Multi-spatial Scale Event Detection from Geo-tagged Tweet Streams via Power-law Verification

Yi Han, Shanika Karunasekera, Christopher Leckie, Aaron Harwood
School of Computing and Information Systems
The University of Melbourne
Parkville, Australia
{yi.han, karus, caleckie, aharwood}@unimelb.edu.au

arXiv:1906.05063v2 [cs.SI] 21 Aug 2019

***Abstract*—Compared with traditional news media, social media nowadays provides a richer and more timely source of news. We are interested in multi-spatial level event detection from geo-tagged tweet streams. Specifically, in this paper we (1) examine the statistical characteristic for the time series of the number of geo-tagged tweets posted from specific regions during a short time interval, *e.g.*, ten seconds or one minute; (2) verify from over thirty datasets that while almost all such time series exhibit self-similarity, those that correspond to events, especially short-term and unplanned outbursts, follow a power-law distribution; (3) demonstrate that these findings can be applied to facilitate event detection from tweet streams. We propose two algorithms—*Power-law basic* and *Power-law advanced*, where *Power-law basic* only checks the existence of power-law distributions in the time series from tweet streams at multi-spatial scales, without looking into the content of each tweet, and *Power-law advanced* integrates power-law verification with semantic analysis via word embedding. Our experiments on multiple datasets show that by considering spatio-temporal statistical distributions of tweets alone, the seemingly naive algorithm of *Power-law basic* achieves comparable results with more advanced event detection methods, while the semantic analysis enhanced version, *Power-law advanced*, can significantly increase both the precision and the recall.**

***Index Terms*—self-similarity, power-law distribution, multi-spatial event detection**

I. Introduction

Social media, especially Twitter, has become an increasingly more popular source of news. Compared with traditional forms of media, such as TV and newspapers, it often provides more timely information about various type of incidents. We are interested in real-life event detection at multi-spatial levels from geo-tagged tweet streams.

We initially investigated using Poisson models to monitor the fluctuations in the time series of the number of geo-tagged tweets posted within a bounding box during a short time interval, *e.g.*, ten seconds to one minute. However, our experimental results observe a relatively high false positive rate for this Poisson model based event detection method. This observation motivates us to reexamine the properties of these time series. Specifically, this paper aims to answer the following questions:

Section III: What are the statistical characteristics of the time series? A draw of several time series at different time scales, *i.e.,* the number of tweets posted every 1, 10, 60, 1000 seconds, shows that burstiness persists over all these scales, which indicates self-similarity [1], [2]. In order to verify this finding, we collect 33 tweet datasets of different types, generate the corresponding time series by counting the number of tweets posted every minute, and check self-similarity using three popular methods [3], [4]: aggregate variance, R/S and Whittle (please refer to Section II for a more detailed description on self-similarity and the three methods). Our results suggest that all the time series are self-similar.

Section IV: Can the time series be better characterised by other models than the Poisson process? The existence of self-similarity suggests that Poisson models are inadequate to capture the underlying dynamics in tweet streams. Instead, we examine whether a power-law distribution can be validated from these time series, and find that when an event occurs, it is indeed more likely to observe a power-law distribution in the time series generated from geo-tagged tweet streams.

Section V: Can the answers to the previous two questions be applied for event detection from geo-tagged tweet streams? We propose two event detection methods—*Power-law basic* and *Power-law advanced*: (1) *Power-law basic* only checks the existence of power-law distributions in the tweet stream at multi-spatial scales, without looking into the content of each tweet, or using any other information except the geo-location. Our experiments demonstrate that when combined with a Quad-tree [5], [6], this seemingly naive approach can achieve comparable performance with Geoburst [7][1], a widely cited event detection algorithm that considers temporal, spatial and semantic information; (2) *Power-law advanced* improves the algorithm by incorporating semantic analysis via word embedding, and our results suggest that it can significantly increase both the precision and the recall.

The remainder of this paper is organised as follows: Section II provides background information on self-similarity and

[1]Although the improved versions exist (Geoburst+ [8], TrioVec [9]), we do not use them as baselines in this work as they are supervised approaches.

power-law distributions; Section III describes the collected datasets, and checks whether the generated time series exhibit self-similarity; Section IV verifies the power-law hypothesis; Section V proposes two multi-spatial event detection algorithms, *Power-law basic* and *Power-law advanced*; Section VI summarises related work in event detection from social media; and Section VII concludes the paper and gives directions for future work.

II. Background on Self-similarity & Power-law Distributions

In this section, we briefly introduce the fundamental concepts in self-similarity and power-law distributions, including their definitions and the methods to verify them.

A. Self-similarity

Unlike traditional Poisson traffic, where short-term fluctuations average out over a longer period of time, self-similar traffic maintains burstiness at all time scales.

1) Definition: Before giving the definition of self-similarity, we first introduce the concept of an aggregated process: given a process $X = (X_i : i = 1, 2, 3, ...)$, its aggregated process is $X^{(m)} = (X_k^{(m)}, k = 1, 2, 3, ...)$, where m is the block size, $X_k^{(m)} = \frac{1}{m}\sum_{j=m\cdot(k-1)+1}^{m\cdot k} X_j$. In other words, $X^{(m)}$ partitions the original series X into non-overlapping segments of size m, and then averages over each segment.

A process X is called *exactly second-order self-similar* [1], [2] with parameter $H = 1-\beta/2$, $0 < \beta < 1$, if $R_m(k) = R(k) \sim k^{-\beta}$ as $k \to \infty$, where $R_m(\cdot)$ and $R(\cdot)$ are the autocorrelation functions for $X^{(m)}$ and X, respectively. The parameter H is called the Hurst parameter [10], [11], and for a self-similar process, $H \in (0.5, 1)$.

2) Methods to Test Self-similarity: SELFIS [3], [4] is a popular tool for testing self-similarity. It provides a number of methods to calculate the Hurst parameter, and the following three widely used methods are selected in this paper.

- Aggregate variance. A sufficient condition of self-similarity is $V_m = V \cdot m^{-\beta}$, $m \in Z^{>1} = \{2, 3, ...\}$, where V (V_m) is the variance of X ($X^{(m)}$). Therefore, if a log-log plot of V_m/V is drawn against m, then a straight line with slope β larger than -1 indicates self-similarity, and $H = 1 - \beta/2$.
- R/S. For a self-similar process, its *rescaled adjusted range* or R/S statistic can be represented by the relation:
$$\lim_{n\to\infty} E(\tfrac{R}{S}(n)) = C \cdot n^H,$$
where C is a finite positive constant, and n is the number of points in the process. Therefore, in the log-log plot of $\frac{R}{S}$ against n, the slope is an estimate of H.
- Whittle. The Whittle method applies maximum likelihood estimation to the spectral density function of X. It not only estimates H, but also produces a confidence interval.

B. Power-law Distribution

A power-law probability distribution [12], [13] takes the form of $p(x) \propto x^{-\alpha}$, where α is a positive constant often known as the exponent or scaling parameter.

A straightforward way to visualise a power-law distribution is to draw a log-log plot of the complementary cumulative distribution function (CCDF), and a roughly straight line is expected to be seen in the plot. However, this is only a necessary but not sufficient condition for a power-law distribution.

In order to validate that a time series X follows a power-law distribution, we first fit a power-law model to X, and then run the Kolmogorov-Smirnov (KS) test [14]. If the p-value of this significance test is below 0.05, the power-law hypothesis is rejected. Note that if a time series follows a power-law distribution, its log-log transformed CCDF is expected to be qualitatively similar at different scales, and hence it also exhibits self-similarity.

III. Verification of Self-similarity in Tweet Streams

In order to study the statistical characteristic of tweet streams, we have collected the following datasets:

- $D_1 - D_{30}$ (public)—A collection of 30 datasets associated with real-world events from 2012 to 2016, each of which contains from around 2×10^5 to nearly 3×10^7 tweets [15].
- D_{31} (public)—Twitter event detection dataset, which contains more than 120 million tweets (although the majority of these tweets are not associated with any event). The ground truth for 506 events and associated tweets are given [16].
- D_{32}—Twitter dataset shared by the authors of [8], which includes 9.5 million geo-tagged tweets from New York between 2014-08-01 and 2014-11-30.
- D_{33}—Over 920 thousand geo-tagged tweets collected from Melbourne between 2014 and 2018.

These datasets cover different types of tweets: $D_1 - D_{30}$ contain tweets that are only associated with specific events, the majority of which have worldwide impact; D_{31} contains tweets that both do and do not correspond to (mostly) local or less influential events, with the ground truth provided; D_{32} & D_{33} include all geo-tagged tweets from a region during a certain period of time.

Note that because the original datasets of $D_1 - D_{31}$ only include tweet ids, we have used a tool called "twitter-dataset-collector" [17] to download all the tweets. Since these datasets consist of hundreds of millions of tweets, it is infeasible to use the Twitter API to collect them due to the rate limit—it takes a significant amount of time to retrieve all the tweets. Instead, the tool crawls the webpages and reconstructs the original tweets. However, some tweets have already been deleted, and hence are not retrievable in this way. Even for those obtained tweets, the collected information is not as rich as in what the API returns. For example, most of the tweets do not have any location information, and the rest only have a location label—normally a city/town name, rather than specific coordinates. In addition, the second is truncated in the publication time.

For each of the above datasets, we count the number of tweets posted every minute, generate the corresponding time series, and test whether they exhibit self-similarity using three methods: aggregate variance, R/S and Whittle.

A. *Self-similarity in* $D_1 - D_{30}$

We start with the 30 datasets of real-world events with wide impact. As can be seen from Fig. 1a, all the estimates of the Hurst parameter are within the range of 0.5 and 1, which indicates that the corresponding time series are self-similar. Since we are more interested in geo-tagged tweets, we further examine the tweets that have a location label. The results in Fig. 1b suggest that these time series are self-similar too.

Among the 30 events, eleven of them are relatively short-term (from a few days to a couple of weeks), unplanned outbursts: Boston marathon bombing, Ferguson unrest, Gaza under attack, Ottawa shooting, Sydney siege, Charlie Hebdo shooting, Germanwings crash, Paris attacks, Brussels airport explosion, Cyprus hijacked plane and Lahore blast. This is the type of event that we are mostly interested in detecting from tweet streams. Therefore, for these eleven datasets, we extract tweets from close to where the events occurred, as those tweets will be most helpful in event detection. The results in Fig. 1c show that the corresponding time series exhibit self-similarity as well. Note that for the event of "Gaza under attack", insufficient data are collected locally, and hence it is not included.

B. *Self-similarity in* D_{31}

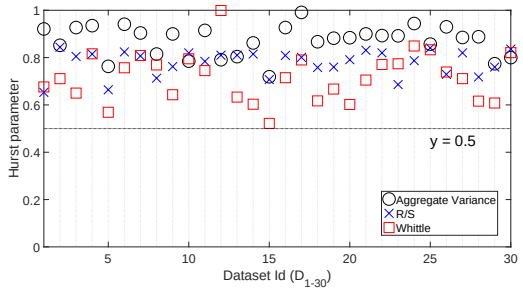
Next we examine the 506 events in the dataset of D_{31}, the majority of which are local or less influential compared to those in $D_1 - D_{30}$. The tool SELFIS requires that a time series should have a minimum length of 64—in our case, since we count the number of tweets posted every minute, this means the event needs to last for at least 64 minutes (suggested by the collected data). However, quite a number of events in D_{31} do not meet this requirement, and hence are not considered. In addition, we also remove events that have less than 50 tweets, or whose time series have a maximum value of less than 10—never did 10 or more tweets get posted about the event within one minute. Finally, 62 events satisfy all three requirements, and the estimates of the Hurst parameter for the corresponding time series are shown in Fig. 2.

As can be seen from the figure, five out of the 186 estimates are below 0.5. We believe that these outliers can be due to a lack of data for the five events: three of the time series have only 64 data points, while the other two have 128. For all events the length of whose time series is equal to or larger than 256, the estimates are all within the rage of 0.5 and 1.

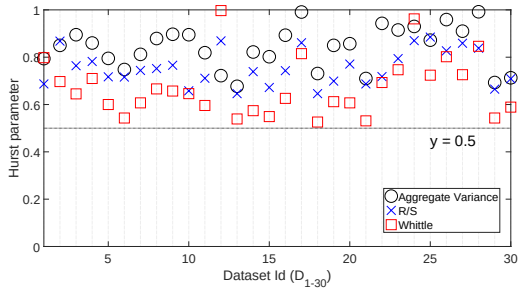
We also calculate the Hurst parameter for the time series of the whole dataset of D_{31}, since the majority of the 120 million tweets are not associated with the 506 events, and the estimates are also between 0.5 and 1.

C. *Self-similarity in* D_{32} & D_{33}

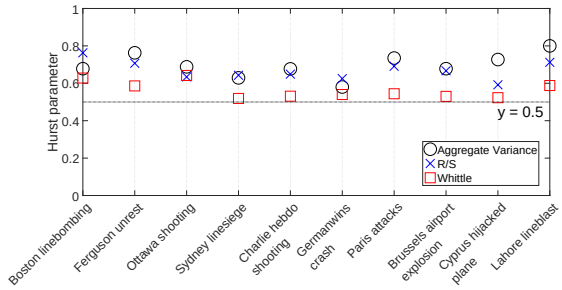
We further test self-similarity in the datasets of geo-tagged tweets collected from New York and Melbourne. The statistics in Table I suggest that the time series generated from these two datasets also show self-similarity. Specifically, we not only check the overall case (level 0), but also zoom into sub-regions by recursively dividing the area into four equal parts (levels 1 to 3). The results indicate that in the city and all sub-region levels, their corresponding time series are self-similar. Table I lists part of the statistics.



(a) All tweets.



(b) Tweets with a location label.



(c) Local tweets (tweets close to where the event occurred), *i.e.,* for the events of (1) Boston marathon bombing, (2) Ferguson unrest, (4) Ottawa shooting, (5) Sydney siege, (6) Charlie Hebdo shooting, (7) Germanwings crash, (8) Paris attacks, (9) Brussels airport explosion, (10) Cyprus hijacked plane and (11) Lahore blast, only the tweets from Massachusetts, Ferguson, Ottawa, Sydney, Paris, France, Paris, Brussels, Cyprus, Lahore or Punjab are counted.

Fig. 1: Hurst parameter estimates for $D_1 - D_{30}$. Events: 1. Boston marathon bombing; 2. Ferguson unrest; 3. Gaza under attack; 4. Ottawa shooting; 5. Sydney siege; 6. Charlie Hebdo shooting; 7. Germanwings crash; 8. Paris attacks; 9. Brussels airport explosion; 10. Cyprus hijacked plane; 11. Lahore blast; 12. Euro 2012; 13. Ebola outbreak; 14. Hong Kong protests; 15. Refugee Welcome; 16. Panama papers; 17. Hurricane Sandy; 18. Typhoon Hagupit; 19. Hurricane Patricia; 20. Nepal Earthquake; 21. Sismo Ecuador; 22. Mexican election 2012; 23. Obama and Romney 2012; 24. Superbowl 2012; 25. SXSW 2012; 26. US election 2012; 27. Indyref 2014; 28. St. Patrick's Day 2014; 29. Brexit; 30. Irish election 2016.

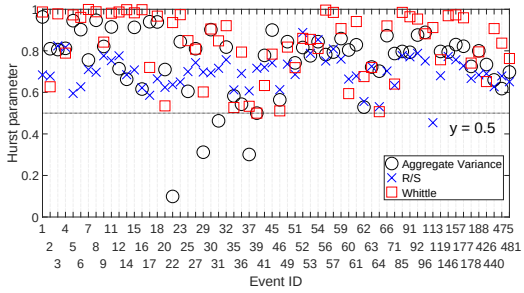


Fig. 2: Hurst parameter estimates for D_{31}. Out of the 506 events, 62 of them meet all the three requirements: (1) lasting for a minimum of 64 minutes, (2) having at least 50 tweets, and (3) the maximum value in the time series is not smaller than 10. For a detailed description of the events, please refer [16].

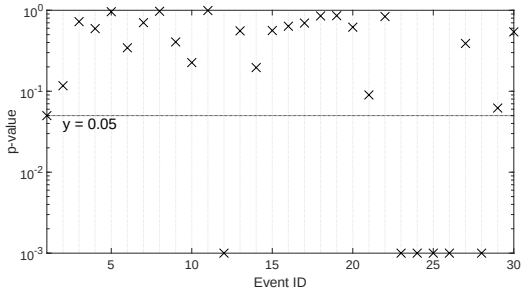
TABLE I: Hurst parameter estimates for $D_{32} - D_{33}$. Level 0: the whole area; Level 1: dividing the whole city into four equal sub-regions: 1-1, 1-2, 1-3, 1-4; Level 2: dividing sub-region 1-1 into four equal parts: 2-1, 2-2, 2-3, 2-4; Level 3: dividing sub-region 2-1 into four equal parts: 3-1, 3-2, 3-3, 3-4.

Level	D_{32}: New York			D_{33}: Melbourne		
	Aggregate Variance	R/S	Whittle	Aggregate Variance	R/S	Whittle
0	0.68	0.68	0.97	0.90	0.75	0.60
1: 1-1	0.90	0.78	0.89	0.87	0.72	0.57
1: 1-2	0.90	0.80	0.84	0.87	0.72	0.57
1: 1-3	0.92	0.80	0.84	0.79	0.66	0.55
1: 1-4	0.90	0.79	0.87	0.82	0.67	0.56
2: 2-1	0.88	0.80	0.85	0.85	0.70	0.56
2: 2-2	0.90	0.81	0.82	0.85	0.71	0.56
2: 2-3	0.90	0.82	0.77	0.77	0.66	0.55
2: 2-4	0.89	0.82	0.79	0.78	0.65	0.55
3: 3-1	0.85	0.82	0.82	0.81	0.67	0.55
3: 3-2	0.86	0.82	0.77	0.80	0.68	0.55
3: 3-3	0.84	0.84	0.71	0.76	0.65	0.56
3: 3-4	0.87	0.82	0.75	0.72	0.59	0.54

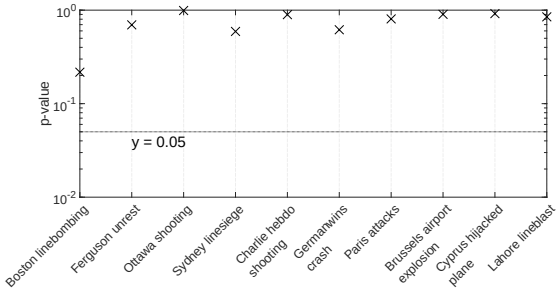
In summary, our results in this section demonstrate that ***self-similarity widely exists in different types of Twitter datasets, in terms of the number of tweets posted every minute***. This conclusion can be extended to different time intervals due to the self-similarity. For example, for D_{32} & D_{33}, since each tweet's exact publication time (to the precision of a second) is known, we also check the time series of the number of tweets posted every 10 and 100 seconds, and the results also indicate self-similarity.

IV. Existence of Power-law Distribution in Event Tweet Streams

In this section, we first examine whether the generated time series from Twitter datasets follow a power-law distribution. If this is the case, it explains self-similarity—a time series that follows a power-law distribution is also self-similar. Second, we reveal an important finding that when an event occurs it is much more likely to observe a power-law distribution in the tweet stream, compared with when no event occurs. This finding suggests that the existence of a power-law distribution can be used to help event detection from tweet streams.



(a) Tweets with a location label. For Events 12 (Euro 2012), 23 (Obama and Romney), 24 (Superbowl 2012), 25 (SXSW 2012), 26 (US election) and 28 (St. Patrick's Day 2014), the original p-values of 0.0 are replaced with 0.001, in order for them to be plotted.



(b) Local tweets (tweets with a location label that is close to where the event occurred) only.

Fig. 3: P-values of the significance test for $D_1 - D_{30}$. A p-value of less than 0.05 indicates that the power-law hypothesis should be rejected.

Recall that in order to test the power-law hypothesis, we follow the approach introduced in Section II: run the significance test, calculate the p-value for the fitted power-law model, and reject the hypothesis if the p-value is smaller than 0.05.

A. Power-law Distribution in $D_1 - D_{30}$

We still start with the 30 datasets of real-world events ($D_1 - D_{30}$). However, we only check the existence of power-law distribution for tweets with a location label, which are useful in local event detection.

Fig. 3 shows the p-values for the time series of (1) the tweets with a location label (Fig. 3a), and (2) the tweets that are close to where the 10 short-term and unplanned outbursts have occurred (Fig. 3b). We can see that in the first case, 24 out of 30 time series pass the significance test, while in the second case, all the 10 time series are with a p-value larger than 0.05. This indicates that when an event occurs, there is high probability that a power-law distribution can be detected in the geo-tagged tweet stream from the surrounding areas.

B. Power-law Distribution in D_{31}

We continue the test of a power-law distribution for the dataset D_{31} of over 500 events to further verify the above

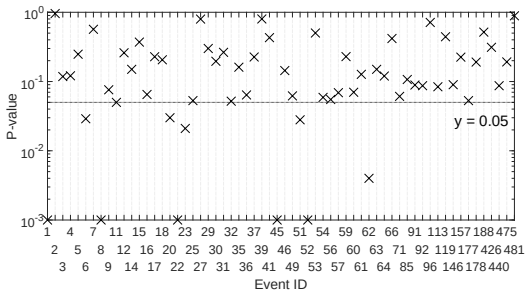


Fig. 4: P-values of the significance test for D_{31}. For Events 1, 8, 22, 45 and 52, the original p-values of 0.0 are replaced with 0.001, in order for them to be shown in the figure.

finding. Note that we only examine the tweets associated with an event according to the provided ground truth. In addition, although most of the tweets in D_{31} do not have a location label, the majority of the events are regional, and hence it is likely that most of the tweets are posted close to where the events have occurred.

As can be seen from Fig. 4, 52 out of the 62 time series pass the significance test. The above statistics in Figs. 3-4 indicate that ***when an event happens, it is likely that the time series corresponding to the geo-tagged tweets from surrounding areas follows a power-law distribution, and hence exhibits self-similarity.***

Up till now, we have only considered tweets of certain events. However, can a power-law distribution be observed as well when no event occurs (*i.e.,* false positives)? In order to answer this question, we randomly extract 100 two-hour intervals from all tweets with a location label, remove those tweets that are associated with any of the 506 events, and check whether a power-law distribution can be detected within the 100 time series corresponding to the remaining tweets, where each time series counts the number of tweets posted every minute during the two-hour interval. The result shows that only 21 of the time series follow a power-law distribution—the percentage is much lower than when an event occurs.

Finally, we consider the overall case where all tweets are mixed together, no matter if they are associated with any event or not, and check whether a power-law distribution can be detected to further examine the probability of false positives. Specifically, we extract 1000 two-hour intervals (potentially with overlap) randomly from all tweets with a location label, and then validate the existence of a power-law distribution in the generated time series. In this case, 25.0% of them pass the significance test, which is also obviously lower than the percentage when an event occurs.

C. Power-law Distribution in D_{32} & D_{33}

In order to verify the last observation in the above subsection, *i.e.,* the overall case, we further test datasets D_{32} & D_{33}, both of which include all geo-tagged tweets from a certain area, not just specific to any events. We take the same approach by randomly selecting 1000 two-hour windows from each dataset, and run the significance test on the corresponding time series. The results are listed in Table II. We believe the high percentage for D_{32} (New York) at Level 0 is because there are significantly more tweets in this dataset, which contains too much noise and leads to false positives. In fact, as can be seen in Table II, if we zoom into a sub-region of New York, and generate the time series by counting the tweets only from there, the percentage decreases quickly.

TABLE II: Percentage of test windows where a power-law distribution can be observed at different spatial scales in the datasets of $D_{32} - D_{33}$.

	D_{32}: New York	D_{33}: Melbourne
Level 0: overall	48.1	5.2
Level 1: sub-region 1-1	17.1	0.2
Level 1: sub-region 1-2	10.7	0.6
Level 1: sub-region 1-3	12.2	0.0
Level 1: sub-region 1-4	14.0	0.0
Level 2: sub-region 2-1	11.4	0.1
Level 2: sub-region 2-2	7.0	0.2
Level 2: sub-region 2-3	17.1	0.0
Level 2: sub-region 2-4	11.8	0.0
Level 3: sub-region 3-1	10.8	0.1
Level 3: sub-region 3-2	9.9	0.0
Level 3: sub-region 3-3	21.2	0.0
Level 3: sub-region 3-4	10.6	0.0

The above experimental results suggest that ***for a collected set of tweets, if a considerable portion of them are about a certain event, then a power-law distribution is likely to be observed in the corresponding time series***. Therefore, we propose to use the existence of a power-law distribution to help detect or verify events from geo-tagged tweet streams. In the next section, we test this idea by building a simple event detection algorithm that ignores the content of a tweet, but only counts the number of tweets posted during a short time period at different geographic scales, and checks whether it follows a power distribution.

V. Application in Event Detection

This section aims to apply the previous finding of the correlation between the occurrence of an event and a power-law distribution in tweet streams for event detection. We first propose an algorithm *Power-law basic* and show that by checking power-law distributions alone, it can achieve comparable results to more complex algorithms that use semantic analysis in addition to spatial clustering, *e.g.*, Geoburst [7], a popular state-of-the-art event detection algorithm. Then we integrate semantic analysis with power-law verification, and show that this improved version, *Power-law advanced*, can achieve significantly better performance.

A. Power-law Basic: Power-law based Multi-scale Spatial Event Detection

We start with a brief problem definition of event detection from tweet streams. For a certain region R, given a stream of tweets $T = \{t_1, t_2, ..., t_n\}$ and a query window $W = \{t_{n-m+1}, t_{n-m+2}, ..., t_n\}$ (m is the number of tweets in W)

that represents currently observed tweets, the aim is to identify a set of tweets $T_i \subseteq W$ that are associated with an event as close to where and when the event occurs as possible.

To solve the above problem, we propose to create a Quad-tree (QT) for each W, the root of which represents the whole region R. If m exceeds the predefined threshold m_s, QT divides R into four equally sized sub-regions, and the process continues until the number of tweets in each leaf node is not larger than the threshold, or the depth of QT reaches the maximum value, *i.e.,* the size of a sub-region has to be larger than a certain value. Once the Quad-tree is built, the detection will be run at all levels, which mitigates the impact of the arbitrary division of space.

As shown in Algorithm 1, we check for the existence of a power-law distribution in each node of QT: for a node N, (1) collect tweets from all children nodes recursively (note that once a node is divided, it does not hold any tweet itself, as all its tweets are moved to one of the four child nodes); (2) divide the query window into multiple time intervals of d seconds, and count the number of tweets posted in each interval to generate the time series S (here d does not need to be 60 as in our previous experiments, *e.g.*, a time series of tweets posted every 30 seconds should still follow a power-law distribution); (3) fit a power-law model to S; (4) run the significance test and calculate the p-value; (5) reject the power-law hypothesis if the p-value is less than 0.05, otherwise create a new event with all the tweets and append it to the final result; (6) repeat (1)-(5) for each child node at the lower levels, so that an event can be detected as close to where it happens as possible.

1) Delay in the Validation of a Power-law Distribution: An important question is: how many data points need to be observed (n_{min}), *i.e.,* the minimum length of the time series, or the delay, to verify a power-law distribution? The answer impacts two important parameters in the above algorithm: the length of the query window, l (in seconds; l is different from m, which is the number of tweets in a query window), and the length of the time interval, d (in seconds), since $n_{min} = l/d$. For example, if $n_{min} = 100$ and $l = 600$, then in order to obtain 100 data points from each query window, the algorithm divides the window into 100 intervals, *i.e.,* counts the number of tweets posted every 6 seconds.

Our experiments on datasets $D_1 - D_{31}$ suggest that when l and d are chosen properly, so that the majority of the elements in the time series are above zero, then the power-law distribution can be verified using the first 60 data points, *i.e.,* $n_{min} = 60$. Normally, a larger value of n_{min} contributes to a lower false positive rate, but too large a value causes few events to be found, which decreases the precision. In the following experiments, we set $60 \leq n_{min} \leq 300$ *and* $1200 \leq l \leq 3600$. A more detailed sensitivity analysis is given in the next subsection.

Algorithm 1: Power-law basic: power-law based multi-scale spatial event detection

Input : Geo-tagged tweets in the query window, W;
Maximum depth of the Quad-tree (QT), D;
Threshold for splitting a node in QT, m_s;
Length of the query window, l;
Time interval to count tweets, d (seconds)
Output : Event list, E

```
Phase 1: Build Quad-tree
    Create an empty Quad-tree QT;
    for tweet t in W do
        if child nodes != NULL then
            Insert t into one of the child nodes based on t's coordinates;
        else if the number of tweets in the current node ≥ m_s && QT's depth < D then
            Split the current node into four nodes;
            Move all tweets including t into one of the four child nodes according to the coordinates;
        else
            Insert t into the current node;
Phase 2: Multi-scale spatial event detection f(N, l)
    for node N in QT do
        Collect tweets from all children nodes recursively;
        Generate the time series S: divide l into multiple intervals of d seconds and count the number of tweets posted during each interval;
        Fit a power-law model to S;
        Run the significance test, and calculate the p-value for the fitted model;
        if p-value < 0.05 then
            Reject the power-law hypothesis;
        else
            Create a new event, append all tweets from children nodes to it, and insert it to E;
        // Detect events at lower levels recursively
        for N' in child nodes do
            E.add(f(N', l))
return E
```

B. Experimental Verification

In order to demonstrate the performance of Algorithm 1, we have tested it against Geoburst on three datasets (we did not choose the improved versions of Geoburst+ [8] and TrioVecEvent [9] because they use supervised approaches, while both Geoburst and our method use unsupervised approaches. In addition, the purpose here is just a proof of concept that power-law verification can be used for event detection):

- All geo-tagged tweets from Melbourne in Jan 2017, with a size of 23.3K;
- All geo-tagged tweets from Los Angeles between 9 February and 22 February 2019, with a size of 13.2K;

- All geo-tagged tweets from Sydney between 12 February and 5 April 2019, with a size of 28.4K.

These three datasets have different levels of event density: Melbourne > LA > Sydney, and we intend to check the performance of our method in all these settings. Specifically, the Melbourne dataset contains the event of "Melbourne car attack" [18], which is the type of event that we are most interested in detecting.

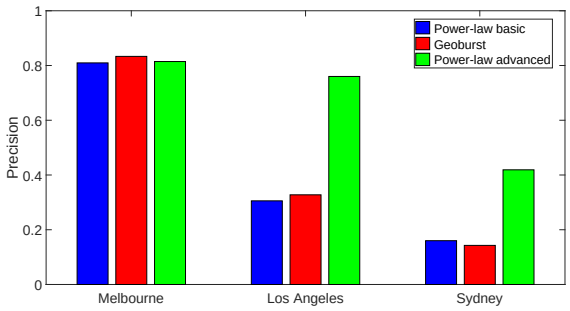
The reason why we do not use D_{31}, although it provides the ground truth, is the lack of accurate location information and publication time: the location is normally a city/town name, and the publication time only has a precision of a minute. As a result, in order to generate a time series with a minimum length of 60, it is necessary to collect tweets for 60 minutes, and since the Quad-tree cannot divide the root node due to the missing coordinates, the algorithm always needs to check the power-law distribution at level 0 against hundreds of thousands of tweets from worldwide, which contains too much noise.

1) Quantitative Analysis: Depending on the density of the data, the parameters are chosen as follows to ensure that there are an appropriate number of tweets in each query window, and sufficient elements in the generated time series are above zero: (1) for the dataset collected from Melbourne, each query window is set to 30 minutes, *i.e.,* $l = 1800$, n_{min} is set to 80, and $m_s = 15$; (2) for the dataset collected from LA, $l = 1200$, $n_{min} = 150$, and $m_s = 50$; (3) for the dataset collected from Sydney, $l = 3600$, $n_{min} = 100$, and $m_s = 50$. To make the results comparable, we set the query window to be of the same length for Geoburst, and all other parameters take the default values in the code shared by the author. For each of the three datasets, we run both algorithms on consecutive query windows covering the whole period. For example, the Melbourne dataset lasts 31 days, so there are 31×48 = 1488 query windows.

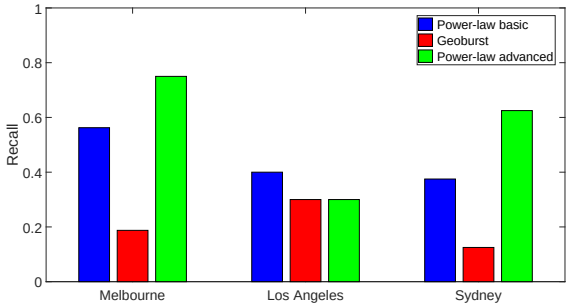
Fig. 5 presents the performance comparison between the two algorithms, which suggests that even though *Power-law basic* does not check the content of each tweet, it achieves comparable performance with Geoburst, in terms of both precision and recall. Note that since the ground truth of the three datasets are not given, it is difficult to calculate the true recall. Therefore, we adopt a similar approach as in [8], [9] and calculate the $pseudo\ recall = N_{true}/N_{total}$, where N_{true} is the number of true events detected by a method, and N_{total} is the number of true events detected by all methods, plus the events hand-picked by us that occurred during the query periods within the chosen cities, including festivals, sport games, natural disasters, etc. Note also that the validity of each event is checked manually.

However, we are not claiming that it is sufficient to detect events just by checking the existence of a power-law distribution, and we further improve the algorithm in Section V-C.

Sensitivity analysis on n_{min}. Fig. 6 shows how n_{min} impacts the number of different events detected and the precision for the dataset of Melbourne, when l is set to 30 minutes and $m_s = 15$. As n_{min} first increases, both the reported events and false positives decrease, and the false positive count decreases



(a) Precision.



(b) Recall.

Fig. 5: Performance comparison of the three event detection algorithms.

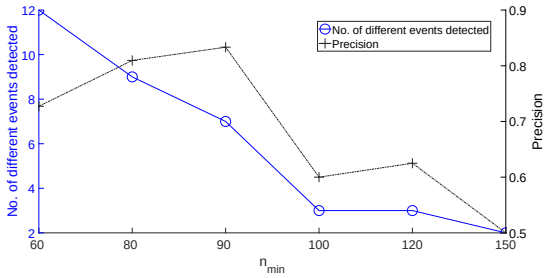


Fig. 6: Sensitivity analysis on n_{min} for the Melbourne dataset. The length of the query window l is set to 30 minutes, so $n_{min} = 60, 80, 90, 100, 120, 150$ corresponds to counting the number of tweets every 30, 22.5, 20, 18, 15, 12 seconds.

faster, so the precision improves. However, as n_{min} gets too large, *i.e.,* counting the number of tweets too frequently, too many elements of the generated time series become zero, causing too few events to be detected, and the precision drops.

Sensitivity analysis on m_s. We further analyse the impact of m_s. Specifically, Fig. 7 shows for the dataset of Melbourne, when $l = 1800$ and $n_{min} = 80$, how the number of different events detected and the total detection time change with m_s. A small m_s value means a larger depth of the Quad-tree, and since the detection is running at each node, the total detection time will be longer, but meanwhile more events are likely to be found.

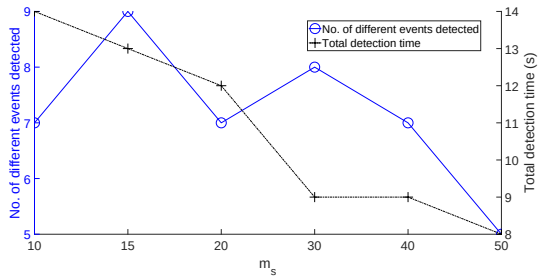


Fig. 7: Sensitivity analysis on m_s for the Melbourne dataset. The length of the query window l is set to 1800 (30 minutes), and n_{min} is set to 80.

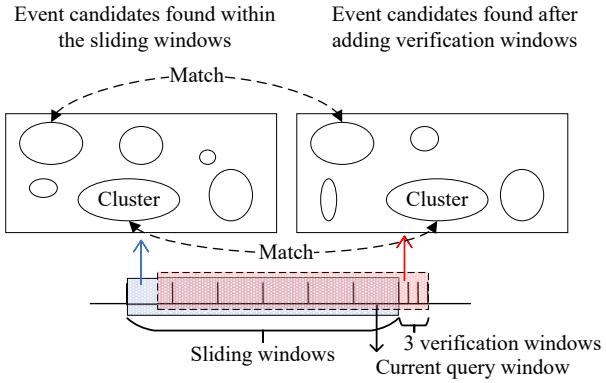


Fig. 8: An illustration of *Power-law Advanced*.

C. Power-law Advanced: Combining Semantic Analysis with Power-law Verification

In order to further improve the performance of the proposed method, we investigate how semantic information can be incorporated to the event detection algorithm.

A common class of existing methods that use semantic information is clustering based approaches, where the first step is to cluster posts/tweets according to their semantic, spatial, temporal, frequency information, etc., and then generate a list of event candidates. Once the candidates are found, the second step is to remove non-event clusters among them. Our finding in this paper suggests that checking the existence of a power-law distribution can be used in the second step to test whether a cluster of tweets are about a real event or not.

In order to demonstrate the feasibility of the above approach, we design an algorithm *Power-law Advanced* that combines fastText (the latest word embedding tool developed by Facebook) [19], BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [20], and power-law verification, where fastText is for embedding the tweets so that semantically similar tweets would also end up close in the vector space, and BIRCH is for clustering the generated vectors. These two methods are chosen for demonstration purposes only, and they can be replaced by other alternatives.

In addition, sliding windows are used: the algorithm keeps the latest N_{SW} query windows, performs event detection, and discards the oldest window while collecting new tweets. In the following experiment, N_{SW} is set to 6, and the size of a query window is set to 30 minutes.

Specifically, the algorithm (Algorithm 2: *Power-law Advanced*) works as described below (also see Fig. 8 for an illustration):

- Embedding. The same NLP tool [21] as mentioned in [7] is used to extract entities and noun phrases from the tweets. These generated keywords are then embedded with the fastText algorithm, and each tweet is represented by the average value of the vectors from all its keywords. A pre-trained fastText model is used in our experiment, and it is re-trained incrementally with the new tweets [22]. The re-training is done in parallel, and hence does not delay the detection. Note that the spatial and temporal information is not included in the embedding since the Quad-tree and sliding windows ensure the similarity in terms of space and time.
- Clustering. Once the tweets are embedded into vectors, we use the BIRCH algorithm to cluster the vectors. The most important parameter in BIRCH is the threshold of the cluster radius. In our experiment we do not directly set a fixed value. Instead, we start with a value close to zero, and increase it by a small step size until either (1) less than 5% of all items are in small clusters, *i.e.,* clusters with a size less than 10, or (2) over half of the items are in the largest cluster, whichever occurs first.
- Power-law detection. The third step is to detect any power-law distribution within each cluster. Note that a Quad-tree is still built and maintained, and the detection is run at all levels of the Quad-tree to mitigate the impact of the arbitrary division of space.
- Verification. If any event candidate is found in the last step, we further collect tweets from the verification window which is set to 5 minutes in our experiment, and repeat the above three steps. The only difference is that the keywords are no longer used, and the original text of each tweet is directly embedded—the rationale is to ensure that both the keywords and texts are semantically close within a cluster. Each event candidate from the last step is then checked against each cluster found in this step. If any two of them share more than half of the tweets, they are considered as a match. If no match is found for a candidate, it will be removed. The verification process is done three times, and an event candidate has to pass all three of them.
- Final clean-up. To further decrease the false positive rate, the last step extracts the top $X(= 10)$ hashtags and mentions for each cluster, and if more than half of the tweets contain any of these hashtags or mentions, the cluster is finally considered as an event.

We test the above algorithm on the same three datasets, and as can be seen in Fig. 5, this semantic analysis enhanced power-law verification method increases both the precision and the recall in most cases.

Algorithm 2: Power-law advanced: integrating power-law verification with semantic analysis

Input : same as *Power-law basic*
Output : Event list, E

```
Extract entities and noun phrases using the NLP tool [21] for each tweet;
Call fastText to embed the extracted keywords;
Cluster the generated vectors using BIRCH;
for Cluster c found in the last step do
    E.add(Power-law basic(·));
for i = 0; i < 3 && E is not NULL do
    Call fastText to directly embed the text of each tweet;
    Cluster the generated vectors using BIRCH;
    for Cluster c′ found in the last step do
        E′.add(Power-law basic(·));
    for Remaining event candidate e ∈ E do
        if there is no match in E′ then
            Remove e;
for Remaining event candidate e ∈ E do
    K ← Extract the top X(= 10) hashtags and mentions;
    if More than half of the tweets in e does not contain any element in K then
        Remove e;
return E
```

In summary, this section has demonstrated that the naive algorithm of checking the existence of a power-law distribution can achieve comparable results against more advanced event detection methods, and its performance can be significantly improved by integrating with semantic analysis.

VI. Related Work

This section briefly reviews the previous work on event detection from social media. Specifically, we take a similar approach as [23] and summarise two types of algorithms: clustering based and anomaly based. In addition, multiscale event detection is also considered.

A. Clustering based Event Detection

This type of detection method takes into consideration all or a subset of temporal, spatial, semantic, frequency and user information to cluster the tweets [7]–[9], [24]–[31]. However, since the generated clusters may correspond to non-events, normally another step is taken to eliminate false positives, *e.g.*, by ranking the candidates based on certain criteria, or training a classifier to decide whether a candidate is a real event.

For example, for each pair of tweets, Geoburst [7] uses the Epanechnikov kernel to calculate their geographical impact, and uses the random-walk-with-restart algorithm to to obtain the semantic impact. In this way, they identify a list of clusters of geographically close and semantically coherent tweets, *i.e.,* event candidates. Finally, historical activities are used to rank these candidates and the top K events are returned. As the improved versions, (1) Geoburst+ [8] replaces the ranking algorithm in Geoburst with a candidate classification module, which learns the latent embeddings of tweets and keywords. Then together with the activity timeline, the module extracts spatial unusualness and temporal burstiness to characterise each candidate event; (2) TrioVecEvent [9] learns multimodal embeddings of the location, time and text, and then performs online clustering using a Bayesian mixture model.

B. Anomaly based Event Detection

This type of method aims to identify abnormal observations in word usage, spatial activity, sentiment levels, etc. For example, Valkanas and Gunopulos [32], [33] use sentiment analysis for event detection, which is based on the idea that the sentiment level fluctuates as people respond to an event to express their opinions. Another example is to detect peaks in Twitter hashtags using a Discrete Wavelet Transformation [34], since these peaks are likely to correspond to real-world events. Specifically, only the hashtags are used, and all the remaining tweet text is discarded. In addition, Vavliakis *et al.* [35] propose Latent Dirichlet Allocation based event detection for MediaEval Benchmark 2012 [36], where the dataset contains 167,000 images from Flickr. They detect peaks in the number of photos assigned to each topic, and identify an event for a topic if it receives an unexpectedly high number of photos.

C. Multiscale Event Detection

Running event detection on a fixed spatial resolution may not help in finding events at different scales. For example, using low resolution spatial data might only capture events occurring on the state or the country level, while high resolution data can help detect events at community or city scales. Therefore, another stream of work intends to detect events at different space resolutions, to better adapt to the unpredictability of real-life events [37]–[39]. For example, Dong *et al.* [37] explore the properties of the wavelet transform for the detection of events at different spatio-temporal scales. In addition, Visheratin *et al.* [39] build a convolutional quad-tree, which instead of dividing a region into four sub-regions of equal size, uses a convolutional neural network to decide a more appropriate division.

VII. Conclusions and Future Work

In this paper, we have (1) verified in more than 30 datasets the existence of self-similarity in the time series of the number of geo-tagged tweets posted within a short time interval from a certain region; (2) demonstrated that a power-law distribution is much more likely to be observed when an event occurs in tweet streams; (3) proposed two event detection algorithms: *Power-law basic* and *Power-law advanced*. *Power-law basic* is based on the validation of power-law distributions at multi-spatial scales, without checking the content of each tweet, or using any information other than the geo-location. Experimental results on multiple datasets show that it can achieve comparable performance with Geoburst, a widely

cited event detection algorithm. *Power-law advanced* improves the algorithm by incorporating semantic analysis via word embedding, and our results demonstrate that it can significantly increase both the precision and the recall.

For future work, we will further study the self-similar patterns in tweet streams. Our current result explains why when an event occurs the corresponding time series shows self-similarity—it follows a power-law distribution. However, we have not examined why the tweet count time series still exhibits self-similarity when there is no event.

In addition, as a separate direction, we will explore other potential ways for embedding and clustering to futher improve the performance of the event detection algorithm. Specifically, in terms of embedding, we are considering (1) dispensing with the Quad-tree and directly embed the location information; (2) representing a tweet using other methods rather than the average value of the vectors for each word that it contains.

ACKNOWLEDGEMENT

This research is funded in part by the Defence Science and Technology Group, Edinburgh, South Australia, under contract MyIP:7293.

REFERENCES

[1] M. Crovella and A. Bestavros, "Explaining world wide web traffic self-similarity," Boston University, Tech. Rep., 1995.

[2] W. Willinger, M. S. Taqqu, W. E. Leland, and D. V. Wilson, "Self-similarity in high-speed packet traffic: Analysis and modeling of ethernet traffic measurements," *Statistical Science*, vol. 10, 1995.

[3] T. Karagiannis and M. Faloutsos, "SELFIS: A tool for self-similarity and long-range dependence analysis," in *1st Workshop on Fractals and Self-Similarity in Data Mining: Issues and Approaches (IN KDD)*, 2002.

[4] T. Karagiannis, M. Faloutsos, and M. Molle, "A user-friendly self-similarity analysis tool," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 81–93, 2003.

[5] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Inf.*, vol. 4, no. 1, pp. 1–9, 1974.

[6] H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surv.*, vol. 16, no. 2, pp. 187–260, 1984.

[7] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. Kaplan, S. Wang, and J. Han, "GeoBurst: Real-time local event detection in geo-tagged tweet streams," in *Proceedings of SIGIR*. ACM, 2016, pp. 513–522.

[8] C. Zhang, D. Lei, Q. Yuan, H. Zhuang, L. Kaplan, S. Wang, and J. Han, "GeoBurst+: Effective and real-time local event detection in geo-tagged tweet streams," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 3, pp. 34:1–34:24, 2018.

[9] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han, "TrioVecEvent: Embedding-based online local event detection in geo-tagged tweet streams," in *Proceedings of the 23rd ACM SIGKDD*. ACM, 2017, pp. 595–604.

[10] H. Hurst and A. S. of Civil Engineers Hydraulics Division, *Long-term Storage Capacity of Reservoirs*. American Society of Civil Engineers, 1950. [Online]. Available: https://books.google.com.au/books?id=aKj4HAAACAAJ

[11] H. Hurst, R. Black, and Y. Simaika, *Long-term storage: an experimental study*. Constable, 1965. [Online]. Available: https://books.google.ru/books?id=7QgSMwEACAAJ

[12] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, 2009.

[13] Y. Virkar and A. Clauset, "Power-law distributions in binned empirical data," *The Annals of Applied Statistics*, vol. 8, no. 1, pp. 89–119, 2014.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[15] A. Zubiaga, "A longitudinal assessment of the persistence of twitter datasets," *Journal of the Association for Information Science and Technology*, vol. 69, no. 8, pp. 974–984, 2018.

[16] A. J. McMinn, Y. Moshfeghi, and J. M. Jose, "Building a large-scale corpus for evaluating event detection on twitter," in *Proceedings of the 22nd ACM CIKM*. ACM, 2013, pp. 409–418.

[17] P. Symeon, "Twitter-dataset-collector," https://github.com/socialsensor/twitter-dataset-collector, 2019.

[18] "January 2017 melbourne car attack," 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=January_2017_Melbourne_car_attack&oldid=894478928

[19] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv:1607.04606 [cs]*, 2016.

[20] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proceedings of ACM SIGMOD '96*. ACM, 1996, pp. 103–114, montreal, Canada.

[21] A. Ritter, "Twitter NLP tools. contribute to aritter/twitter_nlp development by creating an account on GitHub," 2011. [Online]. Available: https://github.com/aritter/twitter_nlp

[22] QinLuo, "Library for fast text representation and classification.: ericxsun/fastText," 2019. [Online]. Available: https://github.com/ericxsun/fastText

[23] N. Panagiotou, I. Katakis, and D. Gunopulos, "Detecting events in online social networks: Definitions, trends and challenges," in *Solving Large Scale Learning Tasks. Challenges and Algorithms*. Springer International Publishing, 2016, pp. 42–84.

[24] H. Becker, M. Naaman, and L. Gravano, "Beyond trending topics: Real-world event identification on twitter," in *ICWSM 2011*, 2011.

[25] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang, "TEDAS: A twitter-based event detection and analysis system," in *Proceedings of the 28th IEEE ICDE*. IEEE Computer Society, 2012, pp. 1273–1276.

[26] O. Ozdikis, P. Senkul, and H. Oguztzn, "Semantic expansion of tweet contents for enhanced event detection in twitter," in *ASONAM*, 2012, pp. 20–24.

[27] H. Abdelhaq, C. Sengstock, and M. Gertz, "EvenTweet: Online localized event detection from twitter," *Proc. VLDB Endow.*, vol. 6, no. 12, pp. 1326–1329, 2013.

[28] G. Fuchs, N. Andrienko, G. Andrienko, S. Bothe, and H. Stange, "Tracing the german centennial flood in the stream of tweets: First lessons learned," in *Proceedings of GEOCROWD '13*. ACM, 2013, pp. 31–38.

[29] M. Walther and M. Kaisser, "Geo-spatial event detection in the twitter stream," in *Advances in Information Retrieval*. Springer Berlin Heidelberg, 2013.

[30] W. Xie, F. Zhu, J. Jiang, E. Lim, and K. Wang, "TopicSketch: Real-time bursty topic detection from twitter," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 8, pp. 2216–2229, 2016.

[31] H. Wei, H. Zhou, J. Sankaranarayanan, S. Sengupta, and H. Samet, "Detecting latest local events from geotagged tweet streams," in *Proceedings of the 26th ACM SIGSPATIAL*. ACM, 2018, pp. 520–523.

[32] G. Valkanas and D. Gunopulos, "Event detection from social media data," *IEEE Data Eng. Bull.*, vol. 36, no. 3, pp. 51–58, 2013. [Online]. Available: http://sites.computer.org/debull/A13sept/p51.pdf

[33] ——, "How the live web feels about events," in *Proceedings of the 22nd ACM CIKM*. ACM, 2013, pp. 639–648.

[34] M. Cordeiro and R. Frias, "Twitter event detection: combining wavelet analysis and topic inference summarization," in *Doctoral Symposium on Informatics Engineering, DSIE*, 2011.

[35] K. N. Vavliakis, F. A. Tzima, and P. A. Mitkas, "Event detection via LDA for the MediaEval2012 SED task," in *MediaEval*, 2012.

[36] (2019) MediaEval 2012. [Online]. Available: http://www.multimediaeval.org/mediaeval2012/

[37] X. Dong, D. Mavroeidis, F. Calabrese, and P. Frossard, "Multiscale event detection in social media," *Data Min. Knowl. Discov.*, vol. 29, no. 5, pp. 1374–1405, 2015.

[38] J. Capdevila, G. Pericacho, J. Torres, and J. Cerquides, "Scaling DBSCAN-like algorithms for event detection systems in twitter," in *Algorithms and Architectures for Parallel Processing*. Springer International Publishing, 2016, pp. 356–373.

[39] A. A. Visheratin, K. D. Mukhina, A. K. Visheratina, D. Nasonov, and A. V. Boukhanovsky, "Multiscale event detection using convolutional quadtrees and adaptive geogrids," in *Proceedings of LENS '18*. ACM, 2018, pp. 1–10.