Pattern Recognition 123 (2022) 108404

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Big data directed acyclic graph model for real-time COVID-19 twitter stream detection

Bakhtiar Amen[a], Syahirul Faiz[b,*], Thanh-Toan Do[c]

[a] Department of Computer Science, School of Electrical Engineering, Electronics, and Computer Science, University of Liverpool, Liverpool L69 3BX, UK
[b] State Islamic Institute of Surakarta (IAIN Surakarta), Indonesia
[c] Department of Data Science and AI, Faculty of Information Technology, Monash University, Australia

ARTICLE INFO

Article history:
Received 28 March 2021
Revised 15 September 2021
Accepted 24 October 2021
Available online 26 October 2021

Keywords:
Anomaly detection
Big data
COVID-19
Directed acyclic graph
Event stream

ABSTRACT

Every day, large-scale data are continuously generated on social media as streams, such as Twitter, which inform us about all events around the world in real-time. Notably, Twitter is one of the effective platforms to update countries leaders and scientists during the coronavirus (COVID-19) pandemic. Other people have also used this platform to post their concerns about the spread of this virus and a rapid increase of death cases globally. The aim of this work is to detect anomalous events associated with COVID-19 from Twitter. To this end, we propose a distributed Directed Acyclic Graph topology framework to aggregate and process large-scale real-time tweets related to COVID-19. The core of our system is a novel lightweight algorithm that can automatically detect anomaly events. In addition, our system can also identify, cluster, and visualize important keywords in tweets. On 18 August 2020, our model detected the highest anomaly since many tweets mentioned the casualties' updates and the debates on the pandemic that day. We obtained the three most commonly listed terms on Twitter: "covid", "death", and "Trump" (21,566, 11,779, and 4761 occurrences, respectively), with the highest TF-IDF score for these terms: "people" (0.63637), "school" (0.5921407) and "virus" (0.57385). From our clustering result, the word "death", "corona", and "case" are grouped into one cluster, where the word "pandemic", "school", and "president" are grouped as another cluster. These terms were located near each other on vector space so that they were clustered, indicating people's most concerned topics on Twitter.

1. Introduction

In December 2019, the global pandemic of COVID-19 hit the world, and people began to worry about the rapid spread of this virus. Researchers, especially from the computer science field, have started to propose many innovative solutions for this pandemic crisis and prevent the spread of this virus [1,2]. Some have used AI and machine learning methods to detect patterns from large-scale real-time video, image, and text data. For video-based cases, researchers utilised computer vision to identify human's body temperature or detecting whether or not people wear masks [3]. As for the text-based cases, we contribute by discovering the virus outbreak or the infection cases from event stream tweets through developing a novel big data stream analytic method.

Every day, over 500 million tweets are posting on Twitter [4], in particular from early 2020, people have started to post information related to the COVID-19 on Twitter. This platform has been utilised as a real-time communication media between world leaders and their citizens, scientists and healthcare organisations. According to Qazi et al. [5], there have been an increased amount of generated real-time tweet contents associated with COVID-19 started from the early weeks of the outbreak, where it reached about 1 million tweets per day. As of April 2020, this has approximately reached 10.5 million tweets per day. This evidence showed that the use of social media, in particular Twitter, has increased during the pandemic. For this purpose, we investigate the adoption of incremental real-time pattern detection from large-scale Twitter events. Hence, we define the keyword "death" as the event, which could lead to the anomaly, and we need to investigate the source and cause of the anomaly in this research.

The aim of this paper is to detect anomalous events associated with COVID-19 from Twitter. Here, we identify our objectives for this research:

1. We build a distributed Directed Acyclic Graph topology model to aggregate large-scale real-time tweets related to COVID-19.

* Corresponding author.
E-mail addresses: Bakhtiar.Amen@liverpool.ac.uk (B. Amen), Syahirul.Faiz@gmail.com (S. Faiz), Toan.Do@monash.edu (T.-T. Do).

https://doi.org/10.1016/j.patcog.2021.108404
0031-3203/© 2021 Elsevier Ltd. All rights reserved.

2. We propose a novel algorithm that uses the predictive statistical analysis technique (i.e., “PESCAD” Algorithm) to detect anomalous events.
3. We examine the frequency and the importance of keywords to figure out what people are thinking on Twitter during this pandemic period.

We will discuss the previous related works in Section 2, and we will break down the detailed methodology of our approach in Section 3. We also illustrate the experiment of our algorithm in Section 4. Finally, we outline the discussion of our results and findings in Section 5.

2. Related works

In this research, we acknowledge the fundamental concept of anomaly or event detection from large-scale data and its applicability in real-world problems from Amen et al. [6]. The theoretical concept of large-scale anomaly detection of both batch and data streams, along with its constraints and limitations, were also discussed in Amen and Grigoris [7]. Meanwhile, the authors in Amen and Grigoris [8] have implemented collective anomaly detection on data sensor streams, where the algorithm's accuracy outperformed compared to Adaptive Stream Projected Outlier Detector (A-SPOT) algorithm. The survey about the anomaly detection technique with its various big data solution technologies is also explained in Habeeb et al. [9], including the performance of various anomaly detection algorithms such as Bayesian Network, Neural Network (NN), and Support Vector Machine (SVM). We can also understand the anomaly detection method using the Isolation Forest Algorithm from [10]. Meng Li et al. [11] proposed a k-Nearest Neighbour algorithm implementation to detect anomaly using blockchain and sensor networks.

According to Amen and Lu [12], a big data framework (e.g., Apache Storm) has been improved and outperformed well to detects large-scale abnormal events in real-time. In [13], Patel et al. introduced the sentiment-based classification to detect the anomaly in Twitter, whereas [14] discussed the sentiment analysis technique using a tree-learning algorithm using Apache Storm framework. Toshiwal et al. [15] discussed the comprehensive Twitter monitoring utility function and how to evaluate throughput as well as the performance of the framework mentioned above in McCreadie et al. [16]. Meanwhile, in Zhao et al. [17], we can observe the example implementation of Apache Storm for anomaly detection in a real-time network.

In [18], Twitter was utilised to monitor incidents such as an earthquake but without using any big data middleware. Gupta et al. [19], discussed how to identify hybrid hashtags for Twitter classification, with several machine learning classification algorithms (i.e., Naïve Bayes, k-Nearest Neighbor, and SVM), and [20] extends the experiment in big data domain with Apache Storm. Both [21,22] demonstrated the Poisson distribution's implementation for detecting the anomalies. However, in Sapegin et al. [22], the method is only used in a non-distributed environment to track log-in accounts. Meanwhile, Turcotte et al. [23] implemented the Poisson factorisation to find the anomaly in user credentials in a corporate network. Keval et al. [24] also briefly discussed anomaly detection using the Poisson probability and machine learning but also not in the distributed problem domain.

To support our understanding of the basic concept of Term Frequency-Inverse Document Frequency (TF-IDF), we acquire the fundamental concept from Manning et al. [25], as well as the method to extract the keyword using semantic association in Liu et al. [26] and learn about keyword relevance using TF-IDF in Qaiser and Ali [27]. From Amin et al. [28], we also discover a novel proposed technique for automatic monitoring for dengue disease detection, based on analysing the Twitter's statuses only, and decide whether the people are infected or not, including for dengue virus control spread.

According to Alom et al. [29], there are several ready-to-use libraries for machine learning utilisation, such as Deeplearning4j, in which we adopt this library since this library provides us with the functionality we require (i.e. Word Embedding, Clustering, and Principal Component Analysis). Meanwhile, the scalability of the previously mentioned library on the GPU-cored distributed computing is also discussed in Li et al. [30]. In [31], Doshi et al. demonstrated the implementation of the leaflet.js library for locating the user's tweets coordinates on the world map and the chart.js library function for the visualisation.

As we mentioned earlier, we acknowledge and are inspired by the research from Sapegin et al. [22], which discussed the Poisson distribution's implementation for detecting the anomalies in a company network. The research applies Point-based anomaly detection explicitly as compared to our research method, Collective anomaly detection. Apart from that, the anomaly detection method is only used in a non-distributed environment to monitor the log-in accounts where ours is on distributed computing. One interesting novelty about the previous research is how they introduce the elbow function as the anomaly detection threshold. The elbow function is one way to find a curvature in the optimum/minimum value of the function (or ‘elbow point’).

Therefore, we initially planned to design our PESCAD algorithm using the elbow function. For the computation of this threshold, the ‘death’ keyword from the last intervals will be counted, and a lower bound is defined and initialised as the upper limit of the ‘elbow function’ calculations. The method will need to calculate a second-order central difference derivative with this ‘elbow function’ where it represents the curvature of discrete data (compatible with the Poisson discrete random variable) [32]. Hence, it is expected to obtain a minimum value of the threshold using the elbow function. We did not adopt this method/function because iterating the curvature array caused a slower system and consumed enormous computing resources (memory). The previous algorithm consists of two outer for-loop. We analyse that the algorithm will have asymptotic order of magnitude (i.e., the Θ-class) as $\Theta(n^2)$. On the contrary, our proposed algorithm (Algorithm 1) will only have one outer loop (i.e., implicitly, as incoming data streams are fed continuously from Twitter). Therefore, the order of our proposed algorithm will only have asymptotic order of magnitude (i.e., the Θ-class) as $\Theta(n)$. Hence, since $\Theta(n^2) > \Theta(n)$, we assume that our algorithm is more lightweight than the previous research.

In summary, the previous related works only focused on particular study cases of abnormal behaviour, or some were too specific types of anomaly detection. On the contrary, in this research, we point out our novelty and our research contribution by proposing a lightweight solution for anomaly detection in real-time Twitter data stream by implementing the Directed Acyclic Graph model and the Poisson distribution.

3. Methods

3.1. Methodology framework

The followings are the overview of the methodology framework of our research:

3.1.1. Theory

We describe the association of our research with the big data problem in the *Big Data Problem Background* (Section 3.2). In *Directed Acyclic Graph Topology Model* (Section 3.3), we explain the topology model design of our system. Meanwhile, in *Grouping Type*

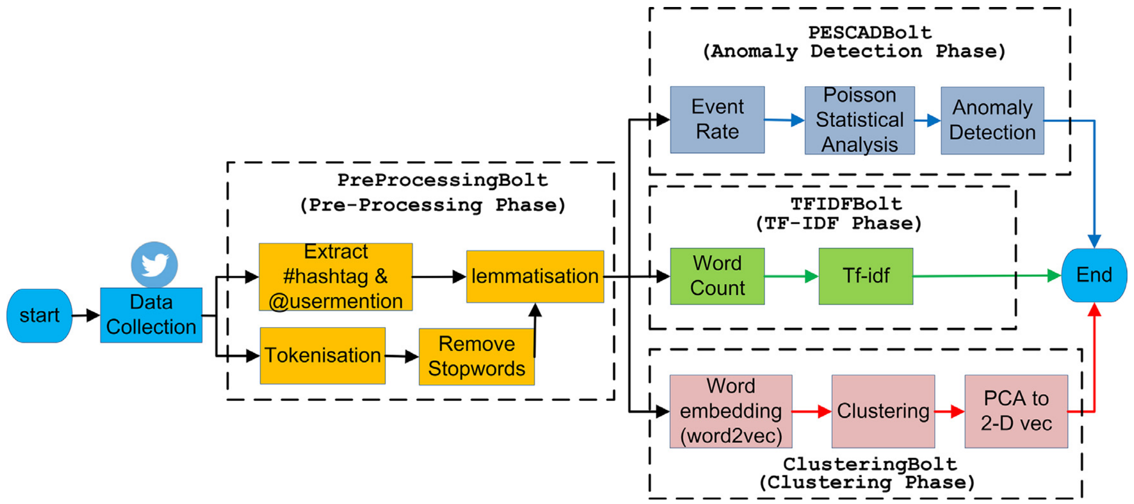


Fig. 1. Methodology framework - implementation.

Algorithm 1 PESCAD algorithm.

```
1:  procedure PESCAD($\{Tweet\}_{i=1}^{D}$)
2:      $SCIE = 0$
3:      $sumTotalEvent = 0$
4:      $sumTotalTweet = 0$
5:      for $tweet_i \in \{Tweet\}_{i=1}^{D}$ do
6:          $sumTotalTweet = sumTotalTweet + 1$
7:          if $is\_event == 1$ then
8:              $\{idStatusList\}$ ← extract $id\_status$ from the current
                  interval and put into array
9:              $SCIE \leftarrow SCIE + 1$
10:         end if
11:         if $time\_interval ==$ 1~min then
12:             $sumTotalEvent \leftarrow sumTotalEvent + SCIE$
13:             $\lambda \leftarrow \frac{sumTotalEvent}{sumTotalTweet}$
14:             $probability \leftarrow Poisson(SCIE, \lambda)$
15:             $PO \leftarrow probability * sumTotalTweet$
16:             $AO \leftarrow sumTotalEvent$
17:             if $AO > PO$ then
18:                 Mark all $\{idStatusList\}$ as collective anomaly
19:                 Print all collective anomalies
20:             end if
21:             Empty the $\{idStatusList\}$ array
22:             $SCIE \leftarrow 0$
23:         end if
24:     end for
25: end procedure
```

(Section 3.4) and *Event Stream* (Section 3.5), we outline two fundamental theories related to how and what data being passed, respectively.

3.1.2. Implementation

We state what tools/software we use to implement our theory in *System Environment* (Section 3.6). In *Data Collection* (Section 3.7), we discuss how we collect our data. After collecting data, we describe how we pre-process the data in *Pre-processing Phase* (Section 3.8) and subsequently break down the data processing in parallel: *TF-IDF Phase* (Section 3.9), *Clustering Phase* (Section 3.10), and *Anomaly Detection Phase* (Section 3.11). Overall, we illustrate our implementation elements, as we explain above, in Fig. 1.

To aid the explanation of Fig. 1, we add the sequence diagram in Fig. 2. From the sequence diagram, we can see that the actor/user initially triggers the Topology class. The Topology class then generates the *PreprocessingBolt* (i.e., it pre-processes the incoming tweet keywords from the 'Spout'). Subsequently, three other bolts (i.e., *ClusteringBolt, TFIDFBolt*, and *PESCADBolt*) are created and work in parallel, as indicated in the 'par' frame in the diagram. After that, the cluster will divide the topology instances according to the server configuration cluster (either standalone or distributed). During the runtime, the user will receive a message regarding the group of anomalous events in the *idStatusList* array from the system.

3.2. Big data problem background

There are challenges/dimensions of big data called 'the 4V's of Big Data': Volume, Velocity, Variety and Veracity. Meanwhile, there are two types of big data processing: batch-based processing and stream-based processing [33]. In batch-based processing, each data block is processed sequentially one by one in a period of time. This processing type is mainly for overcoming the 'volume' challenges of Big Data. The famous framework for this batch processing is Apache Hadoop. On the other hand, stream-based processing is always associated with 'velocity' challenges where the real-time processing of fast-growth data is needed (such as Twitter datastream). For stream processing type, Apache Storm is the forefront framework and is designed to answers the challenges in the velocity aspect. Not only to solve the velocity challenges, but the Apache Storm can also be implemented in larger organisation clusters (i.e., scalable) for online decision making. Therefore, we use Apache Storm since it is scalable and it can process a million tuples per second in real-time from Twitter [34].

3.3. Directed acyclic graph topology model

We utilise the distributed Directed Acyclic Graph topology model in our system and implement using *Apache Storm*, which consists of the *spout*(s) and *bolt*(s) (Fig. 3). The spout is the source of the event stream (i.e., tuple or data), and the bolt is for processing the tuple stream. The arrow lines in (Fig. 3) represent the directed stream of events. In terms of the event stream, Fig. 3 illustrates that the PreprocessingBolt processes the largest number of data tuple. Subsequently, the ClusteringBolt processes fewer tuples from the PreprocessingBolt and the TFIDFBolt processes even fewer tuples (i.e., output only important keywords). Lastly, the PESCADBolt processes the least number of the data tuple which related to these keywords, such as "death", "covid", and "corona".

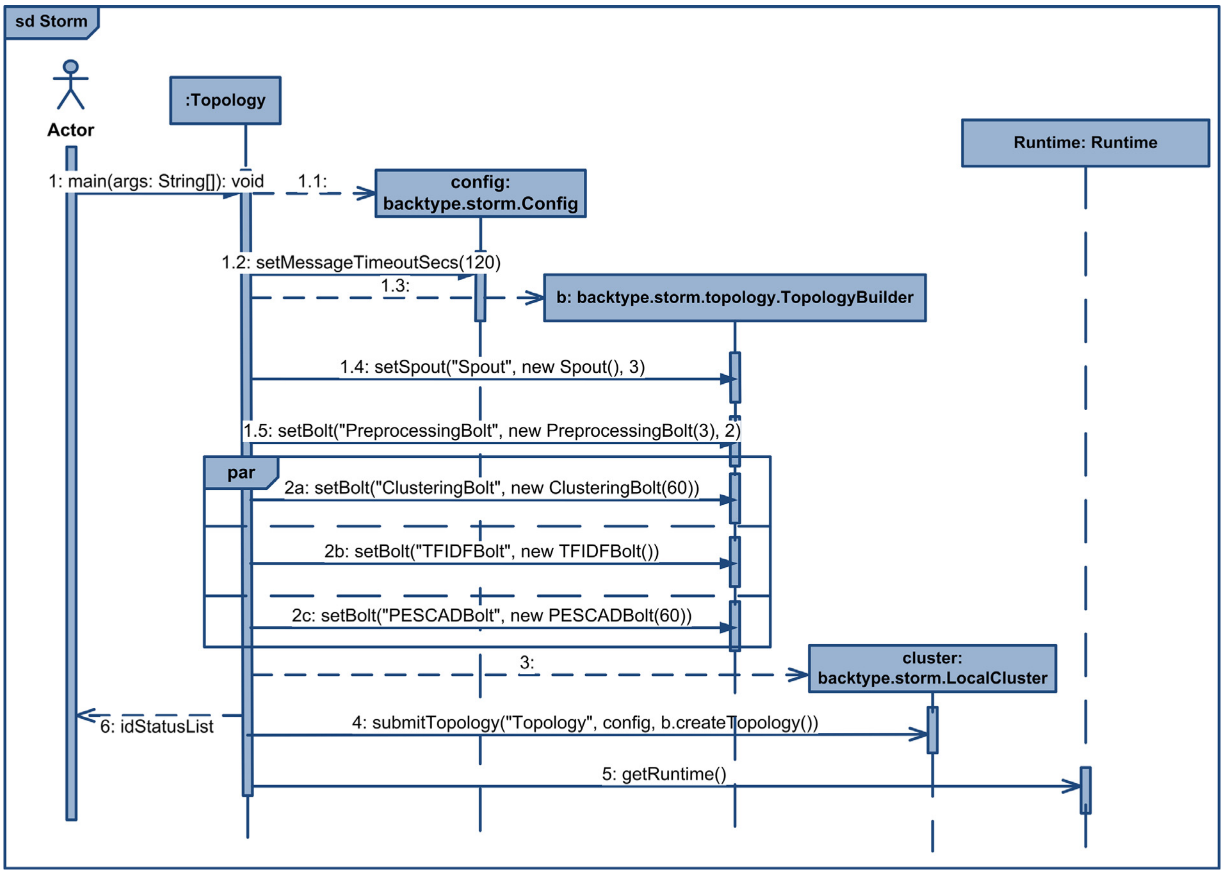

sd Storm
Actor
:Topology
config: backtype.storm.Config
Runtime: Runtime
1: main(args: String[]): void
1.1:
1.2: setMessageTimeoutSecs(120)
1.3:
b: backtype.storm.topology.TopologyBuilder
1.4: setSpout("Spout", new Spout(), 3)
1.5: setBolt("PreprocessingBolt", new PreprocessingBolt(3), 2)
par
2a: setBolt("ClusteringBolt", new ClusteringBolt(60))
2b: setBolt("TFIDFBolt", new TFIDFBolt())
2c: setBolt("PESCADBolt", new PESCADBolt(60))
3:
cluster: backtype.storm.LocalCluster
6: idStatusList
4: submitTopology("Topology", config, b.createTopology())
5: getRuntime()

Fig. 2. Sequence diagram of the system.

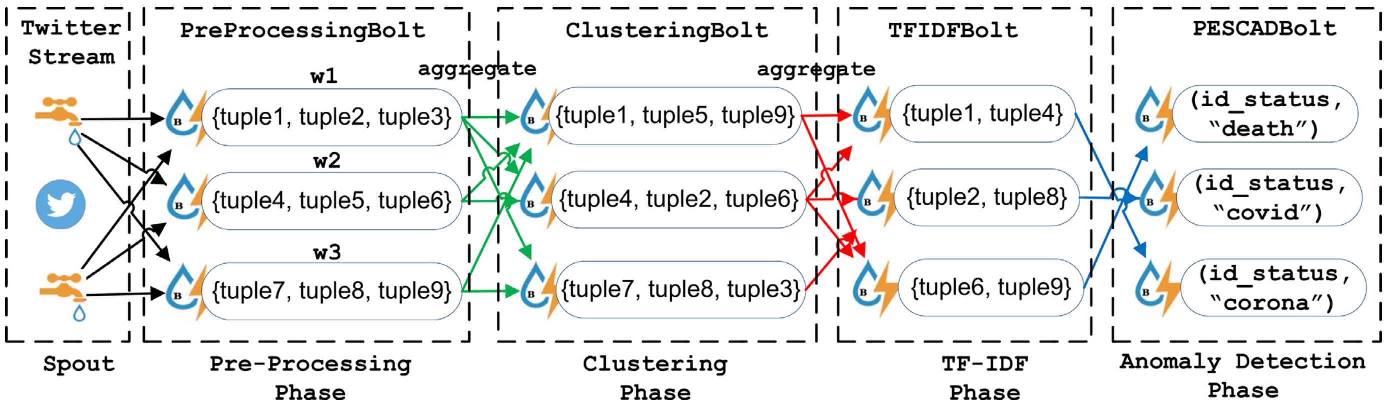


Fig. 3. Example representation of directed acyclic graph topology model of storm framework.

3.4. Grouping type

To pass the tuple (data) from the spout to the bolt or from one bolt to the next bolt, we need to define and implement the type of grouping. This type of grouping is also one fundamental concept in Big Data processing for large-scale, distributed real-time data analytics [15].

Global grouping is a grouping type where all the tuples go to one of the bolt workers. This grouping type is proper when we have to run a computational process with a tuple value. The downside to this grouping type is the network and memory overhead.

Shuffle Grouping is a grouping type where each worker in a bolt is guaranteed to receive the same amount of tuples. The advantage of shuffle grouping is to provide load balancing and avoid overhead as the worker is allocated to the process, and the tuples are partitioned in parallel.

Field Grouping is a grouping type where the tuples are partitioned by the "id" field defined by the programmer/user. We implemented this grouping type in our research to combine collective tuples with identical values into a given worker in a bolt. The example illustration of the grouping type is shown in Fig. 4.

3.5. Event stream

The Twitter stream we collect includes an unbounded information sequence (event) known as a tuple, which is a multi-field, key-value pair data structure [6]. For our three separate primary bolts,

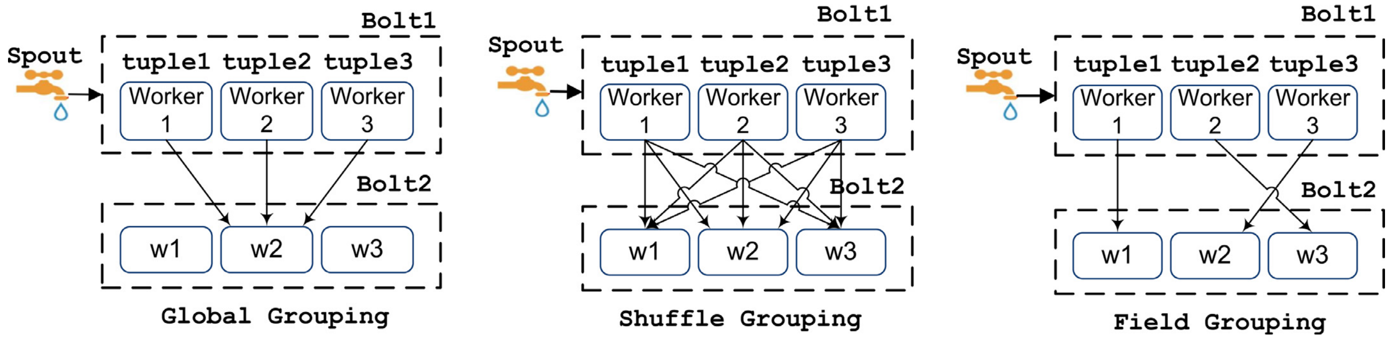


Fig. 4. Grouping type of tuple.

there are three types of event stream equations:

$$S = \{(id\_status_1, is\_event_1), (id\_status_2, is\_event_2), \ldots (id\_status_n, is\_event_n)\} \tag{1}$$

$$S = \{(topic_1, word_1), (topic_2, word_2), \ldots (topic_n, word_n)\} \tag{2}$$

$$S = \{(id\_status_1, word_1), id\_status_2, word_2), \ldots (id\_status_n, word_n)\} \tag{3}$$

Eq. 1 illustrates the grouping formula in our PESCADBolt (i.e. Poisson Event Stream Collective Anomaly Detection Bolt) as the system performs field-grouping for both *id_status* and *is_event* for each event. We assign $is\_event = 1$ if a tweet contains a “death” keyword. Then we sum *is_event* and compare it to the predicted rate, which is calculated using the Poisson probability. If the sum of *is_event* is greater than the predicted rate, we will mark all *id_status* as collective anomalies. As shown in Eq. (2), we also group *word* and *topic* in TFIDFBolt. We calculate the TF-IDF frequency score from each keyword relative to all the tweets collected in our system.

We will then apply the field grouping for *id_status* and *word* in the ClusteringBolt, as given in Eq. (3), which transforms each keyword into a vector with the word embedding technique and groups it into a graphical 2-D representation in its associated clusters.

3.6. System environment

For aggregating the incoming rapid real-time tweets, we require a framework for Streaming Processing Computation, Apache Storm (at least version 1.2.x) [16]. We also require Java JDK 11 installed and Ubuntu 18.04 LTS (for the latest package updates and the convenience of configuration).

We need the Twitter4j library (at least version 4.0) configured with Twitter API keys to access several Twitter entities. To compile the project and use the dependencies needed, we require ‘maven’ (at least version 3.x). We launch our experiments by creating and configuring two nodes in this research: a nimbus node and a supervisor node. In the nimbus node, we configured Apache Zookeeper (at least version 3.4.x) to coordinate the communication between the nimbus node and the supervisor node. Python version 3.x installed for executing part of Storm dependencies.

The nimbus node is also utilised as the visualisation server. All nodes set up on the virtual machines (Virtual Box at least version 6.0.x). The visualisation server uses LAMP, which consists of a database server (MySQL 10.4.11), a web server (Apache 2.4.43), and a programming language (PHP 7.2.31).

3.7. Data collection

We collect the data using the Twitter API, allowing us to retrieve the essential object information (such as accounts, hashtags, tweets), namely as ‘entities’. In our case, we use “status” and “user” entities. From status entity, we extract following fields: “created_at”, “geolocation”, “place”, and “status’. On the user entity, we obtain the “location” and “screen_name” fields. Since we will need to detect the origin/source of the anomaly tweet, both *geolocation* and *location* are essential information. However, in reality, not all user account disclose these two pieces of information. Therefore, we can only show the source/location of the anomalous tweet in the map (more in Section 4.4) only for tweets or account if their *geolocation* and *location* are not null. However, we still list all the anomalous tweets (more in Section 4.3).

3.8. Pre-processing phase

After acquiring the tuple from Spout (“start” mark in Fig. 1), we retrieve the hashtag and user-mention entities, tokenise the tweet sentence, and delete the stopwords on the *PreprocessingBolt* (orange rectangles). Then we do the lemmatisation to get a standardised keyword that suits the dictionary in the real world.

3.9. TF-IDF phase

We measure the number of keywords in *TFIDFBolt* (green/middle-right rectangles in Fig. 1) and determine its frequency (i.e. Term Frequency-Inverse Document Frequency / TF-IDF). The TF-IDF is a statistical formula that indicates the importance of a word in a document relative to a collection of documents or corpus. This research uses this formula to extract the most important keyword from a tweet sentence that does not contain hashtags and user-mentions. We compute the TF-IDF scores and return the keyword with the highest score. Subsequently, we define the keyword with the highest score as the “important” keyword of the tweet. To measure the Term Frequency (TF), we determine how many times a keyword occurs in a tweet divided by the total number of word counts in that tweet sentence (to get the normalised value). Let $|\{w \in T\}|$ be the number of times the keyword shown in a tweet, and let $|T|$ be the total number of all words in a tweet:

$$tf(w, T) = \frac{|\{w \in T\}|}{|T|} \tag{4}$$

For the Inverse Document Frequency (IDF), we determine the natural logarithm of the total number of tweets obtained in our tweet database divided by the number of tweets where a specific keyword occurs. Let *CT* be the collection of the tweet obtained in our database, then, the IDF formula:

$$idf(w, CT) = \log \frac{|CT|}{|\{T \in CT : w \in T\}|} \tag{5}$$

Therefore, the whole TF-IDF formula:

$$tf\_idf(w, T, CT) = tf(w, T) * idf(w, CT) \tag{6}$$

Following is an example of TF-IDF calculation. Given a tweet sentence T containing 200 words where the word w "covid" appears 5 times in that sentence. The term frequency (i.e., tf) for "covid" is then $(5/200) = 0.025$. Subsequently, assume we have 1000 tweets in document collection *CT* and the word "covid" appears in 10 tweet sentences T among these 1000 tweets in document collection *CT*. Therefore, the inverse document frequency (i.e., idf) is calculated as $\log(1,000/10) = 2$. Thus, the Tf-idf weight of the word "covid" is the multiplication of these two: $0.025 \times 2 = 0.5$.

3.10. Clustering phase

In the *ClusteringBolt* (red/lower-right rectangles in Fig. 1), we use the word embedding to represent each keyword as a vector of 100 dimensions. We group the keyword vectors using clustering (k-means clustering) and then project the 100-dimensional vectors into 2-dimensional vectors via the Principal Component Analysis (PCA) to plot the keyword clusters in a 2D visual image.

3.10.1. Word embedding (Word2Vec)

Word2vec is a neural network model with a hidden layer that transforms a word into a real number vector (i.e., *Word Embedding*). This vector represents the coordinate in a high dimensional vector space, such that keyword with a high similarity can be located next to each other [35].

We decide to choose this strategy to represent a keyword as a vector and map it in a visual representation. Our word embedding approach will use the Skip-gram algorithm [36], which employs a set of keywords extracted from the tweets (i.e., as a corpus), then the model loops on the words and applies the current keyword to infer or predict its neighbours (i.e. context). In this research, we implement Word2Vec with the help of deeplearning4j library [29].

With built-in functions, the string collection (*CollectionSentenceIterator*) will be tokenised (*DefaultTokenizerFactory*), and the model will iterate through tokens, and delete the stopwords. After the pre-processing step, the library collects the tokenised keywords, and it selects unique words only, one by one, until they form a vocabulary that consists of 2000 unique words. Each token will be supplied to Word2Vec neural network (using *Word2Vec.Builder()*).

In this project, the size of our vocabulary is 2000 words. Also, we use *windowSize* parameter of 5. Meanwhile, we set the hidden layer size of our neural network (i.e., *layerSize*) with 100, i.e., every word in the vocabulary will be represented by a 100-dimensional vector.

3.10.2. KMeans clustering

Once transformed into a 100-dimensional vector, we use the deeplearning4j library [29] to perform KMeans clustering.

KMeans Clustering is a type of unsupervised learning which clusters finite n instances of the dataset with d dimensional real vectors into k clusters by minimising the distances between data instances and several cluster centres/centroids. The data instances, in this case, are the keywords, and each keyword consists of a vector with real numbers. The distance metric we use in this KMeans Clustering is the Euclidean distance.

3.10.3. Principal component analysis

In this research, we use Principal Component Analysis (PCA) from the deeplearning4j library [29]. PCA is a technique for dimensionality reduction to project and keep the essential information from a higher dimension to a smaller vector subspace. The technique maximises the projected data variance. The word embeddings will be projected to a two-dimensional space using PCA, which allows us to visualise the word clusters.

3.11. Anomaly detection phase

Finally, in *PESCADBolt* (blue/upper-right rectangles in Fig. 1), we used the Poisson Event Stream Collective Anomaly Detection (PESCAD) algorithm. The rate of events (i.e., the keyword "death") is computed as the actual events, and we forecast the predicted number of events, which we want to approximate by using Poisson distribution. If the number of actual events in the interval is greater than the predicted number of events, we shall mark them as a collective anomaly.

The Apache Storm adopts the scalable Directed Acyclic Graph topology design [37], and our PESCADBolt can scale according to our topology definition in our code along with the number of nodes configuration in our cluster (either standalone or distributed). Intuitively, with more nodes in the cluster, it can detect multiple anomalies at the same time.

3.11.1. Collective anomaly detection

In the previous related works, there are three forms of anomaly detection: Point, Collective, and Contextual [38]. *Point anomaly* indicates a single data (point) anomaly compared with the rest of the data, e.g., tracking a user's network intrusion detection. Meanwhile, *Contextual anomaly* is associated with abnormal occurrence in particular/specific circumstance (context), such as the network's obscure intrusion detection late at night. Lastly, the *Collective anomaly* detects the group of abnormal occurrences over a period of time. Therefore, we decide to use collective anomaly detection because we attempt to detect a collection of "death" keyword/event in a specific time interval.

3.11.2. Poisson distribution

Given the average rate event (λ), the Euler's constant number ($e = 2,71828$), we use the Poisson $P(x, \lambda)$ as the statistical method to calculate the probability of x occurrences of the event over a specific period:

$$P(x, \lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{7}$$

We get the λ by dividing the total number of occurrences of events by the total of all tweets:

$$\lambda = \frac{sum\_total\_event}{sum\_total\_tweet} \tag{8}$$

We specify the time interval in one minute because this death event could occur at a monitoring interval of at least one minute from the six weeks of our observation.

3.11.3. Poisson event stream collective anomaly detection (PESCAD) algorithm

This algorithm, PESCAD, is our highlight since event detection is the motivation of our research (Algorithm 1). The main principle is to use the Poisson distribution to measure occurrences using the current interval and compare it to actual occurrences to decide whether an anomaly occurs.

Firstly, we extract from the tuple these fields: *id_status* and *is_event*, and calculate *sumTotalTweet*. From *is_event* counts, we count the *sumTotalEvent (also as "ActualOccurrences")* and *sum current interval event (SCIE)* independently. We will then use two parameters (λ and *sum current interval event(SCIE)*) to calculate the Poisson probability. After that, by multiplying the Poisson probability and the *sumTotalTweet*, we can calculate the *PredictedOccurrences (PO)*. Later, the *ActualOccurrences (AO)* is compared to the

PredictedOccurrences (PO). If the *ActualOccurrences (AO)* is greater than the *PredictedOccurrences (PO)*, we record all of *id_status* as group/collective anomaly in the current interval.

We explain the details of the algorithm as follows. The input of the algorithm is the array of tweet streams with an arbitrary length of *D* which previously converted as the data tuples from the Pre-processing phase. We then initialise *sum current interval event(SCIE), sumTotalEvent*, and *sumTotalTweet* with 0. The *SCIE* counts how many tweets which contain the "death" keyword in the current interval of one minute. The *sumTotalEvent* counts how many tweets which contain the "death" keyword in whole length time monitoring (e.g., 1 h). The *sumTotalTweet* counts the total tweet collected in whole length time monitoring.

For all incoming tweet data tuples, we increment the (*sumTotalTweet*). If a tweet contains a "death" keyword ($is\_event == 1$), we record *id_status* of that tweet into {*idStatusList*} array, and increment the *SCIE*. In every one minute, we add *SCIE* into *sumTotalEvent*. We also calculate λ by dividing *sumTotalEvent* by *sumTotalTweet* which represent the average of event occured in the whole monitoring time (Section 3.11.2). With the λ obtained and *SCIE* we calculate the Poisson probability and we obtain *PredictedOccurrences (PO)*. Meanwhile, we also assign the *sumTotalEvent* count into *ActualOccurrences (AO)*. If *ActualOccurrences (AO)* is larger than *PredictedOccurrences (PO)*, we mark all *id_status* in the {*idStatusList*} array as a group anomaly and print all the anomaly on the system. Subsequently, we empty the {*idStatusList*} array and reset the *SCIE* into 0. The above steps keep iterating until every time we reach 1 min.

4. Testing

After designing our methodology and implemented our algorithm, we then test our system. Since we undertake research related to the outbreak or anomaly detection monitoring, there is a hypothesis and primary aim that we require to test: our system should identify abnormal events/abnormal rate occurring in a specific interval, which leads to an incident, and the system should be able to detect the incident's source at the same time.

This Section 4 only discusses and analyses a subset of our findings from our monitoring on *14 August 2020*. We will discuss the complete result discussion from research and monitoring during *1–30 August 2020* in Section 5. We have also designed a web app[1] with interactive visualisation charts and map chart for the convenience of analysing the anomalies using chart.js[2] and leaflet.js.[3]

4.1. Sensitivity analysis of parameters

We analysed the two essential aspects: *apache storm parameters* and *topology components* of our framework. We subsequently observed whether these two aspects would affect how many tweets we will obtain during a specific monitoring time.

Table 1 illustrates the tuning of apache storm parameters (i.e., *number of workers, number of ackers, maximum task parallelism, maximum spout pending*) [41] and how many the tweets obtained from the respective tuning value parameter.

The *number of workers* denotes how many workers instances in Java Virtual Machine that storm creates for the topology. The *number of ackers* is the number of threads for processing tuple acknowledgements. The *maximum task parallelism* defines maximum number of threads that generates spout and bolts. *Maximum spout pending* specifies how many data tuples have been processed from the spout and ready to be processed by ackers.

[1] The homepage of the Github project: PESCAD Storm

[2] The homepage of the library: https://www.chartjs.org/ [39].

[3] The homepage of the library: https://leafletjs.com/ [40].

We perform five monitoring time durations (i.e., 1, 5, 10, 20, or 60 min) with the above parameters. We use a similar tuning value as demonstrated in Bilal [42].

For example, when we set the *number of workers* = 3 and we start to monitor for 1 min long, we obtained 3254 tweets. The same applies when the number of workers equals 6, 9, and 12 (also in 1 min); we obtained pretty similar amounts: 3045, 3629, and 3390 tweets, respectively.

When we experimented on the other three parameters, we kept obtaining about 3000 tweets in 1 min despite the value we assigned.

However, we significantly collected more tweets than the previous duration when we increased the monitoring duration (i.e., from 5 until 60 min). For instance, when we set the maximum spout pending as 8000, we collected 2988 tweets in one minute. On the other hand, when we increased the duration for 5, 10, 20, and 60 min, we collected 10,230, 22,898, 48,090, and 143,489 tweets, respectively.

Meanwhile, Table 2 shows how the topology components will affect the number of tweets collected. We decided to use 60 min of monitoring time as our baseline, based on the previous Table 1 experiment.

In this experiment, we designed a simple topology with 3 components: *spout, bolt1*, and *bolt2*. A number of *spouts* will collect many tweets, and a number of *bolt1* will receive the tweets from the spout and then forward to *bolt2*. Therefore, we tuned only for *spout* and *bolt1*. We set *bolt2* = 1 because this bolt task is to accumulate the number of tweets collected. If we set *bolt2* into 2 or 3, they would work independently so that we would not be able to record the total number of tweets, and we would have to add them manually.

As shown in Table 2, when we set the *spout = 3, bolt1* = 2, and *bolt2* = 1, we collected 141,545 tweets. It also applies when we increased both spout and bolt, we obtained a fluctuated amount of tweets ranging from 142,899 to 148,695.

We can conclude that we kept obtaining about 140,000 tweets during 60 min of monitoring despite how many topology components we added. However, we will show that if we add more computer nodes (in distributed mode), we will obtain more tweets in Section 5.

4.2. Anomaly detection calculation

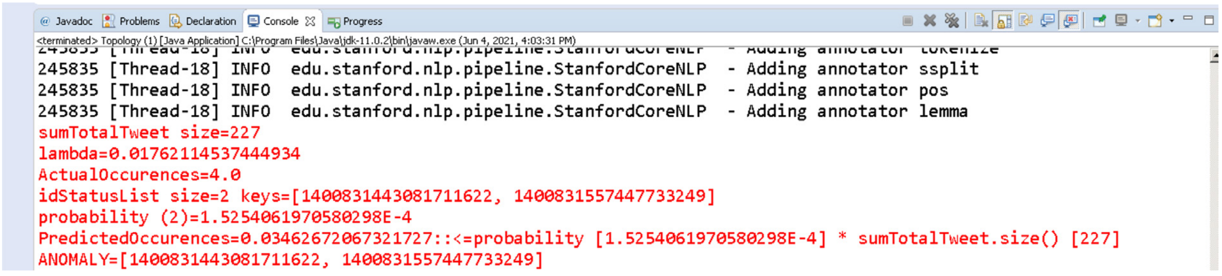
If we execute the system in standalone mode particularly with a suitable Integrated Development Environment (IDE) that displays output consoles we can see our PESCAD Algorithm in detail. Fig. 5 is the excerpt of the example message console during our standalone test run. As we can see, we observe following variables:

1. How many total tweets we capture during all monitoring period (*sumTotalTweet*).
2. The average of events during all monitoring period (*lambda*).
3. The sum of all events detected during all monitoring period (*ActualOccurrences*).
4. The list of status id(s) of the events during the current monitored interval (*idStatusList*).
5. The Poisson probability calculation for the current interval (*probability*).
6. The predicted rate occurrences of the event obtained from the Poisson calculation (*PredictedOccurrences*).
7. The list of all marked anomalies during the current interval (*ANOMALY*).

From Fig. 5, as an example, at that arbitrary moment, we have gathered 227 tweets during the monitoring, and in that arbitrary period, we obtained $lambda = \frac{ActualOccurrences}{sumTotalTweet} = \frac{4.0}{227} \approx 0.0176211$.

Table 1
Storm parameter tuning on tweets obtained.

Monitoring time	Apache storm parameters							
	num of worker	tweets	*num of acker*	tweets	*max task parallelism*	tweets	*max spout pending*	tweets
1 min	3	3254	3	3493	3	3104	2000	3120
	6	3045	6	3623	6	3229	4000	3075
	9	3629	9	3139	9	3043	6000	2959
	12	3390	12	3127	12	3316	8000	2988
5 min	3	10,233	3	10,323	3	11,203	2000	11,232
	6	11,200	6	10,432	6	10,943	4000	10,992
	9	10,523	9	11,293	9	11,123	6000	11,029
	12	10,983	12	10,849	12	10,320	8000	10,230
10 min	3	24,024	3	23,765	3	21,343	2000	24,044
	6	23,049	6	23,658	6	23,984	4000	24,578
	9	22,043	9	22,674	9	23,995	6000	23,989
	12	24,045	12	22,874	12	22,900	8000	22,898
20 min	3	47,945	3	48,783	3	46,458	2000	45,884
	6	48,640	6	46,939	6	47,999	4000	46,989
	9	46,399	9	48,999	9	48,799	6000	47,989
	12	47,989	12	46,989	12	45,889	8000	48,090
60 min	3	148,200	3	146,379	3	148,900	2000	145,778
	6	139,000	6	147,890	6	148,970	4000	143,899
	9	145,980	9	140,887	9	143,478	6000	143,689
	12	146,839	12	144,783	12	144,788	8000	143,489



```
<terminated> Topology (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (Jun 4, 2021, 4:03:31 PM)
245835 [Thread-18] INFO  edu.stanford.nlp.pipeline.StanfordCoreNLP  - Adding annotator tokenize
245835 [Thread-18] INFO  edu.stanford.nlp.pipeline.StanfordCoreNLP  - Adding annotator ssplit
245835 [Thread-18] INFO  edu.stanford.nlp.pipeline.StanfordCoreNLP  - Adding annotator pos
245835 [Thread-18] INFO  edu.stanford.nlp.pipeline.StanfordCoreNLP  - Adding annotator lemma
sumTotalTweet size=227
lambda=0.01762114537444934
ActualOccurences=4.0
idStatusList size=2 keys=[1400831443081711622, 1400831557447733249]
probability (2)=1.5254061970580298E-4
PredictedOccurences=0.03462672067321727::<=probability [1.5254061970580298E-4] * sumTotalTweet.size() [227]
ANOMALY=[1400831443081711622, 1400831557447733249]
```

Fig. 5. The excerpt of the system's console.

Table 2
Topology components tuning on tweets obtained.

Monitoring time	Storm topology components			Tweets
	num of spout	*num of bolt1*	*num of bolt2*	
60 min	3	2	1	141,545
	6	4	1	148,695
	10	4	1	146,898
	12	8	1	144,909
	24	16	1	142,899
	48	32	1	146,288

Since we have collected two events (from a total actual four occurrences) in the current interval (i.e., as shown in the *idStatusList* size), we can calculate $Poisson(x, \lambda) = Poisson(2, 0.0176211) \approx 0.0001525$. We then calculate the event occurrences *PredictedOccurrences* = *Poisson (2, 0.0176211)* * *sumTotalTweet* $\approx$ 0.034626. Since *ActualOccurrences* (i.e., 4.0) is greater than *PredictedOccurrences* (i.e., 0.034626), we mark all of the *id_status* in *idStatusList* of that current interval as the group/collective anomaly and being inserted to *ANOMALY* array.

4.3. Anomaly detection chart

Fig. 6 depicts our findings during 14 August 2020 monitoring. In the figure, the "event" is the tweet that contains the "death" keyword, which could potentially become anomalous. Meanwhile, the "anomaly" is the event tweet that becomes an anomaly according to our statistical computation and Algorithm 1. The figure also shows the varied amounts of events and anomalies corresponding to the grouping time interval per one-minute. We also see at which time the most number of anomalies occurred and which time there is no anomaly at all (i.e., least abnormal event captured). From the figure, we can understand that we obtained the highest cumulative anomaly (three events) on *14 August 2020 at 21:41*.

Apart from that, we also show the list of anomalous tweet on our system, as shown in Fig. 7. If we click one of the tweet lists, we may redirect automatically to the tweet page source (Fig. 9). In the following subsection, alternatively, we can also locate the anomalous tweet's source using the world map.

4.4. Anomaly detection map

Fig. 8 shows the location of anomalous tweets (i.e., only for the tweet which contains *geolocation* or *place* entities). These entities are needed to pinpoint the location and identifying the source of the anomalous tweets/incidents. We plot the anomalous tweets into a tilemap and implicitly construct the URL so that the user can view the original tweet on the Twitter website.

When we click on the located pin and the popup on the map, we can be redirected to the exact web page of the anomalous tweet, which for example, explains the casualties updates on 14 August 2020 reached 46,707 in the United Kingdom (Fig. 9).

It demonstrates that the Collective Anomaly was accurately detected by our system on *14 August 2020 at 21:41*, because three events of "death" in tweets associated with COVID-19 occurred, and our system can correctly determine and pinpoint the source of the incident (i.e., accomplished the second objective of our project). As opposed to this Section 4, which only discuss a spe-

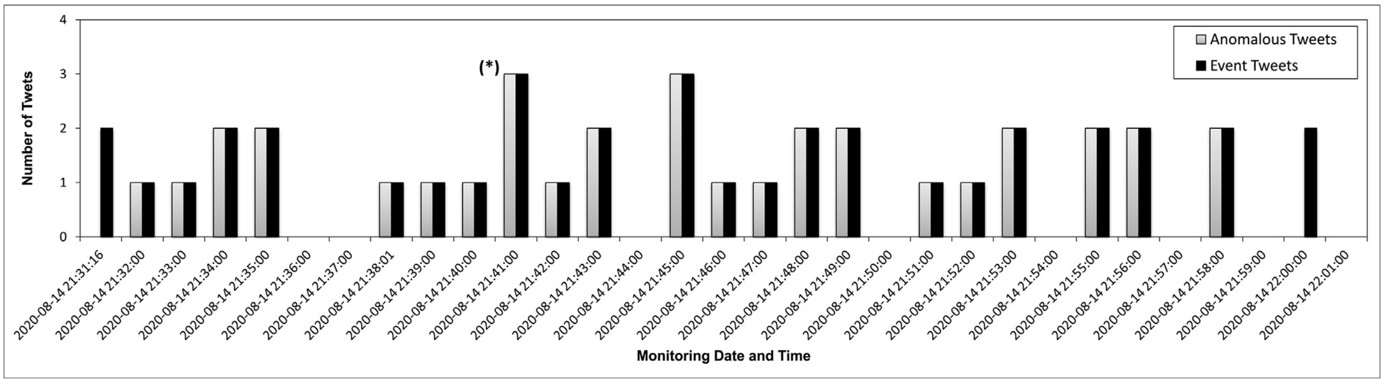
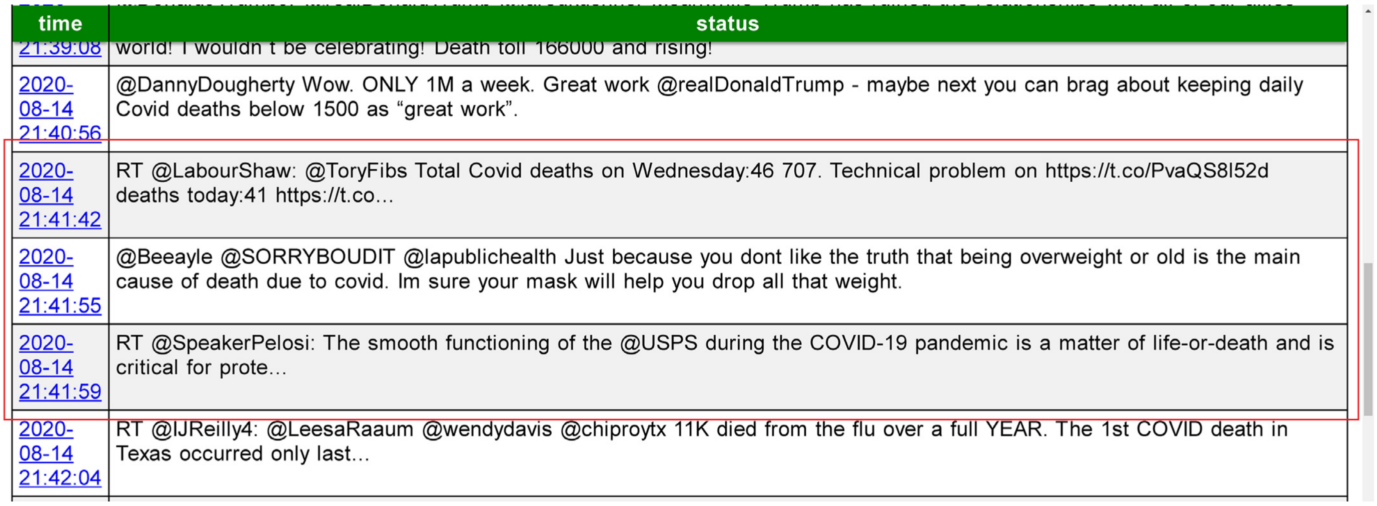
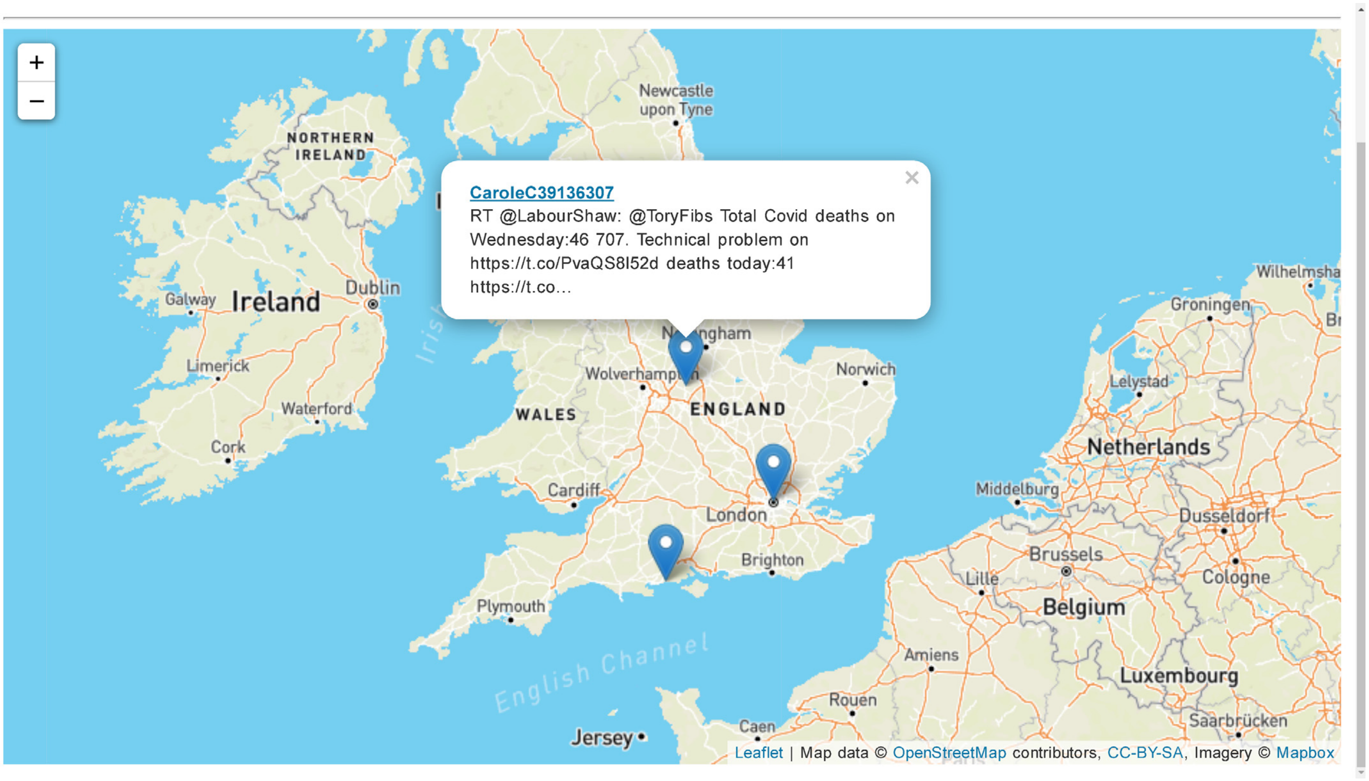


Fig. 6. The collective anomaly detection on 14 August 2020. The asterisk (*) denotes the highest collective anomaly detected (i.e., three anomalies) on that specific time.



time	status
21:39:08	world! I wouldn't be celebrating! Death toll 166000 and rising!
2020-08-14 21:40:56	@DannyDougherty Wow. ONLY 1M a week. Great work @realDonaldTrump - maybe next you can brag about keeping daily Covid deaths below 1500 as "great work".
2020-08-14 21:41:42	RT @LabourShaw: @ToryFibs Total Covid deaths on Wednesday:46 707. Technical problem on https://t.co/PvaQS8I52d deaths today:41 https://t.co...
2020-08-14 21:41:55	@Beeayle @SORRYBOUDIT @lapublichealth Just because you dont like the truth that being overweight or old is the main cause of death due to covid. Im sure your mask will help you drop all that weight.
2020-08-14 21:41:59	RT @SpeakerPelosi: The smooth functioning of the @USPS during the COVID-19 pandemic is a matter of life-or-death and is critical for prote...
2020-08-14 21:42:04	RT @IJReilly4: @LeesaRaaum @wendydavis @chiproytx 11K died from the flu over a full YEAR. The 1st COVID death in Texas occurred only last...

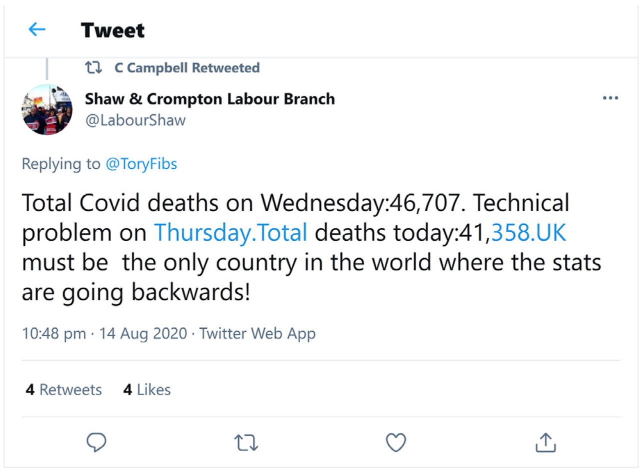
Fig. 7. The excerpt of three collective anomalies detected on 14 August 2020 at 21:41.



CaroleC39136307
RT @LabourShaw: @ToryFibs Total Covid deaths on Wednesday:46 707. Technical problem on https://t.co/PvaQS8I52d deaths today:41 https://t.co...

Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox

Fig. 8. The source location of the specific tweet which contains the anomalous event (i.e., "death" keyword).


Tweet

C Campbell Retweeted

Shaw & Crompton Labour Branch
@LabourShaw

Replying to @ToryFibs

Total Covid deaths on Wednesday:46,707. Technical problem on Thursday.Total deaths today:41,358.UK must be the only country in the world where the stats are going backwards!

10:48 pm · 14 Aug 2020 · Twitter Web App

4 Retweets 4 Likes

Fig. 9. The source of the tweet: status.

cific finding on 14 August 2020, Section 5 will discuss our findings in accumulated monitoring during 1–30 August 2020.

5. Results

We have conducted 15 tests between single versus dual machines (e.g. 15 tests × 2 machines = 30 days) between 1 and 30 August 2020 and compare them, as shown in Fig. 10.

From Fig. 10, we can see that the distributed machine peaked at 10th test (3372 tweets) as compared to the standalone one (1654 tweets). Although dropped at 11th test, the distributed framework still surpassed the standalone one (1330 tweets as compared to 971 tweets respectively), and it shows that the distributed framework outperforms the standalone machine in all tests (i.e., accomplished the first objective of the project).

Fig. 11 illustrates the number of anomalous tweets collected among the total of event tweet during 30 days. The highest anomalous tweet rate detected on 18 August 2020 (134 anomalies from the total of 136 events). When we analyse the tweet source (on that date), multiple tweets are associated with the COVID-19 casualties update. Also, most of the tweets contain debates about whether comorbidity causes death. This problem caused a *spike* in the event rate (i.e., many tweets contained the keyword “death”), and subsequently, our PESCAD algorithm can successfully detect the anomaly (i.e. accomplished the second objective of the project).

Fig. 12 shows the most mentioned words on Twitter during our monitoring, e.g., “covid” (21,566 incidents), “death” (11,799 incidents) and “trump” (4761 incidents). Meanwhile, Fig. 13 showed the words with highest TF-IDF scores, e.g., “people” (0.63637), “school” (0.5921407), and “virus” (0.57385). When we observed Twitter in August 2020, we find that the United States has had trouble with COVID-19, causing people ask to their president at that time (Trump) to be responsible for the increasing death events. Meanwhile, also in August 2020 (in the United Kingdom), the government started to loosen the restrictions, reopening school but causing the increased COVID-19 case, thus resulting in some concerns for the people.

Fig. 14 illustrates the clusters of 25 most mentioned keywords during our monitoring time. In the figure, we can find the keywords “school”, “pandemic”, and “president” grouped into a cluster. Meanwhile, the keywords “death”, “case”, and “corona” are in another separated cluster. We can conclude, the relationship of Figs. 12–14 as follows: Fig. 12 illustrates the keyword occurrence (term frequency), and in Fig. 13, we use that information (word

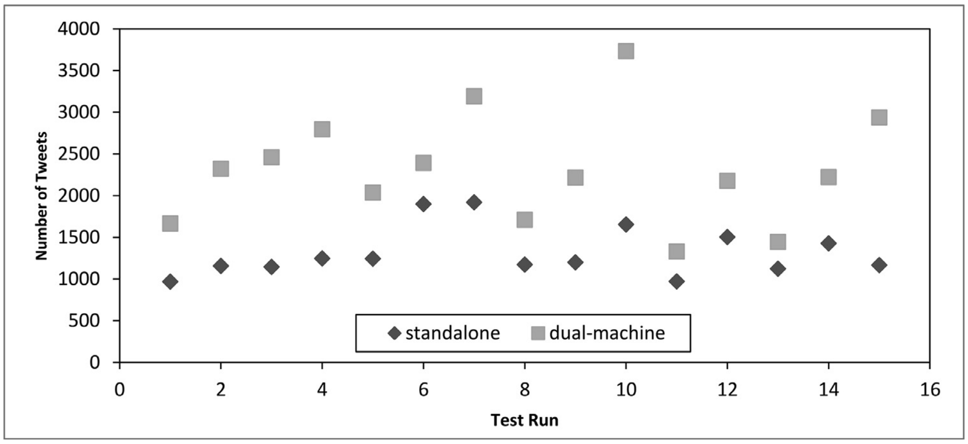


Fig. 10. Standalone vs. Dual-machine Tweet collection comparisons during 1–30 August 2020.

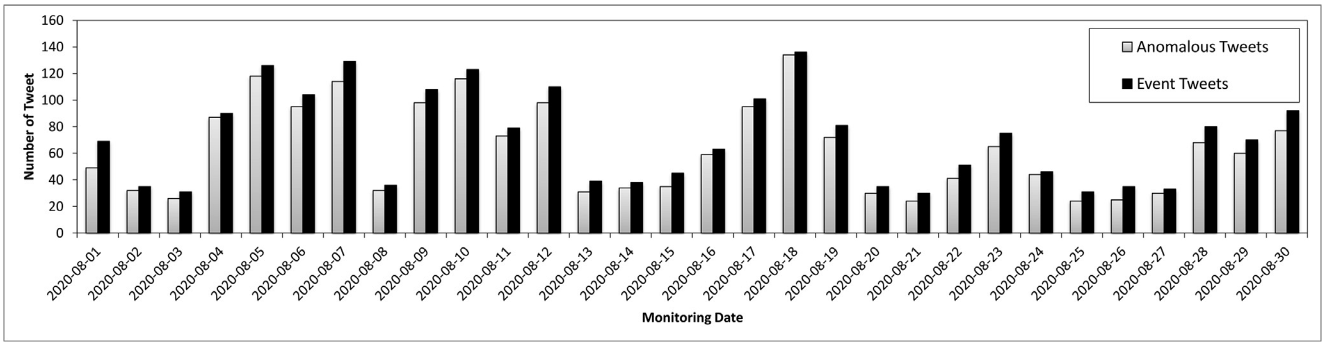


Fig. 11. Anomalous tweets vs. Event tweets collected during 1–30 August 2020.

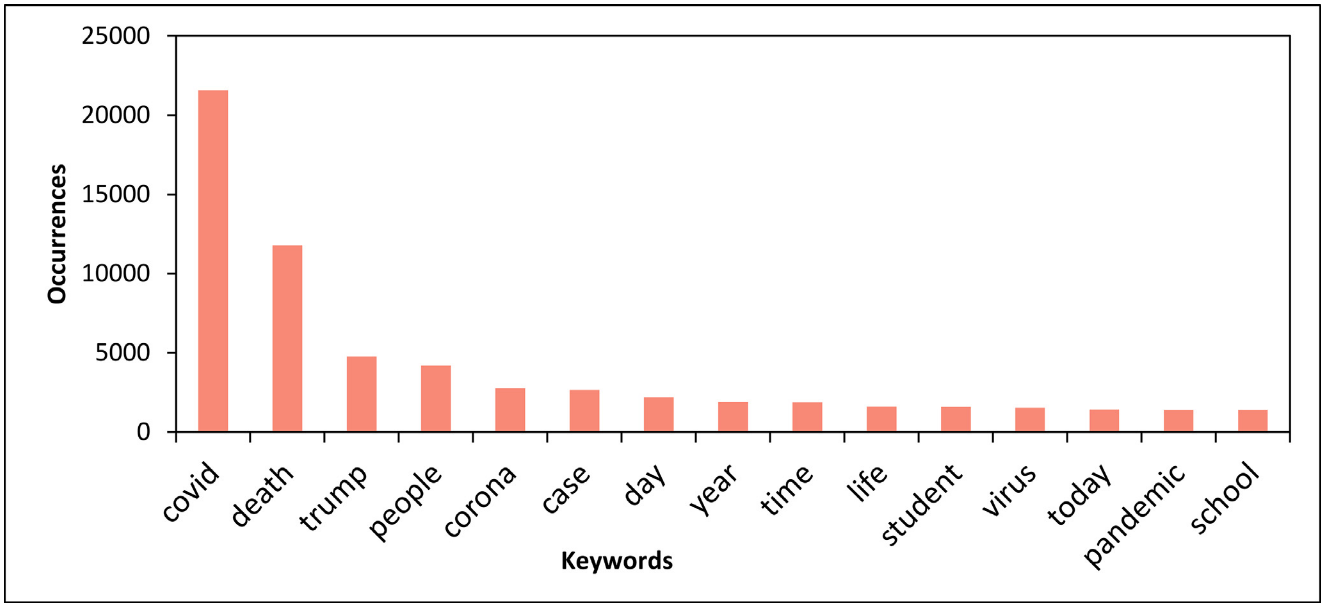


Fig. 12. Word count calculation during 1–30 August 2020.

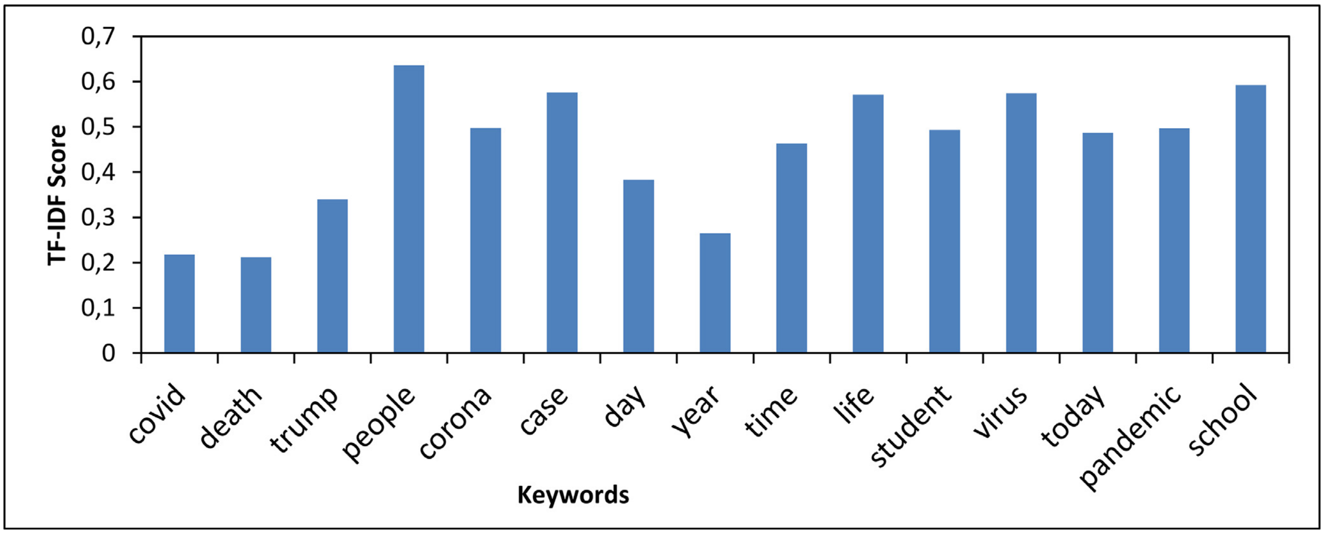


Fig. 13. TF-IDF score during 1–30 August 2020.

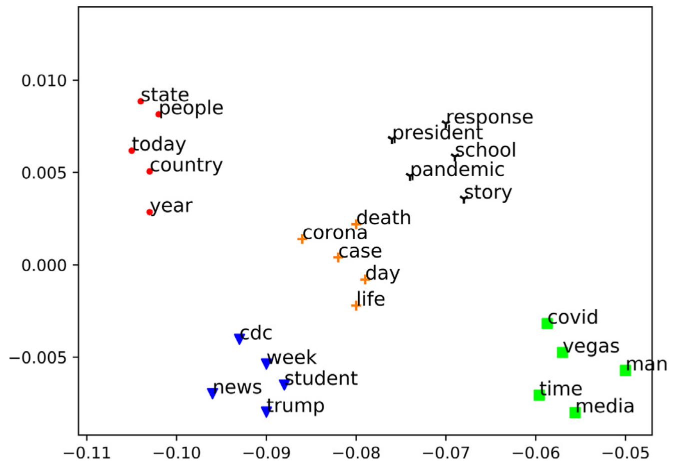


Fig. 14. Word Cluster chart during 1–30 August 2020.

occurrence/term frequency) combined with inverse document frequency, to calculate the TF-IDF, which represents the importance of that word among our collected tweets. Meanwhile, Fig. 14 illustrates how we visualise the keywords in a 2-Dimension graph as if the words are grouped and formed a certain topic grouping. In summary, these three figures illustrate what people concern during this pandemic period (i.e., accomplished the third objective of our project).

6. Conclusion

In conclusion, we have obtained more tweets from distributed computing results than a single machine from *1 to 30 August 2020*. On *18 August 2020*, we received the highest number of anomalous tweets, discussing the pandemic casualties' updates and COVID-19 debates. During our monitoring time, we obtain the three most-appeared words on Twitter: “covid”, “death”, and “Trump”, which illustrate the most frequently mentioned keywords. Meanwhile, the keywords “people”, “school”, and “virus” have the highest score of the TF-IDF and reflects the most important keywords on Twitter. The keywords “death”, “corona”, and “case” are grouped in a cluster, whereas “pandemic”, “school”, and “president” also grouped in another different cluster. Those results indicate what people concern during this pandemic period. We have proven that our distributed Directed Acyclic Graph model framework collected more tweets than the standalone machine. Our system also successfully detect anomalous events from Twitter with its source location and account in real-time.

The weakness of our work is that we require extra testing hours to give us more precise insight into our conclusions, where they may be different from our current results. We also developed our project with minimal resources. Hence, we hope larger organisations (e.g., public health organisations) can benefit from this work by adopting our concept in larger computing clusters. The strength of our work is that we have successfully designed the Directed Acyclic Graph model combined with the Poisson distribution to detect the anomaly (i.e., PESCAD algorithm) so that others can benefit from the idea of the algorithm for their research. Also, although we have built our research with small computer nodes, we have achieved our research’s objectives. For future work, we plan to use more computing resources to improve our performance. We also plan to implement our algorithm and apply our method in other event detection scenarios such as disaster monitoring, earthquakes, fires or storms.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The research was undertaken by Syahirul Faiz, sponsored by the Indonesia Endowment Fund for Education (LPDP) for his study at the University of Liverpool and supervised by Dr Bakhtiar Amen and Dr Thanh-Toan Do.

References

[1] I. Agbehadji, et al., Review of big data analytics, artificial intelligence and nature-inspired computing models towards accurate detection of COVID-19pandemic cases and contact tracing, Int. J. Environ. Res. Public Health 17 (2020) 5330, doi:10.3390/ijerph17155330.

[2] A. Haleem, et al., Significant applications of big data in COVID-19pandemic, Indian J. Orthop. 54 (2020) 1–3, doi:10.1007/s43465-020-00129-z.

[3] O. Bencharef, S. Gazzah, A Survey on how computer vision can response to urgent need to contribute in COVID-19 pandemics, 2020, p. 1. 10.1109/ISCV49265.2020.9204043.

[4] P. Gilabert, S. Segui, Gradient boosting and language model ensemble for tweet recommendation, in: Proceedings of the Recommender Systems Challenge 2020, in: RecSysChallenge ’20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 24–28. https://doi.org/10.1145/3415959.3415997

[5] U. Qazi, M. Imran, F. Ofli, Geocov19: a dataset of hundreds of millions of multilingual COVID-19 tweets with location information, SIGSPATIAL Spec. 12 (1) (2020) 6–15, doi:10.1145/3404820.3404823.

[6] B. Amen, Distributed contextual anomaly detection from big event streams, Doctoral thesis, University of Huddersfield (2018).

[7] B. Amen, A. Grigoris, A theoretical study of anomaly detection, in: Big Data Distributed Static and Stream Analytics, 2018, pp. 1177–1182, doi:10.1109/HPCC/SmartCity/DSS.2018.00198.

[8] B. Amen, A. Grigoris, Collective anomaly detection using big data distributed stream analytics, in: IEEE. Guangzhou, China: IEEE, 2019, 2018, pp. 188–195, doi:10.1109/SKG.2018.00035.

[9] A. Habeeb, et al., Real-time big data processing for anomaly detection: a survey, Int. J. Inf Manag. (2019), doi:10.1016/j.ijinfomgt.2018.08.006.

[10] Z. Ding, M. Fei, An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window, in: ICONS, 2013, pp. 12–17.

[11] M. Li, K. Zhang, J. Liu, H. Gong, Z. Zhang, Blockchain-based anomaly detection of electricity consumption in smart grids, Pattern Recognit. Lett. 138 (2020) 476–482, doi:10.1016/j.patrec.2020.07.020. https://www.sciencedirect.com/science/article/pii/S016786552030266X

[12] B. Amen, J. Lu, Sketch of big data real-time analytics model, in: The Fifth International Conference on Advances in Information Mining and Management, 21st–26th June 2015, Brussels, Belgium, 2015, pp. 48–53.

[13] K. Patel, O. Hoeber, H. Hamilton, Real-time sentiment-based anomaly detection in twitter data streams, in: Advances in Artificial Intelligence, Springer International Publishing, Cham, 2015, pp. 196–203.

[14] A. Rahnama, Distributed real-time sentiment analysis for big data social streams, in: 2014 International Conference on Control, Decision and Information Technologies (CoDIT), 2014, pp. 543–549, doi:10.1109/codit.2014.6996998.

[15] A. Toshniwal, et al., Storm@twitter, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD ’14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 147–156, doi:10.1145/2588555.2595641.

[16] R. McCreadie, et al., Scalable distributed event detection for twitter, in: IEEE, 2013, pp. 543–549, doi:10.1109/BigData.2013.6691620.

[17] S. Zhao, et al., Real-time network anomaly detection system using machine learning, in: 2015 11th International Conference on the Design of Reliable Communication Networks, DRCN 2015, 2015, pp. 267–270, doi:10.1109/DRCN.2015.7149025.

[18] T. Sakaki, et al., Earthquake shakes twitter users: real-time event detection by social sensors, in: Proceedings of the 19th International Conference on World Wide Web, WWW ’10, 2010, pp. 851–860, doi:10.1145/1772690.1772777.

[19] V. Gupta, R. Hewett, Harnessing the power of hashtags in tweet analytics, in: 2017 IEEE International Conference on Big Data (Big Data, 2017, pp. 2390–2395, doi:10.1109/BigData.2017.8258194.

[20] V. Gupta, R. Hewett, Unleashing the power of hashtags in tweet analytics with distributed framework on apache storm, in: 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 4554–4558, doi:10.1109/BigData.2018.8622302.

[21] D.A. Lind, W.G. Marchal, S.A. Wathen, Statistical Techniques in Business & Economics, McGraw-Hill Education, 2008.

[22] A. Sapegin, et al., Poisson-based anomaly detection for identifying malicious user behaviour, in: International Conference on Mobile, Secure and Programmable Networking, Springer, 2015, pp. 134–150.

[23] M. Turcotte, et al., Poisson factorization for peer-based anomaly detection, in: 2016 IEEE Conference on Intelligence and Security Informatics (ISI), IEEE, 2016, pp. 208–210.

[24] K. Doshi, Y. Yilmaz, Online anomaly detection in surveillance videos with asymptotic bound on false alarm rate, Pattern Recognit. 114 (2021) 107865, doi:10.1016/j.patcog.2021.107865. https://www.sciencedirect.com/science/article/pii/S0031320321000522

[25] C.D. Manning, P. Raghavan, H. Schtze, Introduction to Information Retrieval, Cambridge University Press, USA, 2008.

[26] Q. Liu, et al., Text features extraction based on TF-IDF associating semantic, in: 2018 IEEE 4th International Conference on Computer and Communications (ICCC), 2018, pp. 2338–2343, doi:10.1109/CompComm.2018.8780663.

[27] S. Qaiser, R. Ali, Text mining: use of TF-IDF to examine the relevance of words to documents, Int. J. Comput. Appl. 181 (2018), doi:10.5120/ijca2018917395.

[28] S. Amin, M.I. Uddin, S. Hassan, A. Khan, N. Nasser, A. Alharbi, H. Alyami, Recurrent neural networks with TF-IDF embedding technique for detection and classification in tweets of dengue disease, IEEE Access 8 (2020) 131522–131533, doi:10.1109/ACCESS.2020.3009058.

[29] M.Z. Alom, et al., A state-of-the-art survey on deep learning theory and architectures, Electronics 8 (2019) 292.

[30] B. Li, et al., Scaling word2vec on big corpus, Data Sci. Eng. 4 (2019), doi:10.1007/s41019-019-0096-6.

[31] Z. Doshi, S. Nadkarni, K. Ajmera, N. Shah, Tweeranalyzer: twitter trend detection and visualization, in: 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), 2017, pp. 1–6, doi:10.1109/ICCUBEA.2017.8463951.

[32] M. Antunes, D. Gomes, R.L. Aguiar, Knee/elbow estimation based on first derivative threshold, in: 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), 2018, pp. 237–240, doi:10.1109/BigDataService.2018.00042.

[33] S. Benjelloun, M.E. El Aissi, Y. Loukili, Y. Lakhrissi, S.E.B. Ali, H. Chougrad, A. El Boushaki, Big data processing: batch-based processing and stream-based processing, in: 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS), IEEE, 2020, pp. 1–6, doi:10.1109/ICDS50568.2020.9268684.

[34] G. Sharma, V. Tripathi, A. Srivastava, Recent trends in big data ingestion tools: a study, in: R. Kumar, N.H. Quang, V. Kumar Solanki, M. Cardona, P.K. Pattnaik (Eds.), Research in Intelligent and Computing in Engineering, Springer Singapore, Singapore, 2021, pp. 873–881.

[35] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: Y. Bengio, Y. LeCun (Eds.), 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, Workshop Track Proceedings, 2013, pp. 1–12.

[36] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: C.J.C. Burges, L. Bottou, Z. Ghahramani, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States, 2013, pp. 3111–3119. https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html

[37] N. Cassavia, E. Masciari, Sigma: a scalable high performance big data architecture, in: 2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2021, pp. 236–239, doi:10.1109/PDP52278.2021.00044.
[38] V. Chandola, et al., Anomaly detection: a survey, ACM Comput. Surv. 41 (3) (2009), doi:10.1145/1541880.1541882.
[39] Chart.js, Open source HTML5 charts for your website, 2020, (http://www.chartjs.org/). Accessed: 2020-07-03.
[40] V. Agafonkin, Leaflet - a javascript library for interactive maps, 2020, (https://leafletjs.com/). Accessed: 2020-07-11.
[41] H. Herodotou, Y. Chen, J. Lu, A survey on automatic parameter tuning for big data processing systems, ACM Comput. Surv. 53 (2) (2020), doi:10.1145/3381027.
[42] M. Bilal M.and Canini, Towards automatic parameter tuning of stream processing systems, in: Proceedings of the 2017 Symposium on Cloud Computing, SoCC '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 189–200, doi:10.1145/3127479.3127492.

Bakhtiar Amen received his Ph.D. at the University of Huddersfield in 2018. He worked at Aston University (UK) as a post-doc research fellow. He is member of British Computer Society (BCS), ACM, and IEEE. Now he is working as a lecturer in the Department of Computer Science at the University of Liverpool.

Syahirul Faiz received the BEng in informatics engineering from Telkom University, Indonesia, in 2012. He received the M.Sc. in big data and high-performance computing from the University of Liverpool, UK, in 2020, with distinction, under Dr. Bakhtiar Amen's and Dr. Thanh-Toan Do's supervision. His interests include machine learning and big data analytics.

Thanh-Toan Do is a Senior Lecturer at the Faculty of Information Technology, Monash University. He obtained his Ph.D. in computer science at the French National Institute for Research in Computer Science and Control (INRIA) in 2012. From 2013 to 2016, he was a Research Fellow at the Singapore University of Technology and Design. From 2016 to 2018, he was a Research Fellow at the Australian Centre for Robotic Vision and the University of Adelaide. From 2018 to 2020, he was a Lecturer at the University of Liverpool. His research interests include computer vision and machine learning.