "event snapshots" idea as a tuple $s = (e, \Delta_t, tweets)$. The tuple consists of a set of *tweets* that are correlated with an entity e for a time period Δ_t. This definition could be mostly considered for celebrity-related events like popular actors or singers. An alternative definition that considers sentiment information, is given in [83]. The authors define the task of event detection as: "The identification of those messages that alter significantly and abruptly the emotional state of a large group of people."

It is clear that the aforementioned definitions are not always consistent with each other. Some of them require the events to happen in a specific geographical region while others do not take into account the space dimension. Other articles state that the time the events take place should be finite and short in duration. Such a definition is not applicable for Global Events that may concern communities for weeks.

Event Types. In the literature we come across the following types of events:

- *Planned:* Events with a predefined time and location (e.g. a concert).
- *Unplanned:* Events that are not planned and could happen suddenly (e.g. a strike, an earthquake).
- *Breaking News:* Events connected to breaking news that are discussed in conventional news media (e.g. the result of the elections in Greece discussed by the global press).
- *Local:* Events limited to a specific geographical location. The event impacts only this area (e.g. a minor car accident).
- *Entity Related:* Events about an entity (i.e. a new video clip of a popular singer).

Table 1 summarizes the range of the different event types in terms of space and time. It also reports in which media these events are more probable to be observed in.

The rest of this section presents definitions that unify and extend the ones proposed in the literature. They are based on the observation that *events* can be identified by analysing *actions* of *accounts* in the online social network (OSN).

Table 1. Different type of events and their properties

Event type	Time duration restrictions	Geographical distribution	Observable in
Planned	High	Medium	Social media, news media, event portals
Unplanned	Low	High	News media
Breaking news	High	Low	News media
Global	Low	Low	News media, online sources
Local	High	High	Local media, online sources
Entity	High	Low	News media, blogs

Definition 1. Account *(p): An agent that can participate (i.e. perform* actions*) in a social network after following a registration procedure.*

Accounts can be operated by individuals, groups of people or computational agents (bots). Accounts usually maintain a *profile* in the OSN.

Definition 2. Content object *(c): A textual or binary object that is published or shared via the social network (e.g. text, image, video).*

Definition 3. Action *(a): Depending on the social network, an action, a, can be either: (i) a post of new content (e.g. a new tweet), (ii) an interaction with another profile (e.g. a new follower, a friend request, etc.), (iii) an interaction with another user's content (e.g. a retweet, or a "like").*

It is obvious that some of these actions can be *observable* or *un-observable* by agents that are not connected with the action-generating accounts. In the task of event detection, we are interested in a *set of* N *actions* $A^e = \{a_i, \ldots, a_N\}$ that are correlated with (or caused by) the event e. Such a set of actions has also a temporal definition $T_{A^e} = [t_{(A^e,start)}, t_{(A^e,end)}]$. The actions that an event produces are most of the times in a different time window compared with the actual event, i.e. $t_{(e,start)} < t_{(A^e,start)}$ and $t_{(e,end)} \neq t_{(A^e,end)}$.

A^e is the ideal, ground truth set that contains *all* actions that event e has caused. This set of actions comprises the effect of the event and it is the only source of information that a computational agent can analyse in order to "sense" the event. By analysing A^e, location and actors can be inferred.

Definition 4. Event (e): *In the context of online social networks, (significant)* event *e is something that causes (a large number of)* actions *in the OSN.*

Intuitively, the importance of an event can be measured by the mass of the actions that it generates. It is implied that, in event detection, we are interested in significant events. Naturally, significant events can have global or local character.

The textual *representation* or *summary* $R(e)$ of an event could be a title heading or a set of keywords. An event is linked with a specific time frame $T_E = [t_{(e,start)}, t_{(e,end)}]$ (duration of the event). An event is sometimes correlated with a set of *involved actors* I^e and a location Loc^e.

Definition 5. Event detection in an online Social Network: *Given a stream of actions A_n of the OSN n, identify a set of real-world events and provide* some *of the following information:*

(a) the (textual) representation of the event $R(e)$,
(b) a set of actions that relate to this event $A^e \subset A_n$,
(c) a temporal definition of the set of actions

$$T_A^e = [t_{(A^e,start)}, t_{(A^e,end)}]$$

(d) a location loc_e that is correlated with the event,
(e) the involved actors I^e.

Currently, approaches presented in the literature only provide some of the above information. This is totally acceptable in some applications.

In other words, the problem of event detection could be defined as: "Given a stream of actions A_n in an online social network n identify all tuples $E = \{e_1, \ldots, e_M\}$", where M is the number of events and

$$e_i = \prec R(e_i), A^{e_i}, T_A^{e_i}, loc_{e_i}, I^{e_i} \succ$$

4 Organization of Methods

We present an organization of Detection approaches under two perspectives. We firstly organize methods according to the technique they utilize (clustering, first story, etc.) (Sect. 5). Then we organize approaches according to whether they are looking for New or Past events or whether they are operating off-line or online (Sect. 6). Details of each algorithm are presented in the following section. An earlier overview of Twitter specific event detection approaches can be found in [9]. Although most work on event detection is using Twitter data, we describe techniques on other sources as well (Youtube, Flickr, etc.). Furthermore, we provide a hierarchical organization of the methods and emphasize on architectural issues, evaluation procedures, dataset availability and dataset labeling.

4.1 Taxonomy

In this section we present a taxonomy of the related work based on the fundamental data mining techniques that they utilize (clustering, outlier detection, classification, etc.). An illustration of the taxonomy can be seen in Fig. 1. Details, as well as more references will be presented in the next section.

Most event detection algorithms tackle the problem, at least in a first stage, as a Stream *clustering* task. The identified clusters are organized into "event-clusters" or "non-event-clusters". This assignment can be either supervised or

unsupervised. In the unsupervised case, a scoring function is used that is usually based on features extracted from the clusters. In the supervised case, a classifier is trained either using textual features, structural features, or both. The advantage of the supervised approach is that the classifiers automatically "learn" the task based on historical cases. On the other hand, a training set should be available and the classifiers must be retrained periodically.

A different approach targets at the detection of *anomalies* in the content of the network. The idea is to first model the content in normal circumstances and then detect outlying messages. The first step is to build language models capturing term usage from historical data. When, for example, a group of terms demonstrates increased usage, then this is considered an indication of an event. Typically, sentiment information is utilized along with the assumption that significant deviations in sentiment indicate events.

An alternative strategy is to use novelty scores on incoming messages. Novelty scores are mostly used in the *First Story Detection* (FSD) problem. FSD is usually applied on news streams and aims at detecting the first story about an event by examining a set of 'neighbour' (i.e. similar) documents. That is, if a message is significantly different from its nearest neighbours, it is considered novel and indicative of a new event. Auxiliary sources of information like Wikipedia are exploited in order to identify evidence for the detected events.

Some methods, focus on events concerning *specific* topics such as a music band. After the messages that talk about these topics are identified, the algorithms detect anomalies. For example, in [94], authors find events about the NFL 2010–2011 games. [7] detects increases in flu-related messages while the TEDAS system [52] focuses on crime-related and disastrous events. An issue with these approaches is that the topic should be known a priori and other event types will not be identified. We refer to this category as *topic specific* event detection.

According to Topic Detection and Tracking task (TDT[3]) the two main approaches of event detection are *Document Pivot* and *Feature Pivot*. In document pivot techniques, clustering is used to organize documents according to their textual similarity and neighbours are identified through direct comparison. These approaches were mainly used in TDT challenges. However, they are not directly applicable to social media like Twitter or Facebook. The first issue is that not all documents are related to events (e.g. memes) as it is assumed in the TDT challenge. A second problem is that Document Pivot techniques require batch processing and are not scalable to large amounts of data.

Feature Pivot techniques focus on event topics that were previously unseen or growing rapidly. Many Feature Pivot techniques focus on burst detection. Bursts could be defined as term or sentiment deviations. Kleinberg et al. [44] define a finite state automaton to detect bursts in documents streams, while [31] model words as a binomial distribution in order to detect bursts. Similarly in the Twitter Monitor system [55] a streaming algorithm called Queue Burst is used in order to detect bursts on the Twitter stream. In general, Feature Pivot techniques focus on change and burst detection of text features. Most algorithms

[3] http://www.itl.nist.gov/iad/mig//tests/tdt/1998/.

from TDT that are applied in social media are mainly Feature Pivot algorithms. We will discuss Pivot algorithms in more detail in the following sections.

4.2 NED vs. RED and Online vs. Offline

Even from the TDT era (see previous section), two important categories of event detection were identified. These were the retrospective event detection (RED) and the new event detection (NED). RED focuses mostly on identifying previously unknown events from historical collections [92]. NED targets at events from live streams in real time [5]. RED mines historical data in order to detect events that were not previously known. There is no time constraint since the events already happened in the past and their identification could not support decision making. On the other hand, NED has online nature involving real time event detection with the goal to support crucial decision making (e.g. in emergency situations).

Most articles related to event detection in social media focus on online event detection. Online event detection aims at deciding if a message is about an event as soon as it arrives without the need of time consuming batch processing. The offline event detection algorithms require complex procedures that can not be used in real-time processing. They are useful mostly for retrospective event analysis where execution time is not a major requirement. Some hybrid approaches have an online part that is used for real-time analysis of messages and an offline part that post-processes data. Such an offline part may be the training of a classifier. Typical examples of online approaches are those that cluster messages and then use scoring functions to decide which clusters are event clusters. Approaches like [12] require offline components since the classifier used for the cluster categorization requires training.

5 Event Detection Methods

Following the taxonomy presented in the previous section, we present here representative methods from each category in more detail. Initially the clustering based approaches are presented including supervised and unsupervised scoring techniques. Then, approaches based on anomalies, such as keyword bursts, are discussed. After that, First Story Detection approaches inspired by TDT are presented. The Section concludes with methods that focus on detecting specific events.

5.1 Clustering Based Event Detection

Event detection approaches from social media streams is often faced using clustering of messages. After that, the clusters are classified as "Event-Related" or "Non-Event-Related" (see Fig. 1). This assignment could be resolved with supervised or unsupervised learning. In the supervised case, clusters are classified using *a learning algorithm* such as Naive Bayes or Support Vector Machine based on a

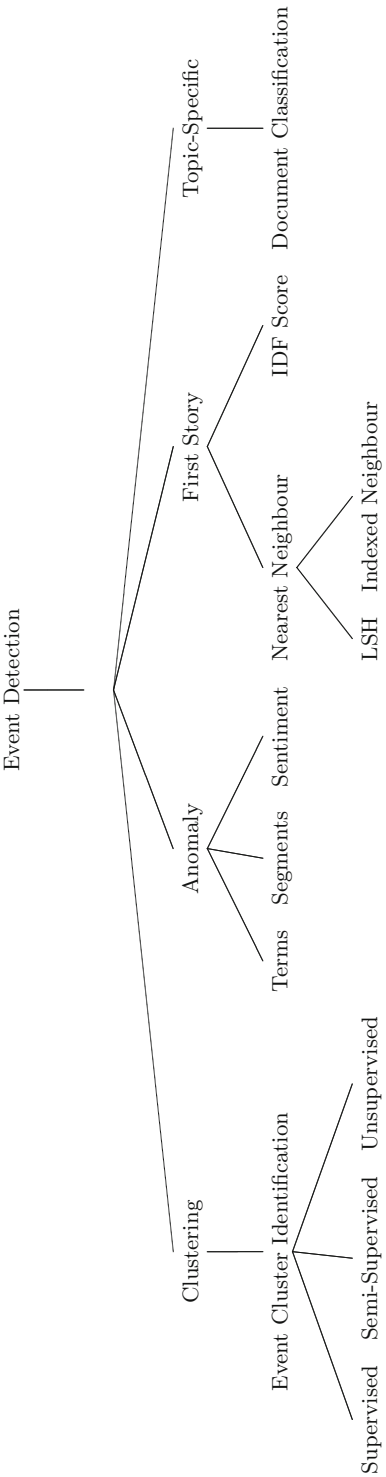

Fig. 1. An abstract taxonomy of event detection approaches.

certain group of extracted features. On the other hand, in the unsupervised case, clusters are classified according to *a scoring function*. The main difference is that in the supervised case a set of labelled clusters is required in order to train the classifier. As we will discuss in Sect. 8, labelled data sets require large amounts of annotation effort on a periodic basis due to model retraining requirements. The rest of this Section presents the supervised approaches and continues with the unsupervised. Figure 2 presents an abstract workflow of the clustering based event detection.

Unsupervised Cluster Identification. In this section methods that identify clusters as "event" or "non-event" based on a scoring function will be presented. This is the unsupervised case since no training set is required.

The first system we will present is the EvenTweet system [1]. EvenTweet is based on an initial clustering of keywords according to their spatial signature. Keywords that appear on the same location will be included into the same cluster. These keywords receive a score according to their level of burstiness, their spatial distribution and other time-related features. Burstiness is calculated according to frequency deviations from the mean. The spatial signature is calculated using geo-referenced tweets containing the keywords and it is fixed on a set of pre-defined cells on a grid. Keywords with low burstiness and high spatial entropy are filtered out as noise. Each cluster receives a score equal to the sum of its keywords' score. The top-k clusters according to their score are the candidate event Clusters. EvenTweet applies online clustering by dividing the stream into sliding windows. Windows are sub-divided into time frames. Keywords' scores are calculated per time frame. Cluster scoring is updated when a new time-frame is complete.

Similarly, in [60] the authors followed an unsupervised approach for detecting events from Twitter. Their idea is to utilize the semantic relationships of terms during the clustering procedure. They propose to cluster expanded TF-ID vectors. An expanded vector has values even for terms that are missing from the document if they are semantically related with those that are present. The cosine similarity is used as a distance metric. This approach leads to clusters of tweets that discuss the same topic. The paper presents two semantic expansion methods. The first one detects a set of co-occurring words from a static corpus and then the document vector is expanded by these co-occurrences. The second one treats each word as a vector of co-occurrences.

Using this representation, the authors calculate the cosine similarity among all vectors. Vectors having similarity more than 0.8 are assumed to be semantically related. A document vector is expanded by the semantically related terms being present at the neighbour documents. As a result, correlated words even if they do not appear in the message should have a weight in the expanded vectors. Finally, the method associates events with the largest obtained clusters.

A similar method, focused again on term vector expansion, is followed in [61]. In this case only tweets containing hashtags are utilized. Hashtag correlations are exploited in order to expand the vectors that are now solely contain hashtag

information. The results are improved in comparison to [60] especially in how fast the events are identified. Such a result is not expected since document (word) information is discarded. Nevertheless, on specific event types it turns out that hashtag features alone are sufficient.

On [47] the authors describe an event detection method that is based on topic clustering of the tweets. The clustering involves both textual and social features such as the unique users that posted messages about the event. They give a definition of *user diversity* within a cluster as the entropy of its users. The more users a cluster has the higher the diversity. Then they formally define the event detection method as an optimization problem where the goal is to both maximize the documents similarity as well as the user diversity and prove that it is NP-hard. As a result, they use an approximate time efficient and one-pass online clustering algorithm in order to cluster tweets topically. Then the clusters are periodically checked for their user diversity and those with a diversity more than a threshold are identified as event clusters.

The TwEvent system [51] implements the idea of using tweet segments (N-grams) instead of unigrams. Segment extraction is based on Wikipedia corpus and the Microsoft N-gram service[4]. Segments are selected according to their appearances on historical data using a "cohesiveness" metric formally defined on the article. Thus, only coherent segments are considered while the rest are filtered out. The segment extraction algorithm has linear complexity. In the next step, they approximate the frequencies of the segments and detect the bursty ones. Candidate event segments are identified based on burstiness and number of unique authors. Then, candidate segments are clustered using a modification of the Jarvis Patric algorithm [41]. According to this algorithm, two segments result in the same cluster if one is the nearest neighbour of the other. The similarity between two segments s_a and s_b is based on a time-indexed sliding window Wconsisting of m parts. The similarity $sim_t(s_a, s_b)$ is defined in the following Equation (Eq. 1)

$$sim_t(s_a, s_b) = \sum_{m=1}^{M} w_t(s_a, m) w_t(s_b, m) Sim(T_t(s_a, m), T_t(s_b, m)) \quad (1)$$

$Sim(T_t(s_a, m), T_t(s_b, m))$ is the similarity of the tweets concatenation containing segments s_a and s_b during the sub-window m. $T_t(s_a, m)$ is the concatenation of tweets containing the segments s_a during the sub-window m. $Sim(T_t(s_a, m), T_t(s_b, m))$ is the similarity of the concatenated documents $T_t(s_a, m)$ and $T_t(s_b, m)$ is extracted from the associated segments s_a and s_b respectively. This similarity is based on the cosine similarity of the TF-IDF vectors. Weight $w_t(s_a, m)$ equals to the ratio of tweets containing s_a to tweets that do not. A problem with this approach is that the computation complexity of the Jarvis Patric algorithm is $O(n^2)$. However, the authors state that the algorithm should perform well for a small number of tweets. A score to each cluster is assigned according to the number of segment appearances in Wikipedia. The top clusters are classified as event related clusters.

[4] http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx.

[30], similarly to [52], considers Twitter as a social sensor where the users provide valuable information for the authorities. The authors use their system to detect flood events analyzing tweets from Germany for a period of eight months. They follow a visual analytics approach in order to present flood-related Tweet messages on the map. Two interesting approaches are presented. The first one is based on increased local tweet activity. The second assumes that similar messages appearing in nearby locations can possibly refer to disastrous events taking place in that area.

For their first approach (increased spatial twitter activity), the authors divided Germany into a number of cells using Voronoi Polygons. All these cells are associated with a normal activity' characterized by a mean and a standard deviation. When increased activity is detected an event alarm is triggered.

The second approach, at first, filters out irrelevant messages (not related to floods) using a dictionary approach. The OPTICS [6] density based clustering algorithm is utilized in order to identify similar messages. Validation of clusters is achieved through the use of external sources. If the tweets contained in a cluster correspond to a news story then the cluster is identified as an event cluster. The authors conclude that the second approach is more effective than the first one.

The authors of [2] utilize content and social features of the Twitter network in order to detect events. Following a similar path to many of the aforementioned approaches, clustering comprises the first step of the method. Clustering is topic based since textual features are analyzed. However, an important difference with other approaches is that the clustering takes into account user profiles. The algorithm is named Cluster Summary (CS). Cluster centroids consist of two parts: (i) the *content* summary which is a term-frequency matrix and (ii) the *user* summary which is a user-frequency matrix. The distance metric of the clustering algorithm is a linear combination of the two summaries:

$$Sim(S_i, C_i) = \lambda * SimS(S_i, C_i) + (1 - \lambda) * SimC(S_i, C_i) \tag{2}$$

Content similarity $SimC$ is based on TF-IDF [21] and utilizes the cosine distance. The structure similarity $SimS$ depends on how many users the tweet and the cluster have in common. The authors associate each tweet with all the followers of the author of the tweet.

In the same article, the authors state that the user summary of a cluster can be represented using a randomized counting data structure called Count-Min Sketch [23]. The Count-Min Sketch could be used to approximate the user frequency using constant amounts of memory. Count-Min Sketch is a data structure that overestimates the counters of an element . A possible drawback of the Count-Min Sketch for such applications is that its error rate increases with time [23].

Once a cluster is formed then its size is periodically checked according to its recent history. If the cluster growth over two consecutive sliding windows is more than a predefined threshold the cluster is identified as an event.

Supervised Cluster Identification. In this group of approaches, decision about event clusters is made through supervised machine learning classifiers. The classifiers take advantage of textual features as well as other attributes that are usually domain dependent.

In [86] the authors initially clustered the tweets according to their spatio-temporal information using a set of predefined rules. The clustering algorithm presented is simple, online and fast. The clusters formed are considered event-candidate clusters. In the next step, textual and non-textual features are extracted from the clusters in order to train a classifier. The top extracted features according to individual feature evaluation are the following:

- Unique authors
- Word overlap
- Number of mentions
- Unique coordinates
- Number of fourthsquare[5] posts

For example, the more unique coordinates or fourthsquare posts within a cluster, the more likely this is an event cluster. Three classifiers are trained using a manually annotated dataset: a Decision Tree, a Naive Bayes classifier and a Multilayer Perceptron. The classifiers are compared using only textual features against using both textual and non-textual features. The result showed a statistically significant improvement when all features are used. This is an indication that in some cases the text itself is not enough.

Another algorithm based on message clustering followed by supervised classification is described in [12]. The authors, as in similar approaches, set the requirement of not knowing apriori the number of clusters and they use an online threshold-based clustering method. The documents are presented as vectors that are TF-IDF weighted using a bag-of-words approach. The clustering algorithm is simple as in [86]. When a new point has a distance less than a threshold from the nearest centroid it is added to that cluster, otherwise a new cluster is created. Then, features are extracted from the clusters in order to train the classifier. The features are topical, temporal, social and Twitter specific. Temporal features may describe deviations on the volume of common terms as well as changes on their usage frequency. Social features capture interactions among users. Topic based features capture the thematic coherence of the cluster. Twitter specific features are often present in non-event clusters, for example Twitter tagged conversations that do not correspond to a real-world event (e.g. the hashtag #ff "Follow Friday"). When the feature extraction is complete, a Support Vector Machine is trained and compared against a Naive Bayes classifier using only textual features. The conclusion is that the manually extracted features provide a very important advantage over the baseline method using only text features. The clustering step is fast and online. However, training the Support Vector Machine is computationally demanding involving parameter tuning and is prone to over-fitting especially when the training set is relatively small.

[5] https://foursquare.com/.

The event detection method presented above, originally described in [12], is revisited in [70] in order to boost the clustering procedure. The authors state that the original online clustering algorithm takes into account just textual information in the form of TF-IDF vectors. They then propose the usage of two new features from the similarity function used in the clustering. The first feature originates from the parsing of URLs within the document, favoring documents with the same or similar URL. The intuition behind that is the fact that documents that contain the same URL should result into the same cluster since the shared URL indicate that the documents correspond to the same event. The second feature is called "Bursty Vocabulary". This is a set of keywords per cluster that exhibit bursty behaviour identified trough a computationally inexpensive outlier test on consecutive sliding windows. The frequency of the identified bursty keywords during the next sliding window is estimated and used for the assignment of new documents to the cluster. The two features presented so far are textual. However, the authors suggested that temporal features should be used also during the document allocation to clusters. They propose the usage of a Gaussian attenuator, highly similar to the one presented in [73], that takes into account the time of the latest cluster document and the time of the document to be assigned. This temporal feature is embedded to the clustering similarity function and penalizes inactive clusters. The above improvements not only supply more textual information to the clustering algorithm but also exploit temporal information resulting into a textual-temporal method.

Another interesting system is described in [73]. TwitterStand targets at detecting tweets that relate to *Breaking News*. In contrast with other approaches, it does not utilize information extracted only from the Twitter API. Their data originate from tweets of the top-2000 users with the most tweets, the 10 % of the public tweets, the Twitter Search API[6] and an API named BirdDog that receives tweets from a large number of Twitter users.

A Naive Bayes classifier is built in order to classify tweets as "news-tweets" or "junk-tweets". Their approach of classifying tweets obtained from keyword based searches is similar to [52]. The classifier is trained on a static corpus consisting of tweets labelled as "junk" or "news". A smaller dynamic corpus is exploited to periodically update the classifier. This corpus consists of tweets related to news reported by conventional media. "News" tweets are clustered into topics. The clustering algorithm used is called leader-follower [29] and allows content and temporal clustering. Regarding content similarity required by the clustering algorithm, the TF-IDF weighted vectors of the tweets are utilized. The similarity metric is a variant of the cosine similarity containing a temporal factor. Content similarity between document d and cluster c is defined as:

$$\delta(d, c) = \frac{TFV_t \cdot TFV_c}{||TFV_t|| \cdot ||TFV_c||} \tag{3}$$

where TFV_t and TFV_c are the term vector of the tweet (TF-IDF weighted) and the cluster centroid respectively. In order to capture time the similarity metric

[6] https://dev.twitter.com/docs/api/1.1/get/search/tweets.

is expanded with a Gaussian attenuator (Eq. 4). T_t and T_C are the tweet time and the time of the latest tweet respectively.When the clustering is complete the system presents the clusters on a map by estimating the location of tweets using a text-based geo-tagging technique.

$$\hat{\delta}(d,c) = \delta(d,c) * e^{\frac{-(T_t - T_c)}{2\sigma^2}} \tag{4}$$

EventRadar [15] follows a similar idea in order to detect localized events. A term vector is created for each tweet at pre-processing. Then, unigrams, bigrams and trigrams are extracted and the algorithm examines if in a recent history H there are tweets that contain these n-grams. If these tweets are close in space and time, they are considered as event candidates. DBSCAN is utilized for clustering the tweets. A Logistic regression classifier is trained in order to reveal which of the clusters are real events. The features used for the classification task include two Boolean variables related to the tweets locations and keywords. In addition, they include the number of tweets containing relevant keywords for a period of seven days as a feature for the classification. The final result is a list of events described by a set of keywords as well from a set of representative tweets.

The work in [69] aims at recognizing controversial events. These are events where users express opposing opinions. The authors use the idea of an *entity snapshot* as the sum of the *tweets* related to an entity e published during a period Δ_t. It is defined as a triple $s = (e, \Delta_t, tweets)$. Some of these snapshots are about real-events related to an entity e and are called *event snapshots*. Snapshots are created from entities gathered from Wikipedia, tweets referring directly (using the @ symbol) or indirectly to these entities published during Δ_t. The next step of the method is to train a Gradient Boosted Decision Tree to classify snapshots as event or non-event snapshots. Features from Twitter as well as from external sources, including sentiment information, are utilized. For each of the detected events a regression model outputs a controversy score. These models mainly use textual features extracted from annotated samples as well as from a sentiment lexicon and a controversy lexicon derived from Opinion Finder[7] and Wikipedia pages.

Semi-supervised Cluster Identification. In [39] a semi-supervised approach is utilized since the labelling of Tweet clusters, involving thousands of tweets, is a quite time consuming task. The system tracks events in news media, extract keywords, and labels tweets that contain similar keywords as 'events tweets'. These tweets are able to propagate their label to related tweets using the social structure of Twitter. Social ties such as message re-tweets, mentions and hashtags are used to propagate labels. Using a reputable seed the authors are able to obtain a training set and propagate the observed labels. The next step is to construct the wavelet signal for every term that appears in the tweets. For these signals auto-correlation is calculated and common words or words appearing every day are filtered out (e.g. high auto-correlation). For example the hashtag

[7] http://mpqa.cs.pitt.edu/opinionfinder/.

"#ff" (follow Friday) appearing every Friday would be filtered out. For the resulting set of words a cross-correlation matrix is calculated. This matrix is presented as a graph in order to apply graph partitioning [89]. This way word groups are created and tweets are clustered to these word groups according to their content. For the classification of the clusters as event or non-event, a Support Vector Machine is trained using TF-IDF weighted document term features. Named entities are removed from the terms in order to avoid over-fitting issues. The event-clusters identified are then spatially grouped according to the tweets' geo-locations. For tweets that do not contain geo-location, the location is propagated from related tweets using social ties. In the end, the system provides a visualization of the event clusters on a map. We should note that such an approach could be highly valuable in cases where a reputable seed is available allowing label propagation without the need of manual annotation.

Clustering Approaches Summary. Table 2 presents an overview of the clustering-based approaches used for event detection. The second column notes the clustering algorithm that is utilized in the approach. The third and forth column present the features and the similarity metric that are exploited. Lastly, "scoring" indicates how the event cluster identification is achieved (supervised or unsupervised). Table 3 presents an overview of the supervised approaches, along with features and classifiers used.

Table 2. A summary of the clustering approaches used for event detection.

References	Clustering type	Features	Similarity metric	Scoring
[1]	Keyword clustering	Spatial	Cosine similarity	Unsupervised
[52]	Topic	Segments	Content similarity	Unsupervised
[60,61]	Tweet clustering	Expanded TF-IDF vectors	Cosine similarity	Unsupervised
[86]	Spatio-temporal	Spatio-temporal	Rule based distance	Unsupervised
[12]	Topic	TF-IDF vectors	Cosine similarity	Supervised
[30]	Spatio-temporal density	Spatio-temporal	Distance threshold	Unsupervised
[73]	Content-temporal	TF-IDF vectors, Temporal	Modified cosine similarity	Unsupervised
[39]	Topic clustering	Term vectors	Overlapping tweet terms	Supervised
[39]	Term clusters, spatial clusters	TF-IDF vectors	Cross-correlation	Semi-Supervised

5.2 Anomaly Based Event Detection

The methods of this section follow the path of identifying *abnormal* observations. Examples include: unexpected word usage in the last time window, irregular

Table 3. A summary of the supervised approaches used for event detection.

References	Features	Algorithms
[12]	Temporal, topical, social	Naive Bayes, SVM
[86]	Textual, spatial, temporal	Decision Tree, Neural Net
[73]	Textual	Naive Bayes
[15]	Textual	Logistic Regression
[69]	Social (internal and external), textual	Gradient Boosted Decision Tree

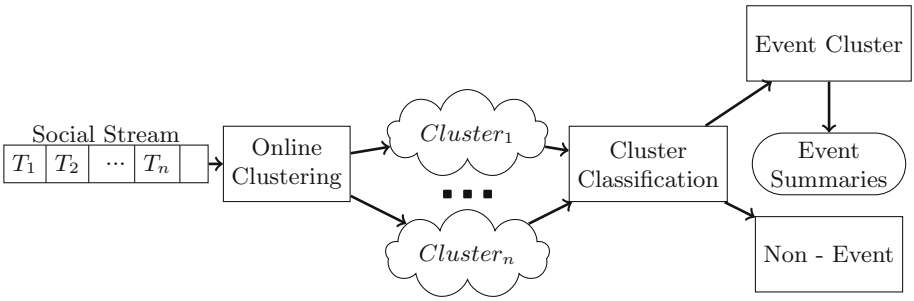

Fig. 2. An example of the general event detection approach using stream clustering. The online clustering component groups tweets that are close in space. The Cluster Classification module uses a supervised or an unsupervised method in order to classify the clusters as event cluster. For each of the event clusters a summary is extracted using different summarization methods.

spatial activity, or a distribution of emotion that is different from the average. The approaches discussed in this section track the social stream and raise an alert for an event candidate when an anomaly is observed.

The methods presented in [82,83] focus on identifying events from social media using a sentiment analysis. The main idea is that users will respond to an event in order to express their opinion causing this way fluctuations in the sentiment levels. According to the authors, when an event happens it affects the emotional state of a group of people that are close to the event. In the proposed system, users are initially clustered according to their geographic locations and their messages are aggregated over sliding windows. For each sliding window, sentiment sensors are responsible for specific regions (e.g. a sensor per city or district). The emotions of each region are analysed over four sentiment classes. When a significant deviation in sentiment levels is detected, an event alarm is triggered. The system is compared against the EdCow [89] system that uses keyword count deviations instead of sentiment information. Authors of [83] report that TwitInsight outperforms EdCow on its capability to detect events. In addition, TwitInsight is much faster since it does not depend on computationally expensive procedures and is able to run in real-time.

A strong point of the above work is the sentiment level outlier detection method. The authors of [83] assume that the sentiment level distribution is *unknown amd changes* overtime. The idea is to dynamically estimate the *Probability Density Function* (PDF) in a streaming fashion and use it for detecting outlying sentiment levels. The tool used for this methodology originates from [78] where a streaming estimation technique for unknown Probability Density Functions (PDF) is utilized based on kernels and dynamic sampling. The resulting PDF is used in order to perform non-parametric density outlier detection.

Authors in [22] use the Discrete Wavelet Transformation in order to detect peaks on hashtag usage in Twitter that will point on real-world events. This approach is similar to [89] with the difference that only hashtags are used while the rest of the text is discarded. The approach utilizes Map-Reduce jobs to extract hashtags and create their time series. Time series consist of aggregated counts of tweets containing the hashtag over five minute time intervals. Discrete Wavelet Transformation is used in order to detect *bursts* of hashtags since that could indicate events. The events are summarized using a fast online version of Latent Dirichlet Allocation (LDA) based on Gibb's Sampling. Topic modelling is used in order to represent events as a mixture of latent topics. This approach did not focus on real-time event detection but rather on batch data analysis.

Outlier tests that consider hashtags similarly to [22] are presented in [25,46]. Hashtags are commonly used to indicate *topics* but some times correspond to real *breaking news events*. Moreover, in some cases, they represent "memes" or "virtual events". The authors in [25] extracted content features from hashtag including "frequency instability", "meme characteristics" and "authors entropy". They classified hashtags as "Advertisements", "Miscalculation", "Breaking News" and "Memes". Their method is able to discriminate breaking news from meme-hashtags regardless of language. Similarly in [46], hashtags are considered to be associated with event or with memes. The authors extracted hashtag features like the number of words used with a hashtags, the number of replies a tweet with a hashtag is getting, number of URLs, and more. Using these features and a training-set they utilize a set of supervised classifiers including Random Forests and Support Vector Machines. According to their report, discrimination between event-hashtags and meme-hashtags is successful with 89.2 % accuracy.

Watanabe et al. [88] built the Jasmine system in order to detect *local* events in real time. They used the streaming Twitter API to collect tweets from Japan. A location database is created from messages posted on Forthsquare[8]. This database is utilized in order to geo-tag tweets not including location information. The approach is simple and fast. Based on geo-tagged tweets, popular places are identified using a hashing algorithm called "geo-hash". According to this algorithm, close locations result into the same hashing bucket. From the most popular places, keywords that describe the localized event are extracted. Jasmine could lead to an interesting mobile application that detects local parties or concerts instead of larger scale events like earthquakes.

[8] https://foursquare.com/.

Focusing on a different social network, in [85] the authors describe their approach for the MediaEval Benchmark 2012[9]. This dataset contained 167 thousands images from Flickr and the challenge is to find (a) technology events that took place in Germany, (b) soccer events that took place in Hamburg and (c) demonstrations and protests that took place in Madrid. The research team used some preprocessing techniques involving removal of common words and text cleaning. They also used the Google Translate API[10] in order to translate non-English words. Based on the image description text, they classified pictures based on their TF-IDF vectors. As for the pictures with no textual information, user profile information is utilized. Since the challenge required topic-specific event detection, Latent Dirichlet Allocation is utilized in order to extract topics. For the event detection task they used peak detection on the number of photos assigned to each topic. If a topic received more photos than expected, an event is identified for this topic. Since this approach requires computationally expensive procedures such as LDA, it is not easily applicable for high rate streams.

5.3 First Story Detection

The authors of [66] tackle the problem of detecting the first story about a news event. This problem is known as *first story detection (FSD)* and is equivalent to the problem of new event detection (NED) (see Sect. 4). A common approach to solving the FSD problem is to calculate for every document in the corpus the distance to their nearest neighbours [5]. If this distance is larger than a threshold, this document is considered novel and a "First Story". This unsupervised approach is extended in [48] to a supervised method using as features the distance, the entity overlap as well as the term overlap utilizing a SVM classifier.

Osborne et al. in [66] suggests that the conventional approach described by [5,48] will not scale for streaming data and therefore proposes a more efficient approach. Nearest neighbour calculation is computationally intensive. Even fast nearest neighbour algorithms such as KD-Trees and Indexing-Trees [93] will not scale in the case of large and fast social streams. The authors propose the usage of a hashing technique in order to detect the nearest neighbour. It is called *Locality Sensitive Hashing (LSH)* [77] and is a hashing scheme that provides an approximate nearest neighbour in constant time. LSH hashes documents to buckets. If the documents are similar, they are hashed into the same bucket. However, the approach is randomized and errors may occur. In order to reduce the variance of the neighbour errors, [66] uses multiple LSH data structures. That is, a document is hashed to multiple buckets, one per LSH data structure, using different hashing families. The neighbour computation similarly involves the exploration of all these buckets.

In order to bound the memory requirements as well as the number of computations per incoming document they restrict the maximum size of a LSH bucket and the computations performed within it by discarding old invaluable documents. Similar approaches are used in [53] in order to bound the memory

[9] http://www.multimediaeval.org/mediaeval2012/.
[10] https://cloud.google.com/translate/docs.

consumption using careful delete operations. This system detects the First Story about an event and then new documents that are similar are linked together in order to create *event threads*. Threads are document sets that represent the event. This is similar to the clustering approaches we discussed in the previous sections. The Threads are created and then presented in a sorted list according to their size, number of users and thread entropy (expressed as the distribution of terms). They compare their results to [5] and they suggest that event detection performance is similar. However, the efficiency of [66] is improved from [5] achieving constant memory and computation time per document resulting in a streaming FSD solution.

Petrovic et al. in [59] observed that the FSD system of [66] had low Precision due to many false positives. This is something expected since most tweets are not about real-world events. The authors use two streams in order to detect events. They constructed a stream of Wikipedia page views using Wikipedia logs[11]. The method in [66] is ranking the event threads according to their entropy in order to identify the top-k events. In [59] the ranking is modified in order to take into account Wikipedia page views that are related to the event. For every detected thread, the Wikipedia stream is checked for outlying behaviour (i.e. an unexpectedly large numbers of views) in pages that had a similar title. A very important drawback of this method is that Wikipedia Stream lags on average two hours behind the Twitter Stream causing problems for real time event detection. This is explained by the fact that users initially discuss the event topic on the social platform and then some of them may visit the relevant Wikipedia page. An overview of the system is presented in Fig. 3.

Osborne et al. [67] extends [66] in order to cope with "tweet paraphrases". That is, the feature vector is extended with synonyms of existing terms. This approach may remind to the reader the method described in [60] where the term vectors are semantically expanded. In [67] the authors used online sources in order to create a list of paraphrases while in [60] the authors computed term co-occurrences from a static Twitter corpus. The idea of [67] is to use a term-to-term matrix Q in order to exploit term synonyms. Using this matrix the similarity of

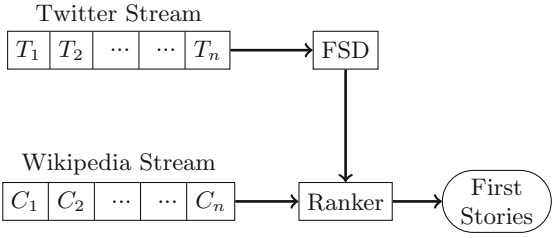

Fig. 3. Overview of the approach presented in [59]. A Wikipedia page-view stream is utilized in order to validate events detected by the FSD algorithms

[11] http://meta.wikimedia.org/wiki/Data_dumps#Content.

two tweet vectors x and y, is computed as:

$$Sim(x, y) = y^T Q x \tag{5}$$

Since such a similarity computation requires intensive matrix-vector multiplications they use a heuristic to calculate the inner product faster by using the square root matrix of Q. Since the square root matrix computation is also expensive $O(n^3)$ they approximate $Q^{\frac{1}{2}}$ as if Q is a sparse matrix. Using the heuristics their system is only 3.5 times slower from the original FSD system [66]. The improvement in the Precision of the system is significant achieving higher Precision than the state of the art UMASS system [4].

A similar approach that uses Locality Sensitive Hashing for event detection is followed in [42]. The approach is quite similar to [66]. The authors utilize Twitter posts and Facebook messages. They use LSH in order to group messages into buckets. Their algorithm works in two phases. In the first one, new events (first stories) from both sources (Twitter, Facebook) are independently identified and stored. In the second phase, first stories are hashed into buckets and the corresponding messages are stored as 'event messages'.

The authors in [65] applied an FSD system on Twitter and on a news feed that is referred as Newswire. The performance of FSD is evaluated on both Twitter and the newswire. In an additional experiment, the 27 events originally detected in [66] are used in this work in order to clarify which of the events will be present in both media. They found that almost all events appeared on Twitter and newsWire. However, the events appeared in different time points in the two streams. Events related to sports usually appear faster on Twitter since users post about them while they happen. On the other hand, on events related to breaking news, the newswire stream had a minor advantage.

An efficient first story detection method is presented in [43]. The authors focus on new event detection using a novelty score that is based on term-usage. The main goal of this approach is to detect novel documents avoiding the computation of distances among similar documents. Such an approach is very useful since neighbor computations in the TF-IDF weighted vector space could be computationally intensive for a large corpus. The proposed algorithm uses the Inverse Document Frequency (IDF) per keyword as a novelty score component. Each document receives a novelty score that is the sum of its terms' IDF weights. That is, a document is considered novel if its terms are novel. If the novelty score is above a threshold the document is detected as event related. In the same work, the authors suggest also the probabilistic IDF (pIDF) as a scoring function. Given a term q taken from a corpus C, pIDF is defined as:

$$pIDF(q, C) = log \frac{N - df_q}{df_q} \tag{6}$$

df_q is the frequency of the term q among the documents in the corpus C and N is the size of the corpus C. The probabilistic IDF violates a set of rules about a scoring function since it can take negative values. The authors state that this is beneficial since it penalizes documents if they contain common terms. Notably,

score calculation using IDF weights [43], is invariant of the corpus size and the complexity of processing a document d is $O(|d|)$. Where $|d|$ is the number of terms used in the document. However, it should be noted that since social media such as Twitter consist of extremely dynamic content, the IDF scores should be periodically updated in order to reflect accurately the content of the stream.

5.4 Topic Specific Event Detection

The methods presented so far aim at identifying events that could be of any type. This section, presents algorithms that target at identifying and tracking events of a specific predefined type.

One of these efforts is the TEDAS system that is described by Li et al. in [52]. TEDAS is built for recognizing criminal and disastrous events such as tornadoes, floods or law-breaking evidence. The system collects tweets using the Twitter API and returns tweets related to crime and disaster using topic related keywords. Tweets are captured using an initial keyword seed that is predefined by the authors. This seed is expanded according to co-occurrences with keywords from the received tweets. In other words, the system looks for paraphrases or for semantically linked terms. Since not all tweets containing these keywords are about crimes and disasters, a classifier is trained using Twitter features such as use of hashtags, mentions and some predefined pattern-features in the content. Such a pattern feature is the presence of time and location in a tweet. The system clusters all crime and disaster (CDE) tweets according to their spatial information and presents them to a map. The high-level description of the TEDAS system can be seen in Fig. 4.

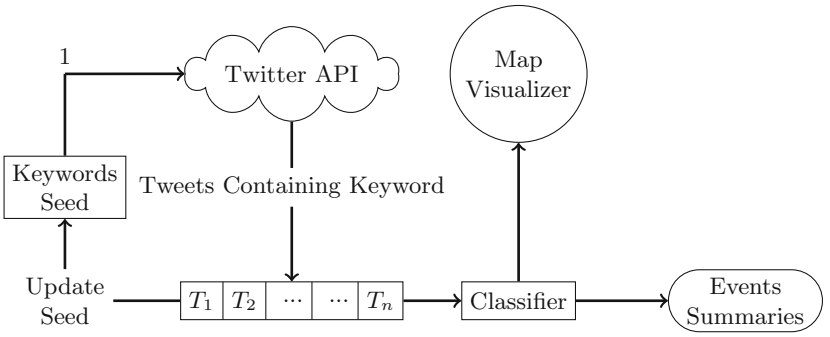

Fig. 4. The TEDAS system. An initial seed with crime/disaster keywords in applied to the Twitter API in order to collect relevant tweets. Based on the result set, the keyword list is expanded. Each tweet is classified as event or non-event, and in the second case it is presented to the map.

Another approach that targets events of specific type is that of Sakaki et al. [72]. This work focuses on earthquakes. Initially, tweets containing at least

one earthquake-related query-term are collected. Assuming that these tweets actually talk about earthquakes the authors train a Support Vector Machine on these data. For the classification task, they used tweet statistics (#words in a tweet), textual features (terms of a tweet) and context features (keywords before or after the query term). The goal is to detect the location and the trajectory of the event. The geo-tagging of tweets is exploited to detect the location of the event. Finally, Kalman Filters and Particle Filters aid in identifying the trajectory of the event.

Packer et al. [62] expand a seed of keywords related to a topic using external structured information. In order to collect tweets about a topic (e.g. a music band), they use RDF structured information to identify related entities. For example, many music bands have an entry in DBPedia[12], a large RDF database based on Wikipedia. Therefore, entities related to the band (e.g. band members) could easily be extracted. These additional entities are used to extract tweets that refer to the topic. Events related to a topic are identified according to the number of the times the corresponding entities are mentioned in tweets. According to the experimental evaluation the usage of the external sources gave a boost in the event detection performance. Furthermore, the authors observed an important correlation between the actual time period of the event and the time the related entities are mentioned in Twitter. This observation suggests that users usually tweet during an event.

The work presented in [94] focuses on detecting sports-related events. The case study is the National Football League games of the 2010–2011 season. Using a lexicon-based heuristic the authors collect relevant tweets. For identifying events, they propose a sliding adaptive window-based method. If the ratio of relevant tweets in the second half of the window is larger than a predefined threshold then that is an indicator that something is happening. The window size is adapted when the tweet-ratio(of relevant tweets) highly deviates from that of the previous window. The algorithm is able to detect game related events such as *touchdowns* and *interceptions*. The idea of using an adaptive sliding window is quite interesting since it will enable capturing events of different magnitudes.

The approach proposed in [7] targets at detecting influenza incidents using Twitter. Similarly with above, flu-related keywords are utilized in order to collect a number of potentially relevant tweets. A classifier is then trained in order to filter tweets that are not relevant. The classifier is built on bag-of-words text features. By considering the output of the classifier (relevant tweets), a time series is created based on flu-related tweets count. In order to evaluate their results, the authors compared their methods to Infection Disease Surveillance reports from clinics and to Google trends[13]. They conclude that this type of events can be tracked via Twitter and detected before Google trends and even before a potential break out. A similar approach is suggested in [34] where search engine queries are utilized instead of social media messages.

[12] http://dbpedia.org/About.
[13] http://www.google.com/trends/.

Medvet et al. [58] focus on keywords that suddenly received increased popularity. Their algorithm monitors words related to a predefined topic. Whenever these words demonstrate an increased frequency - compared with their history, are identified as candidate event keywords. The most recent tweets containing these terms are classified according to sentiment (positive, negative or neutral). Tweets from these three classes are then used to generate a summary for the event. This application could be very useful for brand related events and also for market research software in order to track product feedback.

6 Architecture

From a computational point of view, an apparent obstacle in social media analysis is Big Data management. Real time detection in Web 2.0 data requires algorithms that efficiently scale in space and time. Extreme data volumes impel researchers and engineers to consider distributed environments. Inevitably many of the papers discussed in Sect. 5 focus on architectural aspects and suggest frameworks suitable for real-time social media analysis. Methods that are focused on smaller data volumes without intensive computations however are able to perform in real-time even with the usage of a single machine.

The frameworks proposed in the literature recently can be organized in the following categories:

- *Multi-Component: Single Machine or Distributed.* The system consists of many components, each of which is responsible for a different task. Many times, the components that considered to be a bottleneck, are replicated on multiple machines in order to increase throughput if possible.
- *Data Stream Topologies:* Multiple nodes are responsible for different tasks. These approaches utilize a stream topology that is suitable for scaling with high-rate data input. A common configuration for this case is Storm[14] along with a NoSQL database like MongoDB.

6.1 Architectures of Multiple Components

In this section we provide an overview of systems built for event detection that utilize a multiple components structure running on a single or multiple computers.

In [86] the authors focus on identifying events in *real-time*. They utilize a sample of the Twitter stream that produces 3 million tweets per day. The core of the system is a MongoDB[15] database. MongoDB is suitable for storing data such as tweets in JSON format. The choice of this type of database is supported by the fact that MongoDB supports geo-spatial and temporal indices. Another important feature is that it can easily scale in number of instances. In case data rate increases (e.g. due to Twitter increased popularity) MongoDB would deploy

[14] http://storm.incubator.apache.org/.
[15] https://www.mongodb.org/.

additional machines. The system consists of multiple components. These are a Twitter Fetcher, a Cluster Creator, a Cluster Updater and a Cluster Scorer. Initially a single machine is used, however modules such as the Cluster Scorer could be easily replicated on more than one machine to handle increased load. This architecture is presented in Fig. 5.

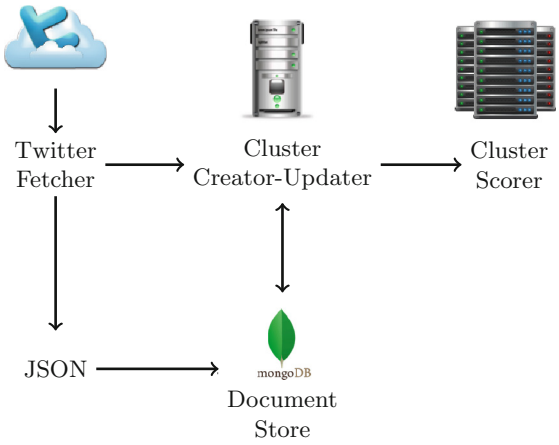

Fig. 5. The architecture of [86]. The four components are the Twitter Fetcher, the Cluster Creator-Updater and the Cluster Scorer. The cluster scorer is represented as a set of servers since it could be replicated.

In [22] the author established a MongoDB database to store Twitter data. The choice of MongoDB is justified by the requirement of having the pre-processing done by Map-Reduce jobs. MongoDB supports Map-Reduce jobs as javascript functions. Map-Reduce as a pre-processing engine is an intuitive choice given that tasks like noise filtering and natural language processing are computationally demanding. The dataset consisted of 1.7 million tweets per day. However, the method did not target at run-time processing since it only considered Retrospective event detection. The dataset is processed in batch steps that involved computationally expensive procedures like Discrete Wavelet Transformation (DWT) and Latent Dirichlet Allocation (LDA). The main contribution of this work is that it provides an insight on the capabilities of map-reduce for efficiently preprocessing textual information. MongoDB as well as other NoSQL databases such as CouchDB attracted the interest of the research community due to their document storage and scaling capabilities.

Abdelhaq et al. [1] focused on identifying local events in a streaming fashion. The system is implemented as a plug-in for the JOSM[16] framework. Similarly to [86] it consists of a number of modules. The Tweets Repository module is responsible for gathering tweets using the Twitter API. The Buffer module keeps in

[16] https://josm.openstreetmap.de/.

main memory the last window of tweets. The window is indexed according to time frames in order to ensure quick access to the data. In addition, this module maintains a table with word-count statistics calculated from the streams' history. This component results in large amounts of memory requirements. Sketch randomized data structures could be a solution in such cases and be applied for keeping the word-count table in main memory. The Content Processor module is responsible for the data processing task. This component could be distributed on more machines since its operations are easily parallelized. The last module is the Localized Event Detector and this is the component that actually performs the event detection and it is triggered at predefined times.

The Jasmine system that is presented in [88] uses a large sample of Twitter (15 % of the original stream) that leads to a stream of 15 million tweets per day. The basic components are: (a) the Tweet Fetcher, that is responsible for downloading tweets, (b) the Geotag allocator that is responsible for assigning locations to tweets - this module takes advantage of a locations database where places are stored using the Solr[17] search engine for efficient text search, (c) the Popular Place Extractor that keeps a list of the most popular places, and finally, (d) the Key Term extractor that identifies the most popular words in tweets in order to summarize the events. The system is able to run in real-time but also maintains a history of the stream in order to support retrospective event detection or any other type of post processing.

An overview of the system is presented in Fig. 6.

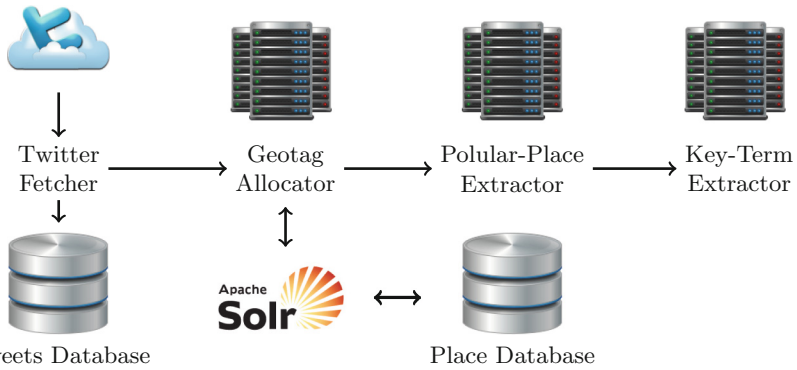

Fig. 6. The architecture of Jasmine System [88]. The first component is the Twitter Fetcher that receives tweets from the 15 % of the Twitter stream and stores them in a database. The geotag allocator geotag the tweets using the places database. The popular place component keeps in memory the most popular places. Finally, the Key-Term extractor extracts key-terms for localized events.

The TwitterStand system presented in [73] used 4 different sources of information: (a) Twitter Gardenhose (deprecated privileged Twitter API providing

[17] https://lucene.apache.org/solr/.

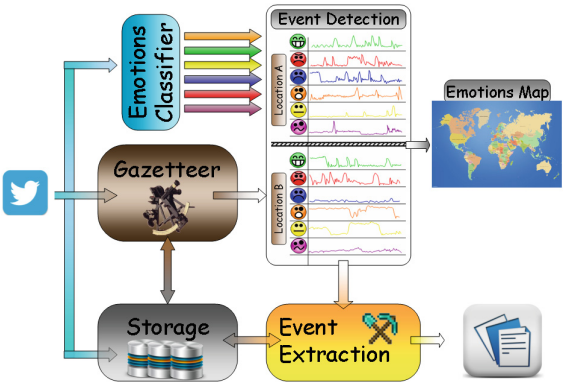

Fig. 7. The architecture of the TwInsight system [84].

10 % sample of tweets), (b) the BirdDog service (deprecated API for receiving posts from up to 200,000 Twitter users), (c) a 2000 user stream and (d) a keyword stream of 2000 terms. The first component of the system is responsible for collecting the tweets from the sources. The next component is a fast Naive Bayes classifier with the purpose to filter out "junk" from "news" tweets. The classifier is not trained on a static corpus but it is updated from a dynamic one. This justifies the choice of the Naive Bayes classifiers since it has a low update computational cost. Another benefit is that its simplicity makes it tolerant to increased data volumes. The third component is the Clusterer that performs topic clustering using textual features. The Clusterer depends on the classifier component since it clusters only the tweets that are classified as "news". The last module is the Geo-Tagging Component that groups together the topic-clustered tweets according to their geo-location. The above system is represented as a graph of connected stream processing units, each of them receiving the output of the previous one, defining a processing topology. Thus, it is straightforward to think that it could be implemented from a stream processing framework such as Apache Storm[18] and distributed on multiple processing engines if necessary. Units that act as bottlenecks can be enhanced with more cores.

Another system structured in multiple-components is presented in [84]. Two approaches are presented. The first one uses Twitter data while the second one exploits mobile information. Figure 7 presents the architecture of the approach operating on Twitter data. This system is utilized by the TwInsight system which is presented in [83]. The first layer consists of the Twitter feed fetcher. In the second layer, an emotions classifier, a Gazetteer and a storage component are included. The Gazetteer assigns geo-locations to tweets and users utilizing an algorithm presented in [81]. This part can be a bottleneck for the system and therefore can be replicated to multiple machines. Emotions are assigned to tweets using a Machine Learning classifier. The storage component contains a database that stores the tweets with their extracted meta-data from the previous

[18] https://storm.apache.org/.

two components including emotion as well as location information. Finally, the resulting tweets are processed in the third layer by the event extractor. This component performs the event detection and provides a summarization of the detected events. A visualization component on the third layer is responsible for providing sentiment level information on a map.

The purpose of INSIGHT's[19] Twitter Intelligent Sensor Agent (ISA) is to detect in real-time traffic or flood related incidents in the city of Dublin. The architecture of the Twitter-ISA consists of multiple components similar to [83]. The first component is a Tweet fetcher responsible for gathering topic related tweets through the Twitter Filtered API[20]. This enables tracking of specific users, keywords and locations in order to collect a decent number of topic related tweets. Since the majority of tweets do not include location information, a Geotagger is utilized. The Geotagger analyzes the tweets and checks whether there are references to places. If this is the case, it assigns coordinates to tweets using Open Street Maps[21] and a Lucene[22] index following the method described in [27]. The resulting tweets are forwarded to the Text Classifier component that identifies tweets that talk about traffic or flood incidents. All identified event-related tweets are stored to a MongoDB database for further analysis. The bottleneck of the system are the Geotagger and the Text Classifier components. However, these components are easily replicated on multiple machines each of them handling a different sub-stream of the original stream without any impact on the effectiveness of event detection. The architecture of the Twitter-ISA is presented in Fig. 8.

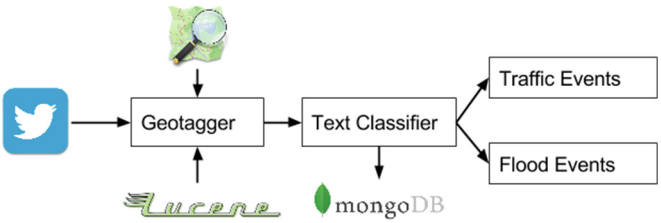

Fig. 8. The architecture of the Twitter-ISA of the INSIGHT system.

6.2 Data Stream Topologies

The authors in [56] suggested a distributed framework for high-volume data streams like the Twitter Firehose (nearly 100 % of the Twitter feed). They propose a Storm topology in order to enable parallel and distributed computations

[19] http://www.insight-ict.eu/.
[20] https://dev.twitter.com/streaming/reference/post/statuses/filter.
[21] https://www.openstreetmap.org.
[22] https://lucene.apache.org/.

implementing the first story detection algorithm described in [66]. A key component of this algorithm is finding the nearest neighbor of a document. The basic idea of the distributed streaming topology is to divide the Twitter stream into sub-streams without reducing the evidence that could aid event detection.

The first topology layer is the Vectorizer that converts the tweets to the vector space using a bag-of-words approach. The next layer is the "Hashing" that distributes the tweets on different processing units. The authors suggest the usage of Locality Sensitive Hashing (LSH) in order to partition the documents into multiple LSH-buckets, with similar content, belonging on Storm Bolts. The intuition behind the multiple-bucket partitioning, using multiple LSH data-structures, is to reduce the nearest neighbour error caused by LSH and is described in more detail in [66]. Each document is sent to multiple bolts involving extra communication cost but reducing the LSH error. Those bolts belong to the Local Distance layer where each of them reports the nearest neighbour to a document identified from its buckets. In the next layer named Global Distance, the nearest neighbors are aggregated and the one with the smallest distance from the new document is selected. Documents who have distance more than a threshold from the global nearest neighbour are sent to the "K-Means clustering" layer that performs online clustering, the rest are discarded. Each of the formed clusters may correspond to a real world event. The storm topology is presented on Fig. 9.

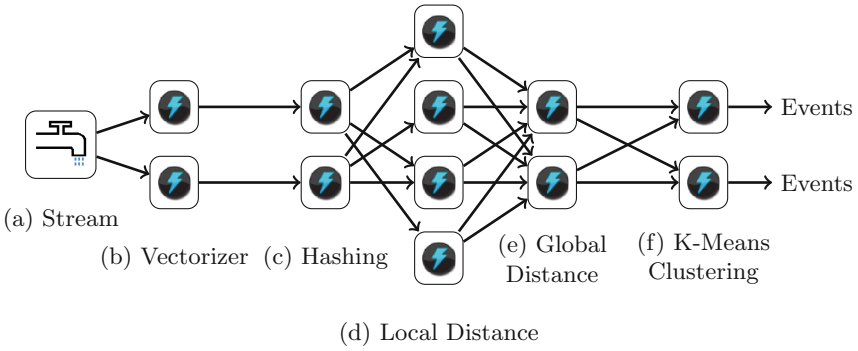

Fig. 9. The storm topology with multiple bolts per component. The (d) local distance layer is allocated the most cores since it is the most computationally intensive.

They found that the fastest layer is, as expected, the Vectorizer. The slowest one is the Local Distance. This is explained by the fact that this bolt implements a nearest neighbour computation involving similarity calculations on high dimensional vectors. It is important to measure the slowest layer in order to allocate cores where it is necessary. The authors suggest that the throughput of the system scales linearly as more and more cores are added to the right bolt layers. After investigating the number of cores that will be needed in order to process the entire Twitter Firehose (5000 tweets/s) the authors concluded that 70 cores or 9 8-core machines will suffice.

6.3 Summary

In this section, system architectures that are capable of performing real-time event detection are presented. Most of them focus on the Twitter sample stream and the Twitter Garden hose with almost 10 million tweets per day. For managing and mining the sample stream (1 % of Twitter) a distributed modular architecture is required.

Notably, the 1 % of the Twitter stream on March of 2014 was about 8 million tweets per day. This volume is similar to the volume of the Gardenhose API (10 % of Twitter) provided some years before. This fact confirms the growth of the Twitter usage over the years. If one is willing to process the entire Twitter stream she needs to turn to architectures like Storm and to use the appropriate number of processing units. For a smaller dataset such as a stream of 1 million tweets per day a single machine with high amounts of main memory and processing power will be able to process the data at run-time. It is important to note that not all modules involved in the methods presented can be replicated. This is due to the fact that not all algorithms can be parallelised without affecting the quality of results (e.g. [66]). A summary of the above approaches that contains the stream rate and the approach architecture is presented on Table 4.

7 Applications

In this section, we present a set of interesting applications of event detection systems and methods. The applications range from generic global events to celebrity specific incidents.

In [36] the authors used Twitter to identify tweets that are about health issues. This study investigates what types of links the users consult for publishing health related information. A similar application is presented in [7] where authors collect tweets about Influeza and identify flu outbreaks. Their results are similar to Google-trends based flu outbreak detection especially in the early stages of the outbreak. It is easy to see the potential social impact of such applications.

[72] focuses on identifying earthquake incidents with Twitter users as sensors. The authors make an effort to detect the location and the trajectory of the phenomenon. The system monitors Twitter and emails citizens when an earthquake

Table 4. The size of corpus and frameworks used in the papers presented in this Section.

Reference	Data volume	Frameworks
[86]	3 M/Day	MongoDB Spatial-temporal indexes, horizontally Scaling
[1]	Twitter sample stream 10 M/day	JOSM
[88]	15 M/day	Apache Solr, geo-hash
[73]	Above 15 M/day	-
[56]	Gardenhose and Firehose	Storm Topology

is detected. The response time of the system is proved to be quite fast, similar to the Japan Meteorological Agency. In [30] the authors detect flood events in Germany providing visual information on the map. The TEDAS system [52] targets Crime and Disaster incidents by identifying where and when they happened. A map visualization of tweets is available. Flickr and Youtube are utilized in [68] where the goal is to detect content related to an emergency. The above systems help the authorities in detecting real-time incidents as well as in extracting useful information after the event.

Another set of approaches focused on finding global important events for a given time period. These include [12,43,66]. [66,67] emphasize on finding the first story about a new event (new event detection). Such approaches are valuable since they can aid in identifying unexpected events.

Medvent et al. [58] focused on detecting events related to specific brands. They focused on three major brands: Google, Microsoft and Apple. Examples of such events are the release of a new product like the new iPad or Microsoft's Security Essential software. In order to achieve the desired outcome, the authors study the sentiment of the tweets. These techniques are utilized for marketing purposes. A similar approach is presented in [69] where events of controversial sentiment are targeted. Automatic identification of controversies is very useful in order to track and manage a large number of discussion groups.

Noettcher et al. [15], developed an Android application that finds local events given a specific geographic area. The application is able to provide summaries to the users. The Jasmine system detects local events for the user according to the desired size of event and the number of users attending it. It presents summaries and some important tweets per event in order to provide with a short description. This group of applications could support mobile users looking for "happenings" near by.

In the area of sports analytics, the EvenTweet system [1] could detect the start time and the location of football matches for UEFA 2012. The system described in [94] was able to detect National Football League events of the 2010–2011 season. The events include touchdowns, interceptions and goals.

8 Evaluation

Event Detection in social media is a relatively new and rather complex problem, especially when it comes to evaluating the suggested approaches. Most authors have to evaluate their algorithms during a period where important global events take place. This will enable the validation of their techniques. Another approach is to insert artificially event tweets into the stream. In this section a review of the most common evaluation practises is presented. Experimental set-up, utilized metrics, and obtained results are presented. Moreover we will cover strategies for labeling the data and provide links to publicly available datasets.

8.1 Dataset Labelling

Many approaches constructed a dataset using the Twitter API. From the collected tweets they create clusters of messages and label these clusters either using the most important words of the cluster or the centroid of the cluster. Usually, more than one annotators are used and the agreement between them is measured using Cohen's Kappa [90]. Only the annotations with high agreement are used in the most cases while the low agreement annotations are discarded since they are considered noise.

The authors in [2] followed a supervised approach on classifying event-candidate clusters as event or non-event. Their system required a training set for the supervised classifier used by their method. In order to assemble such a dataset they initially extracted 1000 clusters using their one pass online clustering algorithm. Then they manually labelled these clusters. 319 clusters are labeled as events whereas 681 are labelled as non-events suggesting that, as expected, the two classes are slightly unbalanced.

In a similar fashion the authors in [12] tracked the same problem. Sharing the supervised classification idea with [86] they required a training set consisting of example clusters marked as event and non-event. They manually annotated clusters, but these clusters are carefully selected. Instead of annotating the clusters or annotating a random subset, they restricted the cluster selection to the top-20 fastest growing clusters per hour. The assumption behind this is that usually a cluster that suddenly increases in size, will be an event cluster. For the testing set they sample randomly clusters per hour from the whole cluster pool in order to depict the real balance between "event" and "no-event" clusters. The annotators labelled the clusters as "real world event", "Twitter centric activity", "non-event" and "ambiguous". Two annotators provided judgments and Cohen's Kappa is used in order to measure the agreement. The clusters used for the training are 504, favoring the event class due to the careful cluster selection. The test-set consisted only of 300 clusters.

The authors in [66] used the Edinburgh Fist Story Detection (FSD) Corpus. A simply modified version of this dataset will be presented in the next sections. From this dataset they created threads of messages using their threading algorithm described in the same paper. The threading algorithm links related documents according to their textual distance and creates clusters of similar documents. Authors divided the stream in sliding windows and then for every sliding window they extracted the fastest growing threads(clusters) and manually labelled them. The top 1000 fastest growing threads from a sliding window of $100,000$ threads are labelled using two annotators and using Cohen's Kappa coefficient. Clearly, it is important to note that again the reason why the fastest growing clusters are selected for annotation is in order to favor the event clusters similarly to [12]. Otherwise, the dataset would contain only a very small proportion of clusters labeled as "event".

An alternative approach is described in [57]. There, the wisdom of the crowd is utilized through Amazon's Mechanical Turk.

They selected candidate clusters of events using the LSH [66] algorithm and the Cluster Summary algorithm [2]. They also utilized the Wikipedia Events Portal in order to receive event clusters as well as tweets about the detected events. Then the crowd is used to determine if the cluster tweets are about the event. In addition some clever heuristics are used in order to increase the annotators agreement and also to filter out low-quality annotations. On the same time, these heuristics provided the annotators motivation to continue their high quality work.

8.2 Evaluation Metrics and Results

In this section, we provide an overview of the results obtained by various studies presented in previous sections. Given the fact that there is an absence of shared datasets, a direct comparison is impossible. However, the following metrics serve as indicators of performance in various problems. Furthermore we present information on the metrics used in each case.

Many authors decided to test the performance of their algorithms on the TDT5 dataset. This dataset contains news articles extracted from traditional news media and was widely used for the TDT challenge. Naturally, results will diverge in a Twitter dataset since the two information sources are different in many ways. For example, the streaming FSD algorithm [66] demonstrated much better performance on the TDT5 dataset in comparison to a Twitter dataset.

The most common evaluation metrics originate from fields like Information Retrieval and Natural Language Processing. Typical examples are Precision, Recall and the F-Measure in terms of the detected events. In some cases, Accuracy is reported whereas in others the number of detected events is used as an indicator of effectiveness. The latter might be misleading in cases of unbalanced classes such as event detection. At this point, we will define some of the basic metrics frequently used in the literature. Precision is defined in Eq. 7. *Actual Events* (or *True Positives*) is the number of times that the algorithm detected an event and it is actually an event. *Recall* shows the percentage of the actual events that the system is able to identify (see Eq. 8). F-Measure is the harmonic mean of Precision and Recall and it is defined in Eq. 9.

Many approaches can achieve high Recall but with limited Precision due to the large number of False Positives. This is one of the reasons that additional 'filtering' techniques are utilized before or after the core approach. Some methods use ranked versions of the above evaluation metrics such as Precision at k ($P@K$). Such metrics allow the evaluation of methods that can provide an ordered list of predicted events.

$$Precision = \frac{\text{Number of Actual Events Detected}}{\text{Number of Detected Events}} \tag{7}$$

$$Recall = \frac{\text{Number of Actual Events Detected}}{\text{Number of Actual Events}} \tag{8}$$

$$F\text{-}Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{9}$$

The authors in [12] decided to use a manually labelled dataset for the evaluation of their system. Their approach utilized classifiers that are compared based on the F_1 measure. SVMs outperformed the Naive Bayes classifier (0.837 over 0.702).

In [66] the authors evaluate their system in terms of average Precision. This decision is straightforward since labels are available only for the detected events (True Positives). The authors evaluate their first story detection system on the top-k event stories ranked according to different scoring functions. The best obtained results in terms of Precision at k ($P@k$) is 34.0 %.

The same event detection system is tested in [67] where Wikipedia is utilized in order to re-rank the detected events. The authors observed that the use of Wikipedia provided an improvement in Precision. However, as the authors comment, Wikipedia causes a two-hour delay in event detection in comparison with the original approach [66].

The authors in [59] evaluated their system on the same dataset [66]. They conclude that their approach outperformed UMASS [4] while the detection time is only 3.5 times slower than the method in [66]. However, the benefits of the approach are not so clear in Twitter data as they are in FSD data (TDT5).

[43] utilized the corpus of [59]. They used Detection Trade-off (DET) curves for evaluating the effectiveness of the approach. DET curves display the ratio of *Miss Probability* to *False Alarm Probability*). The conclusion is that their system outperformed all baseline approaches included in the experiments. On top of that, a significant improvement is observed in execution time.

In [86] the authors evaluated three classifiers in terms of Precision, Recall and F-Measure. They used a manually labelled dataset by 10-fold cross-validation. Their best performing classifier is a Pruned Decision Tree with a F_1 score of 0.857. They also experimented with the impact of the content and non-content features. The authors observed a statistically significant improvement when all types of features are considered. One could note however that these results deviate from the ones reported in [12,67]. This gap demonstrates the effect of the dataset in an experimental evaluation.

The inventors of TwEvent [51] provided with their own definitions of Recall and Duplicate Event Rate (DER) that are the metrics used in their evaluation. Recall is defined as the number of detected events while DER is the ratio of duplicate events found. DER is useful in order to penalize multiple alerts on the same event. They compared the approach against the EdCow system [89] and concluded that TwEvent achieved an important improvement in terms of Recall (75 over 13 detected actual events). An improvement is also observed in Precision (86.1 % over 76.2 %). The DER metric of TwEVent system and EdCow is 16.0 % and 23.1 % respectively.

Similarly, Popescu et al. [69] utilized a manually labelled dataset that consists of 800 labelled events using two human annotators. Three alternative approaches are compared on Precision at k ($P@k$). The so-called *blended* model performed best with 0.9 Precision at rank-1 and 0.80 Precision at rank-4. The Area Under Curve (AUC) of the three models suggests that they have good discriminative

power in comparison to baseline algorithms. A final note is that the performance differences between the three systems is not statistically significant.

8.3 Available Datasets

The Edinburgh FSD corpus[23] was created in order to test the method in [66]. The dataset contains 51,879,318 tweet IDs. The content of the tweets is removed due to Twitter's terms of use. In order to take advantage of the dataset one has to use the Twitter API to download the messages that correspond to the tweet IDs. The authors identified 27 topics in the data. 3034 tweets are labelled according to the procedure described in [59]. This dataset was created for detecting first stories. However, it is suitable (and was utilized) for general event detection tasks.

A dataset that is not Twitter specific but is useful for event detection evaluation is the NewsWire dataset[24]. The dataset contains links to news articles. The articles contain a timestamp and a relevance value to some of the aforementioned 27 topics [65]. The dataset contains 47751 links to articles.

The dataset of the MediaEval challenge is also available and can be utilized for event detection. The dataset consisted of Flickr images and 1.327 videos from YouTube with their metadata. Another one consists of Instagram pictures instead of Flickr images. The labelled part of the dataset was created using human annotators. This dataset contains 8 event types. These are music events, conferences, exhibitions, fashion shows, protests, sport events, theatrical/dance events and other events.

9 Related Problems

Trend detection is a highly related task to event detection and is commonly applied to social media (e.g. Twitter trending topics) and News portals (e.g. Yahoo News). Many trend detection methods like [13,55] are similar to feature-pivot event detection techniques. In these methods, a keyword burst identification is a core element. Similarly to event detection, scalability for high volumes of data is a major concern.

Information diffusion [37] is another problem that shares many similarities with event detection. Twitter [71,91] and Facebook [10] have been extensively studied on how information flows inside the network. Information diffusion examines the impact of the network structure, which users are influential or why some content becomes viral.

'Event Detection' is a term commonly used in video/image analysis and computer vision [40,79,95]. In this case the goal is to identify in a video feed an incident - usually of specific type. Similar efforts have been observed in image and video streams in social networks like Instagram, Flick and YouTube [63,64,85].

Other domains for event detection emerge as new information sources become available. Mobile and Urban data are now in abundance in smart cities. Hence,

[23] Available at http://demeter.inf.ed.ac.uk/cross/docs/fsd_corpus.tar.gz.
[24] Available at http://demeter.inf.ed.ac.uk/cross/docs/Newswire_Events.tar.gz.

data analysis and event processing techniques as well as complete streaming frameworks are exploited in order to identify incidents in the streets of a city [8,16,17,74]. Data sources that are utilized in such cases are SCATS[25] data (traffic volume information) or vehicle data like public transport data (e.g. GPS location of buses moving around the city). The INSIGHT project develops a system that targets at identifying disastrous events from city data.

10 Conclusion and Open Challenges

In this paper we presented an overview of the most recent techniques for detecting events in online social networks. This is an area of research that emerged during the last years, in parallel with the growth of user participation in social networks. In this overview, we made an effort to organize the most important research lines as well as their results. Furthermore we focused on the architecture element of such systems. Due to large volumes of data, state-of-the-art data stream and database frameworks had to be utilized. Finally we discussed how the evaluation is being executed in event detection and mentioned the most common evaluation metrics and datasets used. We believe that this survey will benefit researchers in the field as well as practitioners working in commercial applications that exploit social network applications.

The problem of event detection is a very challenging one. The definition of the problem in Sect. 3, suggests that there are many dimensions to it. It is not sufficient to detect that something happened, in other words, detect anomalies. Event detection requires the automatic answering of what, when, where, and by whom. After reporting on the most recent efforts in the area, it is clear that no method addressed all of these questions. Therefore, there is a lot of space for improvement towards this direction.

Another challenge that has to be addressed is the lack of public datasets. Privacy issues along with Social Network companies' terms of use hinder the availability of shared data. This obstacle, is of great significance since it relates to the repeatability of experiments and comparison between approaches. It is not hard to observe that most approaches focus on the Twitter platform. This is of course due to the usability and accessibility of the Twitter API. However, a research area that depends on a single data source, as interesting as it is, entails many risks. Nonetheless, it is expected that as new media sources emerge, event detection will remain significant and challenging.

Acknowledgments. This work is funded by the projects EU FP7 INSIGHT (318225), GGET Thalis DISFER and GeomComp.

References

1. Abdelhaq, H., Sengstock, C., Gertz, M.: EvenTweet: online localized event detection from twitter. Proc. VLDB Endow. **6**(12), 1326–1329 (2013)

[25] http://en.wikipedia.org/wiki/Sydney_Coordinated_Adaptive_Traffic_System.

2. Aggarwal, C.C., Subbian, K.: Event detection in social streams. In: SDM, pp. 624–635. SIAM/Omnipress (2012)
3. Allan, J.: Introduction to topic detection and tracking. In: Allan, J. (ed.) Topic Detection and Tracking, pp. 1–16. Springer, New York (2002)
4. Allan, J., Lavrenko, V., Malin, D., Swan, R.: Detections, bounds, and timelines: Umass and TDT-3. In: Proceedings of Topic Detection and Tracking Workshop, pp. 167–174 (2000)
5. Allan, J., Papka, R., Lavrenko, V.: On-line new event detection and tracking. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM (1998)
6. Ankerst, M., Breunig, M., Kriegel, H., Sander, J.: OPTICS: ordering points to identify the clustering structure. ACM SIGMOD Rec. **28**, 49–60 (1999)
7. Aramaki, E., Maskawa, S., Morita, M.: Twitter catches the flu: detecting influenza epidemics using Twitter. In: Proceedings of the Conference on empirical methods in natural language processing, pp. 1568–1576 (2011). http://dl.acm.org/citation.cfm?id=2145600
8. Artikis, A., Weidlich, M., Schnitzler, F., Boutsis, I., Liebig, T., Piatkowski, N., Bockermann, C., Morik, K., Kalogeraki, V., Marecek, J., et al.: Heterogeneous stream processing and crowdsourcing for urban traffic management. In: EDBT, pp. 712–723 (2014)
9. Atefeh, F., Khreich, W.: A survey of techniques for event detection in twitter. Computat. Intell. **31**, 132–164 (2013)
10. Bakshy, E., Rosenn, I., Marlow, C., Adamic, L.: The role of social networks in information diffusion. In: Proceedings of the 21st International Conference on World Wide Web, pp. 519–528. ACM (2012)
11. Becker, H., Iter, D., Naaman, M., Gravano, L.: Identifying content for planned events across social media sites. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM 2012, p. 533 (2012)
12. Becker, H., Naaman, M., Gravano, L.: Beyond trending topics: real-world event identification on twitter. In: Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 2011), pp. 1–17 (2011)
13. Benhardus, J., Kalita, J.: Streaming trend detection in twitter. Int. J. Web Based Commun. **9**(1), 122–139 (2013). http://inderscience.metapress.com/index/906V117647682257.pdf
14. Bifet, A., Frank, E.: Sentiment knowledge discovery in twitter streaming data. In: Pfahringer, B., Holmes, G., Hoffmann, A. (eds.) DS 2010. LNCS, vol. 6332, pp. 1–15. Springer, Heidelberg (2010)
15. Boettcher, A., Lee, D.: EventRadar: a real-time local event detection scheme using twitter stream. In: 2012 IEEE International Conference on Green Computing and Communications, pp. 358–367, November 2012
16. Boutsis, I., Kalogeraki, V.: Privacy preservation for participatory sensing data. In: 2013 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 103–113. IEEE (2013)
17. Boutsis, I., Kalogeraki, V., Gunopulos, D.: Efficient event detection by exploiting crowds. In: Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems, pp. 123–134. ACM (2013)
18. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K.: Measuring user influence in twitter: the million follower fallacy. In: ICWSM 2010, pp. 10–17 (2010)
19. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. Theort. Comput. Sci. **312**(1), 3–15 (2004)

20. Chen, K., Chen, T., Zheng, G., Jin, O., Yao, E., Yu, Y.: Collaborative personalized tweet recommendation. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 661–670. ACM (2012)
21. Chowdhury, G.: Introduction to Modern Information Retrieval. Facet Publishing, London (2010)
22. Cordeiro, M.: Twitter event detection: combining wavelet analysis and topic inference summarization. In: Doctoral Symposium on Informatics Engineering, DSIE (2012). http://paginas.fe.up.pt/prodei/dsie12/papers/paper_14.pdf
23. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. Theoretical Computer Science **55**(1), 58–75 (2005). http://linkinghub.elsevier.com/retrieve/pii/S0196677403001913
24. Cormode, G., Muthukrishnan, S.: What's hot and what's not: tracking most frequent items dynamically. Theoretical Computer Science **30**(1), 249–278 (2004). http://portal.acm.org/citation.cfm?d=1061318.1061325
25. Cui, A., Zhang, M., Liu, Y., Ma, S., Zhang, K.: Discover breaking events with popular hashtags in twitter. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM 2012, p. 1794 (2012). http://dl.acm.org/citation.cfm?d=2396761.2398519
26. Daly, E.M., Geyer, W.: Effective event discovery: using location and social information for scoping event recommendations. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 277–280. ACM (2011)
27. Daly, E.M., Lecue, F., Bicer, V.: Westland row why so slow?: fusing social media and linked data sources for understanding real-time traffic conditions. In: Proceedings of the 2013 International Conference on Intelligent User Interfaces, pp. 203–212. ACM (2013)
28. De Choudhury, M., Gamon, M., Counts, S., Horvitz, E.: Predicting depression via social media. In: AAAI Conference on Weblogs and Social Media, vol. 2 (2013)
29. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, New York (2012)
30. Fuchs, G., Andrienko, N., Andrienko, G., Bothe, S., Stange, H.: Tracing the German centennial flood in the stream of tweets: first lessons learned. In: Proceedings of the Second ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information, GEOCROWD 2013, pp. 31–38. ACM, New York (2013). http://doi.acm.org/10.1145/2534732.2534741
31. Fung, G.P.C., Yu, J.X., Yu, P.S., Lu, H.: Parameter free bursty events detection in text streams. In: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 181–192. VLDB Endowment (2005)
32. Galuba, W., Aberer, K.: Outtweeting the twitterers-predicting information cascades in microblogs. In: Proceedings of the 3rd Conference on Online Social Networks (2010). http://static.usenix.org/events/wosn10/tech/full_papers/Galuba.pdf
33. Ghosh, S., Sharma, N., Benevenuto, F., Ganguly, N., Gummadi, K.: Cognos: crowdsourcing search for topic experts in microblogs. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 575–590. ACM (2012)
34. Ginsberg, J., Mohebbi, M.H., Patel, R.S., Brammer, L., Smolinski, M.S., Brilliant, L.: Detecting influenza epidemics using search engine query data. Nature **457**(7232), 1012–1014 (2009)
35. Go, A., Huang, L., Bhayani, R.: Twitter sentiment analysis. Nature **17**, 1–6 (2009)

36. Goot, E.V.D., Tanev, H., Linge, J.: Combining twitter and media reports on public health events in medisys. In: Proceedings of the 22nd International Conference on World Wide Web Companion, pp. 703–705. International World Wide Web Conferences Steering Committee (2013). http://dl.acm.org/citation.cfm?id=2488028
37. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. In: Proceedings of the 13th International Conference on World Wide Web, pp. 491–501. ACM (2004)
38. Guo, D., Wu, J., Chen, H., Yuan, Y., Luo, X.: The dynamic bloom filters. IEEE Trans. Knowl. Data Eng. **22**(1), 120–133 (2010)
39. Hua, T., Chen, F., Zhao, L., Lu, C., Ramakrishnan, N.: STED: Semi-Supervised Targeted Event Detection (2013). people.cs.vt.edu, http://people.cs.vt.edu/ramakris/papers/kdddemo13_sted.pdf
40. Itti, L., Baldi, P.: A principled approach to detecting surprising events in video. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 1, pp. 631–637. IEEE (2005)
41. Jarvis, R.A., Patrick, E.A.: Clustering using a similarity measure based on shared near neighbors. IEEE Trans. Comput. **100**(11), 1025–1034 (1973)
42. Kaleel, S.B.: Event Detection and trending in multiple social networking sites. In: Proceedings of the 16th Communications and Networking Symposium. Society for Computer Simulation International (2013)
43. Karkali, M., Rousseau, F., Ntoulas, A., Vazirgiannis, M.: Efficient online novelty detection in news streams. In: Lin, X., Manolopoulos, Y., Srivastava, D., Huang, G. (eds.) WISE 2013, Part I. LNCS, vol. 8180, pp. 57–71. Springer, Heidelberg (2013)
44. Kleinberg, J.: Bursty and hierarchical structure in streams. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, p. 91 (2002). http://portal.acm.org/citation.cfm?d=775047.775061
45. Knobel, M., Lankshear, C.: Online memes, affinities, and cultural production. In: Knobel, M., Lankshear, C. (eds.) A New Literacies Sampler, pp. 199–227. Peter Lang, New York (2007)
46. Kotsakos, D., Sakkos, P., Katakis, I., Gunopulos, D.: # tag: meme or event? In: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 391–394. IEEE (2014)
47. Kumar, S., Liu, H., Mehta, S., Subramaniam, L.V.: From tweets to events: exploring a scalable solution for twitter streams. arXiv preprint, arXiv:1405.1392 (2014)
48. Kumaran, G., Allan, J.: Using names and topics for new event detection. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, pp. 121–128. Association for Computational Linguistics (2005)
49. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web, pp. 591–600. ACM (2010)
50. Levenberg, A., Osborne, M.: Stream-based randomised language models for SMT. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, vol. 2. Association for Computational Linguistics (2008)
51. Li, C., Sun, A., Datta, A.: Twevent: segment-based event detection from tweets. Proceedings of the 21st ACM International Conference on Information and Knowledge Management (2012). http://dl.acm.org/citation.cfm?id=2396785

52. Li, R., Lei, K.H., Khadiwala, R., Chang, K.C.C.: TEDAS: a twitter-based event detection and analysis system. In: 2012 IEEE 28th International Conference on Data Engineering, pp. 1273–1276, April 2012
53. Luo, G., Tang, C., Yu, P.S.: Resource-adaptive real-time new event detection. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD 2007, p. 497 (2007)
54. Manku, G.S., Motwani, R.: Approximate frequency counts over data streams. In: Proceedings of the 28th International Conference on Very Large Data Bases (2002). http://dl.acm.org/citation.cfm?id=1287400
55. Mathioudakis, M., Koudas, N.: Twittermonitor: trend detection over the twitter stream. In: Proceedings of the 2010 International Conference on Management of Data, pp. 1155–1157 (2010). http://dl.acm.org/citation.cfm?id=1807306
56. McCreadie, R., Macdonald, C.: Scalable distributed event detection for Twitter. In: 2013 IEEE International Conference on Big Data, 6–9 January 2013. IEEE (2013)
57. McMinn, A.J., Moshfeghi, Y., Jose, J.M.: Building a large-scale corpus for evaluating event detection on twitter. In: Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2013 pp. 409–418 (2013). http://dl.acm.org/citation.cfm?doid=2505515.2505695
58. Medvet, E., Bartoli, A.: Brand-related events detection, classification and summarization on twitter. In: 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pp. 297–302 (2012). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6511900
59. Osborne, M., Petrovic, S.: Bieber no more: first story detection using Twitter and Wikipedia. In: Proceedings of the Workshop on Time-Aware Information Access, TAIA (2012)
60. Ozdikis, O., Senkul, P., Oguztuzun, H.: Semantic expansion of tweet contents for enhanced event detection in twitter. In: 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 20–24 (2012). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6425790
61. Ozdikis, O., Senkul, P., Oguztuzun, H.: Semantic expansion of hashtags for enhanced event detection in Twitter. In: Proceedings of the 1st International Workshop on Online Social Systems (2012). http://www.cs.ubc.ca/welu/woss2012/papers/1-ozdikis.pdf
62. Packer, H.S., Samangooei, S., Hare, J.S., Gibbins, N., Lewis, P.H.: Event detection using Twitter and structured semantic query expansion. In: Proceedings of the 1st International Workshop on Multimodal Crowd Sensing, CrowdSens 2012, p. 7 (2012)
63. Papadopoulos, S., Schinas, E., Mezaris, V., Troncy, R., Kompatsiaris, I.: The 2012 social event detection dataset. In: Proceedings of the 4th ACM Multimedia Systems Conference, pp. 102–107. ACM (2013)
64. Papadopoulos, S., Troncy, R., Mezaris, V., Huet, B., Kompatsiaris, I.: Social event detection at MediaEval 2011: challenges, dataset and evaluation. In: MediaEval (2011)
65. Petrovic, S., Osborne, M.: Can twitter replace newswire for breaking news. In: Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media 2011 (2013)
66. Petrovic, S., Osborne, M., Lavrenko, V.: Streaming first story detection with application to twitter. In: Proceedings of the NAACL (2010)

67. Petrović, S., Osborne, M., Lavrenko, V.: Using paraphrases for improving first story detection in news and Twitter. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 338–346 (2012)
68. Pohl, D., Bouchachia, A., Hellwagner, H.: Automatic sub-event detection in emergency management using social media. In: Proceedings of the 21st International Conference Companion on World Wide Web, WWW 2012 Companion, p. 683 (2012). http://dl.acm.org/citation.cfm?d=2187980.2188180
69. Popescu, A.M., Pennacchiotti, M.: Detecting controversial events from twitter. Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010, p. 1873 (2010). http://portal.acm.org/citation.cfm?d=1871437.1871751
70. Psallidas, F., Becker, H., Naaman, M., Gravano, L.: Effective event identification in social media. IEEE Trans. Comput. **36**(3), 42–50 (2013). http://sites.computer.org/debull/A13sept/p42.pdf
71. Romero, D.M., Meeder, B., Kleinberg, J.: Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In: Proceedings of the 20th International Conference on World Wide Web, pp. 695–704. ACM (2011)
72. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proceedings of the 19th International Conference on World Wide Web (2010). http://dl.acm.org/citation.cfm?id=1772777
73. Sankaranarayanan, J., Samet, H.: Twitterstand: news in tweets. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM (2009). http://dl.acm.org/citation.cfm?id=1653781
74. Schnitzler, F., Liebig, T., Mannor, S., Morik, K.: Combining a Gauss-Markov model and Gaussian process for traffic prediction in Dublin city center. In: Proceedings of the Workshop on Mining Urban Data at the International Conference on Extending Database Technology (2014, to appear)
75. Sharma, J., Vyas, A.: Twitter sentiment analysis. Indian Institute of Technology (2010, unpublished). http://home.iitk.ac.in/jaysha/cs365/projects/report.pdf)
76. Signorini, A., Segre, A.M., Polgreen, P.M.: The use of Twitter to track levels of disease activity and public concern in the US during the influenza A H1N1 pandemic. IEEE Trans. Comput. **6**(5), e19467 (2011)
77. Slaney, M., Casey, M.: Locality-sensitive hashing for finding nearest neighbors [lecture notes]. IEEE Trans. Comput. **25**(2), 128–131 (2008)
78. Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D.: Online outlier detection in sensor data using non-parametric models. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 187–198. VLDB Endowment (2006)
79. Tang, K., Fei-Fei, L., Koller, D.: Learning latent temporal structure for complex event detection. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1250–1257. IEEE (2012)
80. Tumasjan, A., Sprenger, T.O., Sandner, P.G., Welpe, I.M.: Predicting elections with twitter: what 140 characters reveal about political sentiment. In: ICWSM 2010, 178–185 (2010)
81. Valkanas, G., Gunopulos, D.: Location extraction from social networks with commodity software and online data. In: 2012 IEEE 12th International Conference on Data Mining Workshops pp. 827–834. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6406525

82. Valkanas, G., Gunopulos, D.: Event detection from social media data. IEEE Transactions on Computers **36**(3), 51–58 (2013)
83. Valkanas, G., Gunopulos, D.: How the live web feels about events. In: Proceedings of the 22Nd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2013, pp. 639–648. ACM, New York (2013). http://doi.acm.org/10.1145/2505515.2505572
84. Valkanas, G., Gunopulos, D., Boutsis, I., Kalogeraki, V.: An architecture for detecting events in real-time using massive heterogeneous data sources. In: Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining Algorithms, Systems, Programming Models and Applications, BigMine 2013, pp. 103–109 (2013). http://dl.acm.org/citation.cfm?d=2501221.2501235
85. Vavliakis, K.N., Tzima, F.A., Mitkas, P.A.: Event detection via LDA for the MediaEval2012 SED task. In: MediaEval, pp. 5–6 (2012)
86. Walther, M., Kaisser, M.: Geo-spatial event detection in the twitter stream. In: Serdyukov, P., Braslavski, P., Kuznetsov, S.O., Kamps, J., Rüger, S., Agichtein, E., Segalovich, I., Yilmaz, E. (eds.) ECIR 2013. LNCS, vol. 7814, pp. 356–367. Springer, Heidelberg (2013)
87. Wang, Y., Sundaram, H., Xie, L.: Social event detection with interaction graph modeling. In: Proceedings of the 20th ACM International Conference on Multimedia (2012). http://dl.acm.org/citation.cfm?id=2396332
88. Watanabe, K., Ochi, M., Okabe, M., Onai, R.: Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 2541–2544 (2011). http://dl.acm.org/citation.cfm?id=2064014
89. Weng, J., Lee, B.: Event detection in twitter. In: ICWSM (2011)
90. Wood, J.M.: Understanding and computing Cohen's kappa: a tutorial. WebPsychEmpiricist. Web J. (2007) http://wpe.info/
91. Yang, J., Counts, S.: Predicting the speed, scale, and range of information diffusion in twitter. In: ICWSM 2010, pp. 355–358 (2010)
92. Yang, Y., Pierce, T., Carbonell, J.: A study of retrospective and on-line event detection. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 28–36. ACM (1998)
93. Zhang, K., Zi, J., Wu, L.G.: New event detection based on indexing-tree and named entity. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2007, p. 215 (2007). http://portal.acm.org/citation.cfm?d=1277741.1277780
94. Zhao, S., Zhong, L.: Human as real-time sensors of social and physical events: a case study of twitter and sports games. arXiv preprint, arXiv:1106.4300, 1–9 June 2011. http://arxiv.org/abs/1106.4300
95. Zhong, H., Shi, J., Visontai, M.: Detecting unusual activity in video. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, vol. 2, pp. II–819. IEEE (2004)