Information Sciences 625 (2023) 553–577

Contents lists available at ScienceDirect

Information Sciences

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

WhatsUp: An event resolution approach for co-occurring events in social media

Hansi Hettiarachchi *, Mariam Adedoyin-Olowe, Jagdev Bhogal, Mohamed Medhat Gaber

School of Computing and Digital Technology, Birmingham City University, Birmingham, UK

ARTICLE INFO

Article history:
Received 10 November 2021
Received in revised form 21 November 2022
Accepted 1 January 2023
Available online 7 January 2023

Keywords:
Word embedding
Dendrograms
Clustering
Social media

ABSTRACT

The rapid growth of social media networks has resulted in the generation of a vast data amount, making it impractical to conduct manual analyses to extract newsworthy events. Thus, automated event detection mechanisms are invaluable to the community. However, a clear majority of the available approaches rely only on data statistics without considering linguistics. A few approaches involved linguistics, only to extract textual event details without the corresponding temporal details. Since linguistics define words' structure and meaning, a severe information loss can happen without considering them. Targeting this limitation, we propose a novel method named *WhatsUp* to detect temporal and fine-grained textual event details, using linguistics captured by self-learned word embeddings and their hierarchical relationships and statistics captured by frequency-based measures. We evaluate our approach on recent social media data from two diverse domains and compare the performance with several state-of-the-art methods. Evaluations cover temporal and textual event aspects, and results show that WhatsUp notably outperforms state-of-the-art methods. We also analyse the efficiency, revealing that WhatsUp is sufficiently fast for (near) real-time detection. Further, the usage of unsupervised learning techniques, including self-learned embedding, makes our approach expandable to any language, platform and domain and provides capabilities to understand data-specific linguistics.

1. Introduction

Social media services such as Twitter and Facebook generate increasing volumes of data with their growing popularity [1]. These data contain different information ranging from personal updates and general discussions to breaking news. Also, social media platforms support fast information dispersal using their large user bases spread around the world [2,3]. There is also evidence of situations when social media broadcast news faster than traditional news media [4]. The other advantage of social media data is that they highlight the public opinion [5]. Considering these facts, there is a high tendency at present to refer to social media data to acquire newsworthy contents [6]. However, due to the vast volume and high dynamicity of data, it is impractical to analyse them manually to extract important or newsworthy content [7,8]. Therefore, the requirement of automated intelligent processes for event resolution from social media data is crucial for effective utilisation of data and information extraction [9]. Such an automated system will be beneficial for multiple parties, including the general public

* Corresponding author.
E-mail address: Hansi.Hettiarachchi@mail.bcu.ac.uk (H. Hettiarachchi).

https://doi.org/10.1016/j.ins.2023.01.001
0020-0255/

to receive interesting event updates, companies to be aware of their product demands or faults and rescue teams to get quick updates on crises.

Considering the importance of automated event resolution in social media, various approaches have been proposed by previous research involving different techniques, as further discussed in Section 2. These approaches can mainly be divided into three types based on the targeted event output: text, text and time, and time. The text-targeted systems are designed to identify all topics/events in a given corpus similar to topic modelling [10,11]. The text and time-targeted systems capture both event text and time by processing a data stream either offline (as a whole) [12] or online (as documents arrive) [13]. Unlike them, the time-targeted systems are designed as notification systems to notify the event occurred times [14,8]. Among these different types, we found that the online text and time-targeted systems are more useful in event detection, considering their informative output and user requirements. Analysing the used approaches to develop such systems, we noticed that apart from a few notable exceptions, most of the approaches rely only on data statistics without considering linguistics, specifically semantics. Since semantics describe the connections between words and their meanings, a serious information loss can happen without considering them. Further analysing the few methods which involved semantics, we recognised that they also use semantics to extract only the event text. Commonly, these methods used rule-based and prediction-based approaches to capture semantics. Among them, rule-based approaches (e.g. semantic class extraction [15], named entity extraction [16]) mainly targeted filtering important terms without focusing on their relationships. Also, they are less expandable due to language dependencies. The prediction-based approaches mostly used word embeddings considering their ability in capturing linguistic relationships between words [17,18]. However, to the best of our knowledge, all the available word embedding-based approaches use pre-trained models incapable of capturing linguistics specific to the underlying corpus, such as modified or misspelt words. Furthermore, the usage of pre-trained models limits the expandability depending on the model availability.

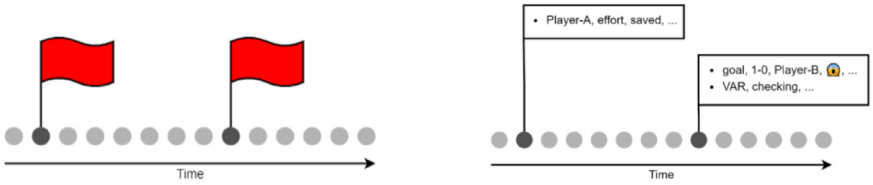
Focusing on the limitations in available approaches, in this research, we propose a novel method named *WhatsUp*, which involves statistics and linguistics to extract both event text and time. Rather than go for fully online processing, we select an intermediate level. We use the concept of time windows, which processes a set of documents that arrive during a period at once. Since different domains have different evolution rates, time windows allow the user to customise the system depending on the targeted domain. To identify event time (windows), we utilised the idea proposed by Embed2Detect [8], considering its promising results. However, as illustrated in Fig. 1, there is a clear distinction between Embed2Detect and WhatsUp. Briefly, Embed2Detect targets notifying users about events by recognising event windows, but WhatsUp targets extracting event windows along with their co-occurred event details. Embed2Detect utilises temporal variations of self-learned word embeddings and hierarchical clusters to detect event windows. Following this idea, we propose different strategies which improve the performance of event window identification along with WhatsUp, including an advanced text similarity measure named *Local Dendrogram Level Similarity* and a similarity change calculation technique named *Positive Similarity Change*. To extract fine-grained event text, we propose a novel clustering algorithm that identifies the co-occurred events in each event window. For clustering, we involve linguistical features captured using self-learned word embeddings and their hierarchical relationships in dendrograms and statistical features captured using token frequency-based measures. In summary, WhatsUp considers all the important features in textual data: statistics and linguistics (syntax and semantics) to detect event time and text of co-occurred events effectively.[1] Also, the usage of self-learned word embeddings allows capturing of the linguistics specific to the underlying corpus. Further, we preserve the universality of our approach by only using unsupervised learning techniques independent of language, platform and domain.

We evaluated the proposed approach in two diverse domains, sports and politics using *Twitter Event Data 2019* [8]. Considering the novelty of the event resolution proposed by this research, we designed a comprehensive set of metrics that automatically evaluate temporal and textual event details based on widely used evaluation metrics to measure performance. According to the comparisons with state-of-the-art methods, our method outperformed all of them with promising event detection. Further, we analysed the efficiency, and according to the results, our approach is sufficiently fast for (near) real-time detection.

Our contributions can be summarised as follows:

1. We propose a novel method named *WhatsUp* which detects both temporal (event occurred time windows) and fine-grained textual (co-occurred event word groups within event windows) event details from social media data streams in (near) real-time, considering statistics and linguistics of underlying data.
2. We introduce a localised version of Dendrogram Level (DL) Similarity named *Local Dendrogram Level (LDL) Similarity*, a similarity change calculation technique named *Positive Similarity Change* and a novel clustering approach along with our approach, which can be used as sole components.
3. We utilise self-learned word embeddings and unsupervised learning techniques in WhatsUp to develop a more robust event resolution approach, overcoming the limitations: less semantic involvement, inability to capture linguistics of underlying data and less expandability to different languages, platforms and domains.

[1] WhatsUp implementation is publicly available onhttps://github.com/HHansi/WhatsUp.



(a) Embed2Detect's output (b) WhatsUp's output

Fig. 1. Embed2Detect vs WhatsUp.

4. We design a comprehensive set of metrics that automatically evaluate temporal and textual details of detected events and release its implementation, aiming to support related evaluations in a unified way.[2]

The rest of this paper is organised as follows. Section 2 discusses the previous work related to event detection in social media. Section 3 describes the problem targeted by this research. Section 4 introduces the proposed approach: WhatsUp for event detection in social media data streams. Section 5 comprehensively describes the conducted experiments and obtained results. Finally, Section 6 concludes the paper with a discussion.

2. Related work

Previous research has proposed various approaches for event detection in social media. Online clustering is a commonly used technique to capture events, including text and time. However, considering some limitations associated with online clustering, recent research focused more on approaches based on data bursts and dynamics targeting the detection of event text, time or both. We further discuss these popularly focused areas below. Additionally, a recent survey study [19] can be referred to for more details about available event detection approaches, including their comparisons.

Online Clustering: Online/incremental clustering is the process of assigning each document to a cluster as documents arrive [20]. The generated clusters provide the textual details of events, and the document which initiated the cluster becomes the first story discussing a novel event that happened at that time. To represent documents during this process, term and document frequency-based vectors [21,22,16] as well as vectors from word embedding models such as Word2Vec [18,23] were commonly used. Since word embeddings capture the semantics of the text, they were found to be more effective than frequency-based vectors [17]. Additionally, different aspects of the events and social media documents were also involved in the clustering process to improve its effectiveness. [15] suggested using semantic classes (proper noun, hashtag, location, mention, common noun and verb) of incoming tweets to find the best cluster match considering their informativeness towards events. Focusing more on the social aspect, [13] proposed to represent a tweet as a social object, including user, time and location details in addition to the textual content. However, it is time and space complex to compare a document with all historic event clusters with increasing data volume. As a solution, a lifespan was defined for clusters so that the inactive clusters can be removed [13,18]. Also, candidate cluster generation methods were suggested to reduce the number of comparisons at a new document arrival. [15] used semantic term indexing and [16] used entity-cluster inverted indices to filter candidate clusters. Further, [16] suggested representing clusters using a central centroid calculated only using the most similar L members of a cluster to faster the cluster update process. However, using rule-based approaches to extract special entities required for candidate cluster detection negatively affects these approachs' expandability to other languages.

Social media data contain personal updates, general discussions, advertisements, etc., except the newsworthy data. Since online clustering processes all incoming documents, resulting clusters can also hold non-event details. Thus, there was a focus on processing only the important documents or pruning detected clusters to identify newsworthy events. Following this insight, bursts in data streams, temporal rule dynamics and community dynamics were commonly used to detect events.

Bursts: In communication streams, bursts involve the transmission of a larger amount of data than usual over a short time. We can expect bursts in data streams at events due to the high user involvement in transmitting data. Following this idea, there was a tendency to detect data bursts and only process the corresponding data to detect events. In this way, more focus could be provided on possible event data, reducing the processing and improving the performance. For instance, [24] suggested finding bursty word n-grams at a time slot using temporal document frequency-inverse document frequency and applying hierarchical clustering on those n-grams to extract event text. However, frequency-based measures fail to differentiate events from general topics (e.g. car, food, music, etc.) because such topics are also discussed massively on social media. Later research proposed using word acceleration to detect bursts, overcoming this limitation in frequency-based measures [25]. Also, there was a focus on the social aspect, considering its importance on social networks and impact on events. [12] suggested using mention anomalies in Twitter data to detect data bursts in an offline manner. They involved a word co-occurrence and temporal correlation-based approach to extract event text during bursts. Other recent research suggested an improved version of Twevent [10] considering different user diversity-based measures (i.e. retweet count

[2] Event Evaluator implementation is available on https://github.com/HHansi/WhatsUp/tree/master/experiments/twitter_event_data_2019/evaluation.

and follower count) to detect bursty text segments in a corpus [11]. They clustered the segments using the Jarvis-Patrick algorithm to identify event text. Also, this approach used Wikipedia as an external knowledge base to ensure the meaningfulness of segments. Overall, the incorporation of the social aspect improved the performance than only involving text frequency-based measures [12,11]. However, it could limit the expandability of the approach because social aspect measures are mostly dependent on the underlying social media platform. Similarly, external knowledge bases also introduce restrictions depending on their availability and language coverage.

Pattern Dynamics: Pattern and rule mining discover item sets in data and their relationships. Following this insight, [14] proposed an approach based on the temporal dynamics of Association Rules (AR) in tweets to detect event occurrences/time. They used frequent patterns in hashtags to generate ARs and revealed that specific rule types (unexpected and emerging) have a high impact on identifying event occurrences. Consequently, in later research, there was more focus on High Utility Pattern Mining (HUPM) because it finds the high in utility item sets [26,27]. When only the frequency is considered, popularly discussed items can also be captured in addition to the event-related items, but the involvement of utility help overcome this issue. [26] used Local Weighted Linear Regression (LWLR) to calculate term novelty and combined it with term frequency to measure term utility. They clustered the detected patterns using a graph-based approach to extract event text. A similar approach is followed by [27], along with a utility calculation based on the growth rate in word frequency. However, these HUPM-based approaches were designed only to capture textual event details in a given corpus without focusing on temporal details, adhering to the original scope of pattern mining approaches.

Community Dynamics: A community is a group of elements that shares common properties. Thus, their temporal variations over data streams can be utilised to detect events. Following this idea, different approaches based on graph theory, clustering, and topic modelling were commonly used to identify communities by previous research. [28] suggested generating keyword graphs per time slot to compare their content similarities at consecutive slots to detect events in each slot. [29] suggested a similar approach using the communities detected from document graphs. Even though graphs comprehensively capture all the intermediate relationships, they can be dense, increasing the computational complexity for high data volumes. Involving clustering techniques, [30] hierarchically clustered the posts to identify communities and used a spatio-temporal similarity of named entities to compare communities. Other recent research used an affinity propagation algorithm to generate clusters and a non-negative matrix factorisation-based method for comparisons [31]. Following the popularity of topic models, [32] used Latent Dirichlet Allocation (LDA) to generate communities per time window and analysed their temporal evolution using Jensen-Shannon divergence (JSD). Additionally, they transferred parameters from the previous topic model to the next model to maintain the online fashion. A similar approach was used by other research, but they used RL-LDA, a modified version of LDA incorporating the retweeting behaviour [33]. Rather than focusing on temporal or platform-based modifications to topic models, some research involved advanced community comparison techniques to improve event detection. [34] suggested comparing topics in consecutive windows using novelty and fading calculated by LWLR and Kullback–Leibler divergence (KLD). Maintaining simplicity, other recent research suggested using shared topic words and temporal variations in topic word count to compare topics [35]. Overall, these modifications to the traditional topic models improved the event detection performance and allowed capturing of the events' temporal details. However, we noticed the requirement to pre-define topic count by topic models as a critical limitation when processing highly dynamic social media data streams.

In summary, previous research focused on detecting event text, time or both, but comparatively, a high focus is given to textual details. The majority of the approaches targeted online processing, while a few targeted offline processing. Among them, online detection of event text and time provides more comprehensive output helpful for real-time information extraction tasks. To detect event time, available approaches mainly used various statistical measures (e.g. word/document frequency, retweet count, JSD, KLD, etc.) without involving linguistics. However, to extract event text, some approaches involved linguistics using external knowledge bases, rule-based approaches, and prediction-based approaches. Overall, the usage of knowledge bases and rules negatively affects the expandability of those approaches. Similarly, using pre-trained embedding models (prediction-based approaches) adds constraints depending on the model availability and incapabilities to capture corpus-specific linguistics. Considering these limitations, in this research, we target proposing a novel method capable of online detection of both event text and time, using statistics and linguistics, which are essential for effective information extraction from textual data.

3. Problem definition

The problem targeted by this research is automatically detecting temporal and textual event details in (near) real-time from social media data streams. Various definitions were used by previous research to describe events [28,12,15,8]. Among them, we consider the Definition 1 [8] because it was designed by combining the main ideas of previous research. The concept behind a data stream is introduced with Definition 2.

Definition 1. *Event:* An incident or activity which happened at a certain time and discussed or reported significantly in social media.

Definition 2. *Social Media Data Stream:* A continuous and chronological series of posts or documents D: $d_1, d_2, \ldots d_{i-1}, d_i, \ldots$generated by social media users.

Available event detection approaches mainly consider two types of input data stream: general and filtered (or focused). In the general scenario, the whole data stream D is processed [36,16]. In the filtered scenario, a user-centred data stream $D\prime$ extracted from the whole data stream D is processed. The filtering was commonly done based on keywords or locations. In keyword-based filtering, a domain-specific data stream will be extracted using a set of keywords [37,18]. In location-based filtering, a data stream composed of a set of documents posted by users in a particular location will be extracted [10,12]. Considering the volume and diversity of the whole data stream and user requirements, processing a filtered data stream was found to be more useful [37,8]. Among the two filtering techniques, the location-based method could add unnatural restrictions due to the unavailability of locations in all posts or user accounts and the possibility to report an event that happened at a particular location by a user located elsewhere (e.g. report while travelling or watching television) [8]. Following these findings, this research focuses on a filtered data stream extracted using the keywords. Definition 3 introduces the concept of the keyword-based filtered data stream.

Definition 3. *Filtered Data Stream:* A filtered or narrowed down data stream which consists of posts that contain at least one of the selected keywords.

To facilitate event time extraction, we use the concept of time windows which is widely used in previous research [37,14,11,8]. Usage of time windows allows system customisation based on the characteristics of the targeted domain and intended update rate. Upon the data arrival via a filtered stream $D\prime$, data will be separated into time windows W: $W_1, W_2, \ldots W_{t-1}, W_t, \ldots$(durations of time) of user-defined length l to assess each window W_t to detect the windows W^d when events occurred/event windows (Definition 4).

Definition 4. *Event Occurred Time Window/Event Window:* Duration of time W_t, where at least one event e has occurred.

To facilitate fine-grained event text extraction, we introduce the concept of co-occurred events (Definition 5), considering the possibility to happen multiple events during an event window W_t^d. Per event e that happened during W_t^d, we focus on extracting a word/token cluster c, which expresses the event's textual details.

Definition 5. *Co-occurred Events:* Multiple events $E_{W_t^d}$ happened/reported within the same time window (event window W_t^d).

In summary, the system described in this paper aims to identify event windows W^d and token clusters that represent the co-occurred events within event windows $E_{W_t^d} : W_t^d \in W^d$, covering the temporal and fine-grained textual event details by processing an incoming filtered data stream $D\prime$ in (near) real-time. Considering the diversity in events, this system requires allowing customisations depending on the focused domains and interesting events using hyper-parameters that mainly capture the details such as event significance, cohesion and update rate. Table 1 shows the list of notations used in this paper.

Table 1
Summary of notations used in the paper.

Notation	Description
d	document/post in a data stream D
$D\prime$	filtered data stream
W	time windows during a period
W^d	detected event windows during a period
W_t	window in W at time t
W_{t-1}	window in W at time $t-1$ (previous time window to W_t)
w	word/token
c	word/token cluster
e	event
E_{W_t}	set of events happened during W_t/co-occurred events during W_t
V_t	word embeddings/vector space learned from W_t
$vocab$	vocabulary
f_t	token frequencies of $vocab$ at time t ($vocab_t$)
dl	dendrogram level
$dl_{r \to x}$	number of dls from root; r to node; x
$dl_{(w_i, w_j)}$	number of shared dls between tokens; w_i and w_j from root
L	set of leaf nodes in a dendrogram
α	change threshold
β	frequency threshold
s	cluster word similarity
m	cluster word count
η	novelty threshold
κ	keyword count
l	time window length

4. WhatsUp

WhatsUp, the algorithm proposed by this paper for event detection has three main steps (1) data processing, (2) event occurred time window identification and (3) event cluster detection. The overall flow is illustrated in Fig. 2. The first step separates the data stream into time windows and per window, learns word embeddings and extracts statistical information. The next step identifies event windows. If any window is identified as an event window, the final step detects co-occurred event clusters in that window. For event window and cluster detection, we use temporal changes in linguistics (syntax and semantics) captured by self-learned word embeddings and their hierarchical relationships and statistics. To the best of our knowledge, both linguistical and statistical features have not been involved in previous research to identify event time and text of co-occurred events together, except for either time or text. Also, since we only involve self-learning and unsupervised techniques, our approach is generally applicable to any language, platform or domain. However, depending on the characteristics of the targeted events, it requires setting a few hyper-parameters for customisation. Each step is further described in Sections 4.1–4.3. Finally, Algorithm4 summarises the top-level idea of WhatsUp, combining all steps.

To conduct the experiments, we used *Twitter Event Data 2019* [8], considering its recency and event coverage. It consists of two data sets from sports and political domains. The sports data set was based on a football match during the English Premier League 19/20. We use sample data from this football data set to explain the concepts used to develop our methodology, only considering the simplicity and cohesion of events in this domain. More details about the data sets are available in Section 5.1.

4.1. Data processing

Under data processing, we initially separate the incoming data stream into windows of non-overlapping fixed periods using the sliding window model [38]. Previous research commonly used window-based approaches considering the high data volume and requirements to capture temporal details [28,14,27]. Also, the time window length can be adjusted depending on the evolution of events in the targeted domain.

Then, we extract each window's linguistical and statistical details to support event window identification and cluster detection. To capture linguistics, embedding models are trained per window using their data. This training captures characteristics and expressions unique to each window's data. While selecting an algorithm to learn embeddings, we mainly focused on semantic accuracy and efficiency suitable for real-time processing. Considering the semantic accuracy, transformer-based embeddings (e.g. BERT, XLM-R) showed improved performance in many NLP applications recently [39–41]. However, they are unsuitable for real-time processing due to the long training time and additional computational complexities introduced to event detection by contextual word senses [8]. Therefore, we decided to use Word2Vec models for this research, considering their ability to learn effective embeddings faster. Among Continuous Bag-of-Words (CBOW) and Skip-gram architectures, the Skip-gram model is selected because it resulted in high semantic accuracy [42,43]. However, we facilitate the easy integration of any embedding model which fulfils the targeted requirements (i.e. good semantic accuracy and efficient learning) by developing the word embedding learner as a separate component. As statistical details, token frequencies at each time window are calculated.

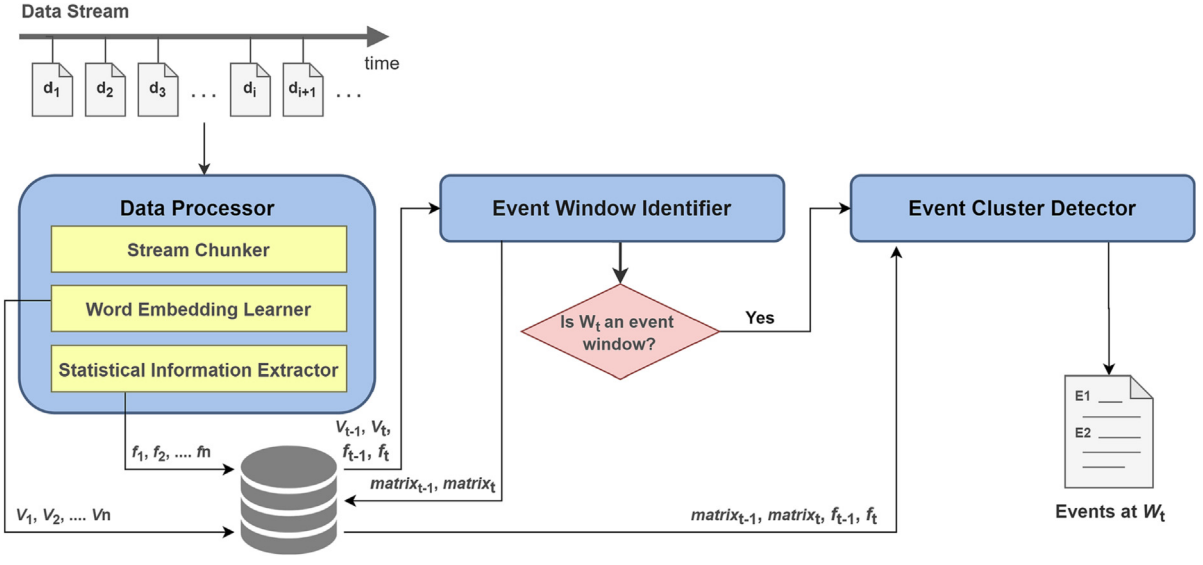


Fig. 2. Overview of the proposed method – WhatsUp.

4.2. Event window identification

We utilise the idea proposed with Embed2Detect [8] to identify event windows. Primarily, a time window W_t is marked as an event window if its temporal textual change compared to the previous time window W_{t-1} is above a predefined threshold (α). The final change is calculated by aggregating two measures based on the temporal variations in token clusters and vocabularies, capturing underlying linguistics and statistics. Any value between 0 and 1 can be picked for α, as normalised values are involved for change calculation, and it mainly defines the significance of the events targeted by the user. The higher the value, the more significance it has. Highlighting some limitations recognised in the available approach, we propose different strategies to calculate cluster similarity, temporal similarity change and vocabulary change to improve the performance of event window identification in this research, as described in Sections 4.2.1–4.2.3. The overall idea of event window identification is summarised in Algorithm 1. It processes a chronological stream of time windows (at least two consecutive windows W_{t-1}, W_t) and returns the windows which are recognised as event windows.

Algorithm 1: Event Window Identification

input: W : time windows during a period $([W_0, W_1, ...W_{i-1}, W_i, ...])$, α : change threshold
output: W^d : detected event windows

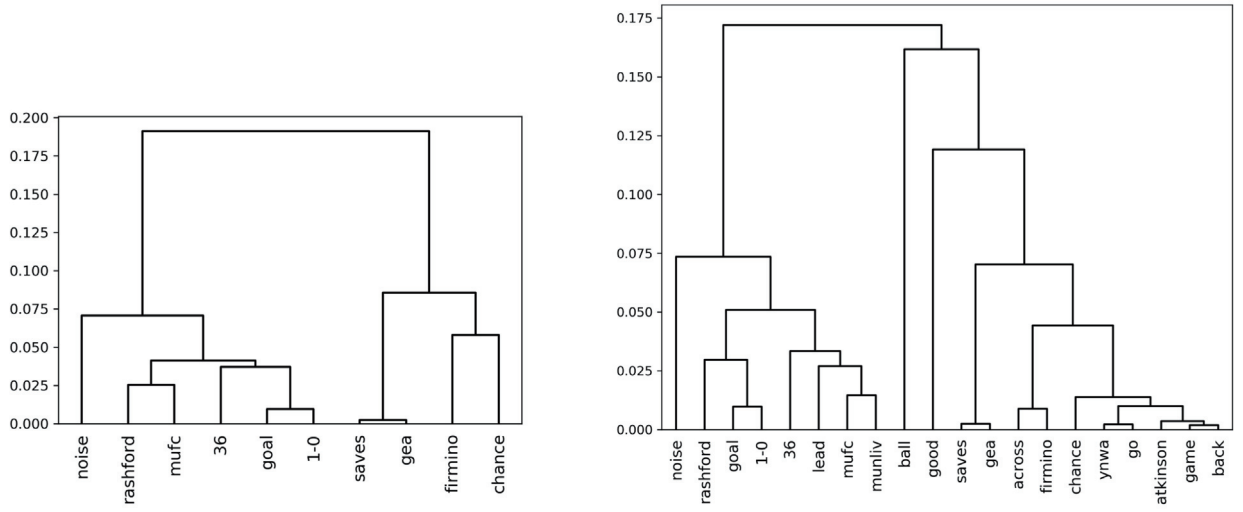
$W^d \leftarrow []$; `// empty array to keep event windows`
for *t=1 to length(W)* **do**
  $vocab_{t-1} \leftarrow$ vocabulary at W_{t-1};
  $vocab_t \leftarrow$ vocabulary at W_t;
  $V_{t-1} \leftarrow$ vector space at W_{t-1};
  $V_t \leftarrow$ vector space at W_t;
  // Compute cluster similarity
  $commonVocab \leftarrow$ common vocabulary for $vocab_{t-1}$ and $vocab_t$;
  $N \leftarrow length(commonVocab)$;
  $matrix_{t-1} \leftarrow$ similarity matrix of $commonVocab$ using V_{t-1};
  $matrix_t \leftarrow$ similarity matrix of $commonVocab$ using V_t;
  // Compute similarity change
  $diffMatrix \leftarrow matrix_t - matrix_{t-1}$;
  $clusterChange \leftarrow$ overall change calculated using $diffMatrix$;
  // Compute vocabulary change
  $vocabChange \leftarrow$ change calculated using $vocat_t$ and $vocab_{t-1}$;
  // Compute overall change
  $overallChange \leftarrow aggregate(clusterChange, vocabChange)$;
  if $overallChange \geq \alpha$ **then**
    $W^d.add(W_t)$;
  end
end
return W^d

Before moving into the calculations, the text needs to be preprocessed for effective results. Under preprocessing, we mainly targeted removing the uninformative tokens using general procedures which are not strictly dependant on domains or languages. Punctuation marks and stop words in the text are removed as uninformative tokens. Also, the tokens with low frequency than a predefined threshold (β) are removed as outlier tokens.

4.2.1. Cluster similarity calculation

A token cluster-based approach was used to measure text similarity within each window in Embed2Detect. Mainly, token clusters were targeted because ideas discussed in data streams form word groups, and their variations can be used to recognise newsworthy events. Embed2Detect proposed the measure named *Dendrogram Level (DL) similarity* to calculate cluster similarity. As the name depicts, it mainly captures the structural relationships in dendrograms.

A dendrogram is a tree diagram that illustrates the hierarchical relationships between objects [44]. Sample dendrograms on selected word sets from tweets posted during the first goal of the football data set are shown in Fig. 3. At the leaf level, each token is in its own cluster. Horizontal lines, which consider as levels during the DL similarity calculation, represent merges between clusters that occur considering the distance. Merges between closer clusters happen at low distances and distant clusters at high distances. To generate dendrograms per window, hierarchical agglomerative clustering (HAC) was applied to word embeddings. The average scheme was used as the linkage method since it involves all the elements in clusters during distance calculation. In average linkage, the distance between two clusters C_i and C_j is measured using



(a) Sample dendrogram 1 (b) Sample dendrogram 2

Fig. 3. Sample dendrograms (y-coordinate denotes the cosine distance and x-coordinate denotes the selected words).

$$D(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{w_p \in C_i} \sum_{w_q \in C_j} d(w_p, w_q), \tag{1}$$

where $d(w_p, w_q)$ represents the distance between cluster elements w_p and w_q which belong to the clusters C_i and C_j, respectively [45]. Cosine distance was used for distance calculation, considering its proven effectiveness on textual measurements [42,43,46] and unbias to word frequency [47].

Dendrogram Level (DL) similarity: DL similarity between a token pair w_i and w_j is calculated by normalising the number of shared levels between those tokens from the root of the dendrogram (Eq. 2).

$$DL\ Similarity_{(w_i, w_j)} = \frac{dl_{(w_i, w_j)}}{max(dl_{r \to x} : x \in L) + 1} \tag{2}$$

The numerator of Eq. 2 represents the number of shared dendrogram levels between w_i and w_j from the root. The denominator represents the maximum number of levels between the root and all leaf nodes. Leaf level is also considered while calculating the denominator to make the similarity between the same word equals 1. Words which are not located in nearby clusters will get a low DL similarity (closer to 0). For example, let us consider the word *'goal'* and pair it with the player name who scored the goal *'rashford'* and some other player name *'firmino'*. If the DL similarities are calculated based on the dendrogram in Fig. 3a, we will receive the values $DL\ Sim._{(rashford,goal)} = 3/6 = 0.5$, and $DL\ Sim._{(firmino,goal)} = 1/6 = 0.167$ with a high similarity between *'goal'-'rashford'* than the other pair. To calculate the denominator, $dl_{r \to goal}$ (or $dl_{r \to 1-0}$) is considered because this path has the maximum number of *dl*s.

The main limitation we noticed in DL similarity is it can be distorted when noise is added to the dendrogram. Analysing from the viewpoint of social media data, it is common to post details about general discussions (or non-event data) during an event window. Even though we focused on filtered data, within that data also, general discussions can be included. Due to this distortion, the maximum level count of the dendrogram can be increased, resulting in a high denominator value for Eq. 2. In such situations, DL similarity becomes incapable of assigning high values to word pairs that are very close. For example, dendrogram 2 in Fig. 3b is generated using the same data used for dendrogram 1 in Fig. 3a, but with more words. In dendrogram 1, we limited the word set to the keywords of events (a) Firmino's attempt saved by Gea and (b) first goal for Manchester United FC by Rashford. To generate the dendrogram 2, we included more words such as *'good', 'go', 'ynwa', 'game'* and *'back'* which are not event-related directly. These additions change the whole structure of the dendrogram, making it closer to a real scenario. Now the dendrogram has 10 highest number of levels ($dl_{r \to game}$). If the similarity values are calculated for the word pairs used above, we will get the values $DL\ Sim._{(rashford,goal)} = 4/10 = 0.4$, and $DL\ Sim._{(firmino,goal)} = 1/10 = 0.1$.

Even though the similarity values can differentiate closer and distant words, we are receiving a very low similarity for the pair *'rashford'-'goal'* even if the tokens are located very closely. This happened due to the noise added to the dendrogram by non-event words. In reality, since event detection is the final aim of the targeted approach, it is impossible to filter event-related words beforehand to generate a less noisy dendrogram. To overcome this limitation in DL similarity, we propose a localised version named *Local DL similarity*.

Local Dendrogram Level (LDL) similarity: The main difference in LDL similarity compared to DL similarity is, to calculate the denominator, it only considers the levels between root and targeted word pair as in Eq. 3. It prevents unnecessary normalisation of the value due to a rare long path in the dendrogram.

$$LDL\ Similarity_{(w_i,w_j)} = \frac{dl_{(w_i,w_j)}}{max(dl_{r\rightarrow x} : x \in \{w_i, w_j\}) + 1} \tag{3}$$

With LDL similarity, we will receive the values $LDL\ Sim._{(rashford,goal)} = 3/6 = 0.5$, and $LDL\ Sim._{(firmino,goal)} = 1/6 = 0.167$using dendrogram 1. For both word pairs, $dl_{r\rightarrow goal}$ is considered to calculate the denominator value as this path contains the maximum number of levels. For dendrogram 2, we will get the values $LDL\ Sim._{(rashford,goal)} = 4/6 = 0.667$, and $LDL\ Sim._{(firmino,goal)} = 1/7 = 0.143$. For *'rashford'-'goal'*, $dl_{r\rightarrow goal}$ and for *'firmino'-'goal'*, $dl_{r\rightarrow firmino}$ are considered to calculate the denominator values. According to the results, unlike the DL similarity, LDL similarity is capable of calculating effective values based on nearby clusters even though the dendrogram is noisy. Therefore we use LDL similarity to calculate the cluster similarity between words in this research.

Summarising the characteristics of DL and LDL similarities, both compute normalised text similarity values within the range of 0–1, based on structural relationships of a dendrogram. Thus, these measures are relative to the dendrogram or underlying corpus rather than absolute. They intend to result in a value closer to 1 if a word pair is similar (or closer) and a value closer to 0 if the pair is distant. However, DL similarity could fail to produce high values for closer words if the dendrogram is distorted with noise, and we proposed LDL similarity, overcoming this limitation. We provide a summary which compares these two measures in Table 2. Also, even though we only focus on similarities in this research following our requirements, both of these metrics can convert to distance measures, similar to cosine similarity, by taking $1 - (L)DL\ Sim$. In this way, the number of non-shared levels between the word pair becomes prominent, resulting in the difference/distance between words. Contrary to similarity measures, values closer to 1 will represent high distances in distance calculations.

After calculating token similarities, they are formatted into a matrix to support efficient future computations. This similarity matrix is a square matrix of size $N \times N$ where N is the number of tokens in the vocabulary. Each matrix cell $matrix[i,j]$ represents the cluster similarity between tokens w_i and w_j. To support matrix comparisons over consecutive time windows, a common vocabulary is used to generate consecutive matrices $matrix_{t-1}$ and $matrix_t$. Since the change at W_t needs to be calculated compared to the previous window W_{t-1} to detect event occurrences, preprocessed vocabulary at t $vocab_t$ is considered as the common vocabulary. Following this idea, Embed2Detect used common vocabularies to generate dendrograms in consecutive time windows. We use the term *relative dendrograms* to refer to such dendrograms.

Relative Dendrograms: To generate relative dendrograms for W_t and W_{t-1}, preprocessed vocabulary at t $vocab_t$ is considered as a common vocabulary. Dendrogram for W_t is generated using $vocab_t$ and word embeddings learned for W_t. Dendrogram for W_{t-1} is generated using $vocab_t$ and word embeddings learned for W_{t-1}, ignoring the words in $vocab_t$ which are not found in the vector space V_{t-1}.

Usage of a common vocabulary can negatively affect the similarity values if $vocab_t$ contains tokens that have a very low frequency at W_{t-1}. If the embedding model has not seen sufficient occurrences, it may result in vectors that are not properly learned for those tokens. This could affect the dendrogram structure and result in incorrect similarity values. A few sample similarity values calculated using relative dendrograms on football data are shown in Table 3. As illustrated in these examples, relative dendrograms return high similarities for word pairs which are rarely used during time windows. Even though the selected word pairs are emerging at t according to the ground truth, cluster similarity change between W_t and W_{t-1} will not capture that state using the resulting negative ('gini'-'movement': −0.2807) and small positive ('goal'-'1–0': 0.1308) values. To overcome this issue, we propose to use non-relative dendrograms.

Non-Relative Dendrograms: Non-relative dendrogram generation does not use a common vocabulary as relative dendrogram generation. Dendrograms for W_{t-1} and W_t will be generated using preprocessed vocabularies $vocab_{t-1}$ and $vocab_t$, respectively. This restricts the involvement of rare words for dendrogram generation, which results in the issue mentioned above. If non-relative dendrograms are used to calculate the similarities between the word pairs in Table 3, due to low frequency, 'movement' and '1–0' will not be included in $vocab_{t-1}$ and similarity between both word pairs at W_{t-1} will be zero. For W_t, the same similarities will be calculated, and cluster similarity change of both pairs will be large positive values ('gini'-'movement': 0.6667. 'goal'-'1–0': 0.9) capable of capturing the emerging state of words.

4.2.2. Similarity change calculation

After calculating token similarities at each time window, their temporal change needs to be measured. The change calculation is mainly based on the *diffMatrix* which holds the token similarity differences between W_t and W_{t-1} ($matrix_t - matrix_{t-1}$). Embed2Detect used *Absolute Similarity Change* to calculate the overall similarity change.

Absolute (Abs.) Similarity Change: Abs. similarity change is calculated as the average of absolute values in *diffMatrix* following the Eq. 4. Only the values at the upper triangular matrix except the diagonal are considered because the matrix is symmetric around the diagonal.

$$Abs.\ Similarity\ Change = \frac{\sum_{i=1}^{N}\sum_{j=i+1}^{N}|diffMatrix[i,j]|}{(N \times (N-1))/2} \tag{4}$$

The overall similarity change should be able to differentiate event and non-event windows assigning a greater change to event windows. An event can suddenly introduce all the event keywords or change an ongoing discussion by introducing a few new keywords to the data stream [14]. If all the keywords are newly introduced, they will show positive cluster changes

Table 2
Summary of cluster similarity measures.

Metric Name	Properties	Summary
DL Similarity (Eq. 2)	Reflexive Non-negative Symmetric	Measures word similarity based on structural relationships in a dendrogram. *Con:* Prone to the noise in the dendrogram.
LDL Similarity (Eq. 3)	Reflexive Non-negative Symmetric	Measures word similarity based on local structural relationships in a dendrogram. *Pro:* Less prone to the noise in the dendrogram with localised calculation.

Table 3
Sample LDL similarity values using relative dendrograms.

w_i	w_j	$f_{t-1} : w_i - w_j$	$f_t : w_i - w_j$	$LDL\ sim._{t-1}$	$LDL\ sim._t$
gini	movement	10–9	40–37	0.9474	0.6667
goal	1–0	16–2	339–244	0.7692	0.9000

compared to the previous time window. If some high impact happens to an ongoing discussion by an event, newly introduced words will become closer to already existing words, weakening some connections they had previously. Both positive and negative similarity changes should be expected in such a situation. Also, during a time window when no event happens, interest in an ongoing discussion can fade, resulting in some negative cluster changes. However, absolute change treats negative and positive values similarly, even if they contain valuable details of temporal event evolution. This issue can solve simply by using non-absolute (Non-abs.) changes, but the high availability of general discussions in social media data can unnecessarily lower the overall non-absolute change. Considering all these facts, we propose a new calculation named *Positive Similarity Change* in this research to identify event windows.

Positive (Pos.) Similarity Change: Under this calculation, a weighted average of positive values in *diffMatrix* is used to measure the overall similarity change. As the weight, the proportion of positive values in the upper triangle of the *diffMatrix* is used, as illustrated in Algorithm 2. Only the positive values are focused on because positive changes occur when words become closer at W_t than W_{t-1} or word groups appear representing emerging events.

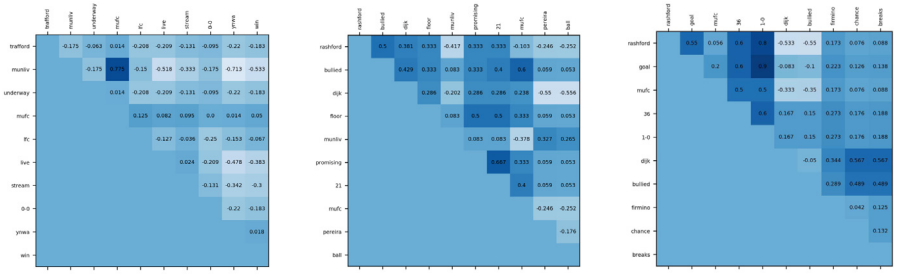
Algorithm 2: Positive Similarity Change Calculation

input: *diffMatrix:* matrix of temporal word similarity differences
output: *posChange*: positive similarity changes
$positiveValues \leftarrow []$; // empty array to keep positive values
$N \leftarrow \text{length}(diffMatrix_{rows})$;
// Get positive values in the upper triangle of *diffMatrix*
for i=1 *to* N **do**
  for j=i+1 *to* N **do**
    if $diffMatrix[i][j] > 0$ **then**
      $positiveValues.add(diffMatrix[i][j])$;
    end
  end
end
// Calculate overall positive change
$average \leftarrow \sum positiveValues/length(positiveValues)$;
$proportion \leftarrow length(positiveValues)/((N \times (N-1))/2)$;
$posChange \leftarrow average \times proportion$;
return $posChange$

To explain the effectiveness of Pos. similarity change compared to Abs. and Non-abs. changes, we selected a few sample time windows with event details from the football data set, summarised in Table 4. Fig. 4 shows the *diffMatrix* generated per these windows. For simplicity, we only used the important keywords of the selected events to generate matrices. However, dendrograms generated considering all data of the targeted window are used to calculate cluster similarities, illustrating the real scenario. At W_{t_a}, no important events happened and *diffMatrix* in Fig. 4a also has low values. At W_{t_b}, an emerging event

Table 4
Sample events occurred during English Premier League 19/20 on 20 October 2019 (Manchester United – Liverpool).

Time window	Event	Description
16:34–16:36 (W_{t_a})	-	No important event reported.
16:52–16:54 (W_{t_b})	Foul	Foul by Marcus Rashford (MUFC) on Virgil van Dijk (LFC).
17:06–17:08 (W_{t_c})	Goal	First goal by Marcus Rashford (MUFC).



(a) *diffMatrix* at W_{t_a} (b) *diffMatrix* at W_{t_b} (c) *diffMatrix* at W_{t_c}

Fig. 4. *diffMatrix* generated for time windows W_{t_a}, W_{t_b} and W_{t_c}).

Table 5
Comparison of similarity change calculation methods.

Calculation	Matrix(a)	Matrix(b)	Matrix(c)
Abs. Similarity Change	0.2005	0.3388	0.2981
Non-abs. Similarity Change	−0.1467	0.2786	0.2092
Pos. Similarity Change	0.0269	0.3087	0.2537

happened, and it results in several high positive values in *diffMatrix* in Fig. 4b. At W_{t_c}, an ongoing discussion is altered by an event, and thus *diffMatrix* in Fig. 4c has a combination of negative and positive values. Overall similarity changes calculated for each of these matrices using Abs., Non-Abs. and Pos. calculations are summarised in Table 5.

As can be seen in Table 5, Abs. change returned a high value for W_{t_a} even though it has no events. Non-abs. change could effectively capture the non-existence of events (at W_{t_a}) and suddenly happened events introducing all the keywords (at W_{t_b}), but it returned a low value when an event altered an ongoing discussion (at W_{t_c}). Pos. change showcased the ability to effectively capture all scenarios with low value at W_{t_a} and high values at W_{t_b} and W_{t_c}. Therefore, we propose to use Pos. similarity change to calculate overall similarity change.

In summary, Abs., Non-abs. and Pos. change compute an overall value for a matrix within the ranges of 0–1, −1–1 and 0–1, incorporating different impacts based on their attributes. As we discussed above, for this research, Pos. similarity change has the best appropriate characteristics because it particularly targets positive values of the *diffMatrix*, which represent words that became closer over time. If a large proportion of the *diffMatrix* have positive values, Pos. change will return a high value (≈ 1), indicating a situation where more words become closer due to an event(s). Contrarily, a low value (≈ 0) means no events or no notable positive changes happen over time. However, we report a comparison between each measure, including their pros and cons in Table 6 to highlight their characteristics helpful for other research.

4.2.3. Vocabulary change calculation and aggregation

In addition to the temporal change in token similarity, vocabulary change is also considered to calculate overall change. In Embed2Detect, vocabulary change is calculated by considering the temporal token variations.

Vocabulary Change (VC): VC is calculated based on consecutive vocabularies $vocab_t$ and $vocab_{t-1}$ following the Eq. 5.

$$Vocabulary\ Change(VC)_{(t-1,t)} = \frac{|w : w \in vocab_t \ and \ w \notin vocab_{t-1}|}{|vocab_t|} \tag{5}$$

The numerator of Eq. 5 represents the cardinality of new tokens that appeared in the vocabulary of W_t compared to W_{t-1}, and the denominator represents the size of the vocabulary of W_t. In this research, we propose to involve frequency changes of tokens in addition to the token variations to calculate vocabulary change. This modification allows focusing on the significant changes ignoring the slight modifications that happened to the discussions.

Table 6
Summary of similarity change measures.

Metric Name	Summary
Abs. Similarity Change (Eq. 4)	Computes a normalised overall value for a matrix, treating both positive and negative values similarly. *Con:* Unable to distinguish positive and negative values and their proportions.
Non-abs. Similarity Change	Computes a normalised overall value for a matrix, accounting for both positive and negative values. *Con:* The output gets dominated by the majority sign.
Pos. Similarity Change (Algorithm 2)	Computes a normalised overall value for a matrix, focusing on a particular sign (positive). *Pro:* Prioritises the focused sign (positive) and reduces the impact by the opposite sign (negative). *Pro:* Can be easily customised to focus on the opposite sign (negative).

Frequency Difference-based Vocabulary Change (FDVC): For this calculation, we consider new tokens that appeared at W_t with a frequency difference above the threshold β, which is used to filter outlier tokens (Eq. 6).

$$FDVC_{(t-1,t)} = \frac{|w : w \in vocab_t, w \notin vocab_{t-1} \text{ and } f(w)_t - f(w)_{t-1} > \beta|}{|vocab_t|} \tag{6}$$

Both VC and FDVC return normalised change values within 0–1. A value closer to 0 indicates that $vocab_{t-1}$ is almost the same as $vocab_t$. If many new tokens are added to the $vocab_t$, VC will return a high value without considering their significance or frequency difference. Thus, this calculation will treat tokens that newly appeared once (mostly due to background discussions) and multiple times (mostly due to events) similarly. Overcoming this limitation, FDVC only returns high values if a large proportion of $vocab_t$ holds new tokens with higher frequency differences above a threshold compared to $vocab_{t-1}$, capturing event occurrences. Thus, we only involve FDVC for the computations in this research. A summary comparing both measures is available in Table 7.

After calculating the temporal textual change using two measures: similarity change and vocabulary change, these values need to be aggregated to calculate the overall change. Similar to the Embed2Detect approach, we also consider maximum and average calculations as aggregation methods to preserve the simplicity of the proposed approach.

4.3. Event cluster detection

After identifying the event windows, the next phase detects co-occurring events/event clusters within those windows to extract fine-grained event details. Within an event window, one or more events would have been reported, and it is important to detect the details of each event separately to let users get quick and concise event updates. For cluster detection, we propose a novel method which focuses on identifying important keywords to extract clusters using the tokens closer to the keywords. Basically, our method has two steps: (1) token ranking (identifying important keywords) and (2) cluster generation, being flexible for easy adjustments for different events. To rank the tokens, we assign a weight to each token considering its textual similarity change over time, as sudden incidents within data streams result in textual changes. If a token has a high positive change value, it means that many words became closer to that token over time, indicating that it is an important keyword of an event or discussion. Thus, we sort the tokens with positive textual change in descending order to identify the important event keywords. We only consider the words in the vocabulary for this step, except for emojis and non-alphanumeric characters, because the target is to rank keywords. We use LDL similarities calculated using word embeddings and their hierarchical relationships and token frequency-based measures for the weight calculation, capturing underlying linguistics and statistics as described in Section 4.3.1. Then, token groups are identified using linguistically nearby tokens to the ranked tokens (i.e. keywords) as such groups represent the ideas discussed together. Similar to the token weighting, we use LDL similarity between tokens to extract nearby tokens, capturing the linguistical relationships as described in Section 4.3.2. Finally, as additional information helpful to users, we occupy each cluster with a novelty measure that indicates

Table 7
Summary of vocabulary change measures. Since these measures are designed to compute vocabulary change of consecutive time windows, the reflexive property is not applicable to them.

Metric Name	Properties	Summary
VC (Eq. 5)	Non-negative Non-symmetric	Measures vocabulary change over time. *Con:* Incorporates significant and insignificant changes with the final value.
FDVC (Eq. 6)	Non-negative Non-symmetric	Measures vocabulary change over time only targeting significant changes. *Pro:* Allows to customise the targeted significance of changes by setting a threshold.

the cluster's newness or newsworthiness. After detecting the clusters, we prune them to filter emerging events as described in Section 4.3.3. The overview of our event cluster detection approach is illustrated in Algorithm 3.

Algorithm 3: Event Cluster Detection

input: W_t^d : detected event window, s or m : cluster detection threshold, η or κ : cluster pruning threshold
output: $E_{W_t^d}$: co-occurred events (event-described token clusters) at W_t^d

$vocab_t \leftarrow$ vocabulary at W_t^d;
$weightedWords \leftarrow \{\}$; // empty dictionary to keep word-weights
$C \leftarrow []$; // empty array to keep clusters
// Rank tokens which showed a temporal textual change
for w *in* $vocab_t$ **do**
  $weight \leftarrow$ weight calculated for w using temporal textual change;
  if $weight > 0$ **then**
    $weightedWords.add(w, weight)$;
  end
end
$weightedWords.sort(descending = True)$;
// Generate clusters
for w *in* $weightedWords$ **do**
  if w *not appeared in any token cluster in* C **then**
    $c \leftarrow$ nearby tokens to w based on s or m;
    $c \leftarrow [w, c]$;
    $Novelty_c \leftarrow$ novelty calculated on c;
    $C.add(c, Novelty_c)$;
  end
end
// Prune clusters
$E_{W_t^d} \leftarrow$ pruned clusters from C as events based on η or κ;
return $E_{W_t^d}$

Unlike the majority of previous research, we propose clustering tokens (not documents). As tokens, we use both words and emojis as emojis play a crucial role in expressing public opinions on social media nowadays. When the token level is considered, multiple events described in a document can be separated, and connections between tokens that do not appear together in documents can be captured. Due to the recent increments made to character limits by social media services, a single document can contain multiple events. Also, due to the diversity of users, the same event can be expressed differently in multiple documents requiring capturing token relationships to get complete event information. Further, our algorithm has advantages compared to the traditional clustering algorithms. Unlike flat clustering algorithms (e.g. K-means), our algorithm does not require the cluster count to be predefined, which is unpredictable due to the dynamicity of data streams. Unlike hierarchical algorithms (e.g. hierarchical agglomerative, hierarchical divisive), our algorithm is capable of assigning the same token to multiple clusters. In reality, there can be situations when the same token is shared among multiple events (e.g. 'goal' is shared between events: *team MUFC scored a goal* and *VAR for the goal is in progress*).

4.3.1. Token weight calculation

To rank tokens, we assign weights to them based on their temporal textual change. We experiment with the following weight calculations in this research. To make the similarity-based computations faster, we use the *diffMatrix* generated during event window identification (Algorithm 1) as a row of the matrix *diffMatrix*[w] contains the values $\{LDL_{(w,w_i)} : w_i \in vocab_t\}$.

1. **Pairwise Similarity Difference (PSD):** The highest similarity difference of the token pairs which contain the targeted token.

$$PSD_w = max(LDL\ sim_{\cdot(w,w_i)} : w_i \in vocab_t)$$

2. **Average Similarity Difference (ASD):** Average similarity difference of all token pairs which contain the targeted token. To calculate the average, we use the positive similarity change calculation proposed in Algorithm 2 considering its effectiveness. In the above, we showed how this calculation could be used for a matrix (Algorithm 2), but similarly, it can also be used with a row of the matrix (similarities between all token pairs which contain the targeted token).

$$positiveValues = \{LDL\ sim._{(w,w_i)} : w_i \in vocab_t\ and\ LDL\ sim._{(w,w_i)} > 0\}$$
$$ASD_w = \frac{\sum positiveValues}{|positiveValues|} \times \frac{|positiveValues|}{|vocab_t|}$$

3. **Frequency Difference (FD):** Normalised frequency difference of the targeted token.

$$FD_w = \frac{f(w)_t - f(w)_{t-1}}{max(f(w)_t, f(w)_{t-1})}$$

4. **Average Aggregations (AVG-*-FD):** Average of similarity-based value and FD. Since there are two similarity-based measures, average aggregation also returns two measures *AVG-PSD-FD* and *AVG-ASD-FD*.
5. **Maximum Aggregations (MAX-*-FD):** Maximum of similarity-based value and FD. Similar to the average aggregation, this also has two measures *MAX-PSD-FD* and *MAX-ASD-FD*.

According to the experiment results described in Section 5.3.2, we found that AVG-ASD-FD outperforms the other weight measures.

4.3.2. Cluster generation

We propose to generate clusters by extracting tokens linguistically closer to the top-ranked keywords. To measure the similarities between tokens, we use LDL similarity computed using dendrograms generated on self-learned word embeddings since it incorporates the token relationships in the vector space and the hierarchical structure, capturing the underlying linguistics. Additionally, we require a threshold to extract nearby words, and we consider two threshold types as follows.

1. **Similarity-based Threshold(s):** Consider all tokens with similarity $\geqslant s$ as nearby words. This is similar to the similarity thresholds used by hierarchical and online clustering algorithms.
2. **Count-based Threshold(m):** Consider m most similar tokens as nearby words.

We conducted experiments using both thresholds and revealed that they have different qualities which are helpful for users depending on their interests and domains specificities as described in Section 5.3.2.

After identifying the clusters, we assign a novelty measure for each cluster to improve its informativeness. The novelty of cluster c is calculated as the average of token weights belonging to it (Eq. 7). Overall, it measures the temporal textual change of that cluster compared to the previous time window and indicates the event significance.

$$Novelty_c = \frac{\sum weight_w : w \in c}{|c|} \tag{7}$$

4.3.3. Cluster pruning

Since detected clusters can hold emerging events and background discussions, they need to be pruned to extract events. The commonly used approach is to assign clusters a value that indicates their importance and filter out the less important clusters using a threshold [15,13,11]. Following this idea, we also suggest pruning clusters using a threshold for cluster novelty (η). Nonetheless, such a threshold highly depends on the underlying data. Thus, even within the same domain, using a static value for multiple time windows will be incorrect if the nature of data and their evolution are different in those time windows. Also, due to the dynamicity and diversity of events, it will be hard to pick such a threshold value by analysing previous data and heuristics.

Considering all these drawbacks, we propose a keyword count-based technique to prune clusters. Rather than pruning the generated clusters, this approach limits the non-event cluster generation. Given a count κ, this will only take the top κ keywords in the ranked tokens to use with the cluster generation step in Algorithm 3. Knowing the data and possible event generation rates, κ can be defined, and it does not strictly depend on other factors as a novelty-based threshold.

Algorithm 4: WhatsUp

input: D' : filtered data stream ($[d_0, d_1, ...d_{i-1}, d_i, ...]$), l : time window length, α : change threshold, s or m : cluster detection threshold, η or κ : cluster pruning threshold
output: E : detected co-occurred events in event windows

$E \leftarrow []$; `// empty array to keep co-occurred events in event windows`
// Data preprocessing
$W \leftarrow D'$ separated into l long time windows;
for W_t *in* W **do**
  $V_t \leftarrow$ vector space learned from W_t;
  $vocab_t \leftarrow$ vocabulary at W_t;
  $f_t \leftarrow$ token frequencies of $vocab_t$;
  $W_t \leftarrow \{V_t, vocab_t, f_t\}$;
end
// Event window identification
$W^d \leftarrow$ Algorithm 1 (W, α);
// Event cluster detection
for W_t^d *in* W^d **do**
  $E_{W_t^d} \leftarrow$ Algorithm 3 $(W_t^d, s$ or m, η or $\kappa)$;
  $E.add(E_{W_t^d})$;
end
return E

5. Experimental study

We analysed the effectiveness and efficiency of WhatsUp using recent data from two diverse domains, sports and politics (Section 5.1). Both time-based and text-based metrics were involved to conduct a comprehensive evaluation (Section 5.2). All the conducted evaluations and comparisons with state-of-the-art methods are reported in Section 5.3. Since different methods require different hyper-parameters that directly affect the performance, we recognised the best settings per method using a common strategy and used the corresponding results for comparisons. Finally, Section 5.4 includes the findings for hyper-parameter sensitivity analysis of WhatsUp, which are helpful to customise the system depending on the targeted events. We implemented WhatsUp in Python 3.7[3] and used an Ubuntu 18.04 machine which has a 2.40 GHz 16 core CPU processor with 16 GB RAM for all experiments.

5.1. Data sets

To conduct the experiments, we used *Twitter Event Data 2019* [8]. To the best of our knowledge, this is the most recent social media data set released with ground truth (GT) event details. It consists of data from two diverse domains, sports and politics. The sports data set was generated focusing on the English Premier League 19/20 match between Manchester United FC and Liverpool FC on 20 October 2019. In total, this data set has 118,700 tweets posted during the period 16:15–18:30. Among them, 99,995 (84.2%) tweets posted during the match were considered for GT preparation. In the above sections, this data set was referred to as *'football data set'*, but for simplicity, we use the name *'MUNLIV'* for future references. The political data set was generated focusing on the Brexit Super Saturday 2019, a UK parliament session that occurred on Saturday, 19 October 2019. In total, this has 276,448 tweets collected during the period 08:30–18:30. Among them, 174,498 (63.1%) tweets posted from the beginning of the parliament session until the vote were considered for GT preparation. The following sections will refer to this data set as *'BrexitVote'*. Considering the evolution rate of each domain, the original study used 2-min time windows for MUNLIV and 30-min time windows for BrexitVote. We also use the same window lengths in this research. Since high evolution rates generate more information during short periods, short time windows are appropriate for domains like sports for real-time event extraction. Contrarily, the domains with low evolution rates, like politics, take longer to generate information and long time windows are appropriate for them. On average, there are 1,724 and 14,542 tweets per 2-min and 30-min time window.

Originally, GT events were extracted from published media reports related to the chosen topics. For a published event, if at least one of its keywords was found in social media data during the time when that event was reported, it was marked as a GT event. This approach does not consider the possibility of variations between the event reporting times in two different media. However, after carefully analysing the data and events, we recognised that it is possible to have slight temporal vari-

[3] WhatsUp implementation is available on https://github.com/HHansi/WhatsUp.

ations in social media events than the events in published media reports. There were situations when either a media got delayed to report than the other. Without considering such variations, some GT events can be missed, and some can be labelled with incorrect time details. To improve the accuracy of GT labels, for each GT event, we manually analysed the temporal variations of GT keyword frequencies in social media data to extract the actual event reported time in social media. We publish the updated version of the GT along with this paper.[4]

MUNLIV data set has 57 2-min time windows during the period considered for GT preparations. According to the original GT, there were 23 time windows labelled as event windows with 27 events. After capturing the event time variations, the updated version of GT has 25 event windows with 31 events. BrexitVote data set has 11 30-min time windows during the period considered for GT preparations. In the original GT, there were 8 event windows with 19 events. A significant change happened to this data set with the updates, returning all 11 windows as event windows with 27 events.

Data Cleaning: We followed a few procedures to clean the data. We tokenised tweet text using the TweetTokenizer model available with Natural Language Toolkit (NLTK).[5] This tokeniser has the ability to tokenise emoticons (e.g.:-) and words specific to the social media (e.g. 1–0, c'mon) correctly. Also, it can remove highly repeated characters to generalise various word forms written by users (e.g. goalll, goallll → goalll). We did not preserve the case sensitivity of tokens because, in social media, people are free to use wordings with different cases without following the standard language rules. Additionally, we removed retweet notations, links and hash symbols in the text. Retweet notations and links are removed because they are uninformative. By removing hash symbols, we can treat hashtags similar to other words during word embedding learning. All these removals were automated using text pattern matching based on regular expressions.

5.2. Evaluation metrics

For the performance evaluation of the proposed method and baselines, identified events need to be compared with ground truth (GT) events considering both temporal and textual aspects. Analysing the previous research, we found methods which use either time-based metrics [14,8] or event text-based metrics [37,27,11] for automated evaluations except for a combination of both. Thus, utilising the available knowledge, we designed the following metrics to perform an overall evaluation.

5.2.1. Time-based evaluation metrics

We involved time windows in evaluating temporal event details using the following metrics. In the equations stated below, the set of all GT event windows in the data set, detected event windows and relevant event windows found in detected windows are represented by W^{GT}, W^{d} and W^{r}, respectively. A detected window is marked as a relevant event window if all the GT events of that period are found in the events detected for that window.

- **Time Window Recall:** Fraction of the number of relevant event windows detected among the total number of event windows in the data set.

$$\text{Time Window Recall} = \frac{|W^{r}|}{|W^{GT}|}$$

- **Time Window Precision:** Fraction of the number of relevant event windows detected among the total number of event windows detected.

$$\text{Time Window Precision} = \frac{|W^{r}|}{|W^{d}|}$$

- **Time Window F1:** Weighted harmonic mean of time window recall and precision.

$$\text{Time Window F1} = 2 \times \frac{precision \times recall}{precision + recall}$$

5.2.2. Text-based evaluation metrics

Detected events and their keywords are compared with GT events using the following metrics. In the equations stated below, $E_{W_t^{GT}}$ represents the GT events reported at time t (or W_t) and $E_{W_t^{d}}$ represents the events identified during the detected event window W_t^{d}. A match between a detected event and a GT event is established if at least one GT keyword is found from the identified event words.

- **Event Recall:** Fraction of the events successfully detected among the GT events happened during the time windows detected as event windows.

[4] The updated version (V2) of the data sets is available on https://github.com/HHansi/Twitter-Event-Data-2019.
[5] NLTK documentation is available at https://www.nltk.org/.

$$\text{Event Recall} = \frac{\sum_{t \in W^d} |e : e \in E_{W_t^{GT}} \text{ and } e \in E_{W_t^d}|}{\sum_{t \in W^d} |e : e \in E_{W_t^{GT}}|}$$

- **Event Relevance:** Fraction of the detected events that are related to the GT events that happened during the time windows detected as event windows among all detected events.

$$\text{Event Relevance} = \frac{\sum_{t \in W^d} |e : e \in E_{W_t^d} \text{ and } e \in E_{W_t^{GT}}|}{\sum_{t \in W^d} |e : e \in E_{W_t^d}|}$$

While matching detected events with GT events to calculate Event Recall, we did not allow the same detected event to match with multiple GT events. Otherwise, if a single event is detected, including all the keywords in the GT events, it will be marked as a prediction of all events, even though having a single large event is not helpful in the real scenario. If the same detected event is matched with more than one GT event, only the best match (match with the minimum number of missed keywords) is kept. For the optimal event assignment, we used the Hungarian algorithm [48] developed for the assignment problem.

Further, we involved a keyword-based metric in measuring the textual coverage of detected events. In the equation below, e_{W_t} is an event detected during the time window W_t, which matches with the GT event e_{GT} reported at the same time window. Each detected and GT event contains a set of words/tokens which describes it. The exact matches between GT and predicted words are considered during the evaluation.

- **Keyword Recall:** Fraction of the correctly detected keywords among the total number of keywords of the GT events that have been matched to some candidate event in the time window under consideration. Only the optimally assigned events are considered.

$$\text{Keyword Recall} = \frac{\sum_{t \in W^d} |w : w \in e_{GT} \cap e_{W_t}, e_{GT} \in E_{W_t^{GT}}, e_{W_t} \in E_{W_t^d}|}{\sum_{t \in W^d} |w : w \in e_{GT}, e_{GT} \in E_{W_t^{GT}}|}$$

5.3. System evaluation

This section summarises the conducted evaluations and obtained results. Since results are highly dependent on the hyper-parameter configurations, we followed a common strategy to identify optimal hyper-parameters during all experiments to generate unbiased comparable results. We evaluated the results of all possible hyper-parameter settings and picked the best results to report in this section and to make comparisons. While comparing the results, we gave priority to Time Window F1, followed by Event Recall, Event Relevance and Keyword Recall. The impact of different hyper-parameters and heuristics behind their selections are discussed in Section 5.4.

5.3.1. Event window identification

For event window identification, we utilised the idea of Embed2Detect [8] and proposed different strategies to improve the performance. Thus, in this section, we mainly focused on comparing the effectiveness of the proposed strategies with the original idea. For evaluations, we only used the time-based metrics because these experiments only identify the event windows. To measure time-based metrics without event clusters, we considered all words which showed any temporal textual change within an event window as event words similar to [8]. The whole group of event words is considered as a single cluster and matched it with multiple GT events following the same criteria defined for event matching. The obtained results are summarised in Table 8 and 9.

We consider the Time Window F1 to compare the results since it computes the harmonic mean between recall and precision. Overall, for MUNLIV, LDL similarity performed better than DL similarity for all strategies. Also, non-relative dendrograms performed better than relative dendrograms. Among similarity change calculations, Non-Abs. similarity change returned the lowest F1 values while Pos. similarity change performed best, agreeing with the theoretical explanations in Section 4.2. A similar pattern is also found in BrexitVote results, but they are not very distinctive due to high values. High results are returned mainly because all the time windows in this data set have events. However, newly introduced strategies (LDL similarity on non-relative dendrograms and Pos. similarity change) improved the F1 by 4.15% for MUNLIV while maintaining the 100% F1 for BrexitVote.

Moreover, we analysed the performance of vocabulary change calculation along with aggregation methods, and the obtained results are available in Table 10. According to the results, average aggregation performed best. Among vocabulary change calculations, FDVC got higher results than VC. Following the theoretical exposure described in Section 4.2 and the obtained results, we can conclude that the performance of event window identification can be improved using LDL similarity

Table 8
Evaluation results: Time Window Recall (R), Precision (P) and F1 of event window identification with different strategies for MUNLIV. T and F indicate relative and non-relative dendrograms. The best result is in bold and Embed2Detect[8] result is with ‡.

Similarity Change	Relative Dendro.	DL Similarity			LDL Similarity		
		R	P	F1	R	P	F1
Abs.	T	0.8800‡	0.5000‡	0.6377‡	0.8400	0.5385	0.6563
Abs.	F	0.8800	0.5238	0.6567	0.8800	0.5238	0.6567
Non-Abs.	T	0.0400	1.0000	0.0769	0.0800	1.0000	0.1481
Non-Abs.	F	0.5600	0.6087	0.5833	0.6800	0.6071	0.6415
Pos.	T	0.2400	0.8571	0.3750	0.5200	0.5000	0.5098
Pos.	F	0.7200	0.5806	0.6429	**0.7200**	**0.6429**	**0.6792**

Table 9
Evaluation results:Time Window Recall (R), Precision (P) and F1 of event window identification with different strategies for BrexitVote. T and F indicate relative and non-relative dendrograms. The best result is in bold and Embed2Detect[8] result is with ‡.

Similarity Change	Relative Dendro.	DL Similarity			LDL Similarity		
		R	P	F1	R	P	F1
Abs.	T	1.0000‡	1.0000‡	1.0000‡	1.0000	1.0000	1.0000
Abs.	F	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Non-Abs.	T	0.3636	1.0000	0.5333	0.0909	1.0000	0.1667
Non-Abs.	F	0.6364	1.0000	0.7778	1.0000	1.0000	1.0000
Pos.	T	0.7273	1.0000	0.8421	0.8182	1.0000	0.9000
Pos.	F	0.9091	1.0000	0.9524	**1.0000**	**1.0000**	**1.0000**

Table 10
Evaluation results: Time Window Recall (R), Precision (P) and F1 of event window identification for different aggregations. Best results are in bold.

Aggregation		MUNLIV			BrexitVote		
Method	Values	R	P	F1	R	P	F1
Avg	LDL sim., VC	0.6800	0.7391	0.7083	1.0000	1.0000	1.0000
Max	LDL sim., VC	0.6800	0.7083	0.6939	1.0000	1.0000	1.0000
Avg	LDL sim., FDVC	**0.7200**	**0.7500**	**0.7347**	**1.0000**	**1.0000**	**1.0000**
Max	LDL sim., FDVC	0.8000	0.6667	0.7273	1.0000	1.0000	1.0000

Table 11
Evaluation results: Time Window Recall (R), Precision (P) and F1 of token ranking. Best results are in bold.

Token Weight	MUNLIV			BrexitVote		
	R	P	F1	R	P	F1
PSD	0.2000	0.2083	0.2041	0.0909	0.0909	0.0909
ASD	0.4000	0.4167	0.4082	0.2727	0.2727	0.2727
FD	0.6400	0.6667	0.6531	0.3636	0.3636	0.3636
AVG-PSD-FD	0.3200	0.3333	0.3265	0.1818	0.1818	0.1818
AVG-ASD-FD	**0.6800**	**0.7083**	**0.6939**	**0.4545**	**0.4545**	**0.4545**
MAX-PSD-FD	0.1600	0.1667	0.1633	0.0909	0.0909	0.0909
MAX-ASD-FD	0.6400	0.6667	0.6531	0.3636	0.3636	0.3636

Table 12
Evaluation results of token clustering. TW stands for Time Window. Best results are in bold.

Method	TW Recall	TW Precision	TW F1	Event Recall	Event Relevance	Keyword Recall
Data set			MUNLIV			
Proposed(s)	0.7200	0.7500	0.7347	0.7742	0.2500	0.7813
Proposed(m)	0.7200	0.7500	**0.7347**	**0.7742**	**0.3304**	**0.8125**
HAC	0.5600	0.5833	0.5714	0.6129	0.2447	0.9423
Data set			BrexitVote			
Proposed(s)	1.0000	1.0000	**1.0000**	**1.0000**	**0.4444**	**0.7917**
Proposed(m)	1.0000	1.0000	1.0000	1.0000	0.1810	0.7053
HAC	1.0000	1.0000	1.0000	1.0000	0.0762	0.6316

on non-relative dendrograms, Pos. similarity change and average aggregation using FDVC. We use this combination for the following experiments.

5.3.2. Event cluster detection

The method proposed for clustering consists of two main steps: (1) token ranking and (2) cluster generation. For token ranking, we experimented with the effectiveness of different weighting mechanisms described in Section 4.3.1 and the obtained results are summarised in Table 11. The target of token ranking is to find important event keywords to use with cluster generation. Therefore to measure the effectiveness of each weighting mechanism, we selected the top n tokens from the ranked list and measured the time-based metrics. For MUNLIV, we set n to 20 and for BrexitVote, we set n to 100 for these experiments. These values are chosen based on the average vocabulary size per time window of each data set. According to the results, among similarity-based weights, ASD obtained better results than PSD. This emphasises the importance of involving all similarity changes of a token for weight calculation. FD also returned relatively high F1 values for both data sets. However, with the average aggregation (AVG) of ASD and FD, we could further improve the results involving underlying linguistics and statistics. Based on the results, we can conclude that AVG-ASD-FD is the best weighting mechanism among others. For latter experiments, we only use this mechanism.

Then we evaluated the effectiveness of the cluster generation step. Since we use the dendrogram generated using hierarchical agglomerative clustering (HAC) to measure text similarities during the clustering, we compared our approach with HAC. The obtained results for both data sets are summarised in Table 12. There are two variants of the proposed approach, which use a similarity-based threshold (s) and a count-based threshold (m). According to the results, both variants of the proposed approach performed better than HAC for both data sets. For BrexitVote, similar Time Window F1 and Event Recall values were returned by HAC, but Event Relevance and Keyword Recall values are lower than the values returned by our approach. However, all the experiments return low Event Relevance values because we only performed clustering in this stage without further pruning to extract event clusters. Considering Event Relevance and Keyword Recall, the proposed approach with m performed best for MUNLIV and s performed best for BrexitVote. In the sports domain, considering a particular match, all events have the same structure with the nearly same number of keywords per event. Therefore m-based approach is capable of performing well for the sports domain. Unlike this, in the political domain, different events in different structures can be discussed. These events can have different keyword counts, and thus, the s-based approach is more appropriate for such domain than the m-based approach.

5.3.3. Overall performance

In this section, we report the overall performance of WhatsUp, comparing it with the state-of-the-art (SOTA) methods. While selecting the SOTA methods, we mainly focused on their effectiveness, efficiency and expandability. Also, we considered different competitive areas to make the baselines stronger. No baselines were selected from the areas of online clustering and pattern dynamics due to some major limitations associated as described in Section 2. More details of the selected methods are as follows.

- **MABED** [12]: This method uses anomalous variations in mention creation frequency and their magnitudes to detect events in an offline manner. Mainly, the mention anomalies are used to incorporate the social aspect of Twitter into event detection. To extract event text, this uses word co-occurrences and their temporal dynamics.
- **SEDTWik** [11]: This method is an extension to Twevent [10] and targets identifying the text of events in a given corpus with their newsworthiness. It clusters the bursty segments in data extracted using Wikipedia titles, applying the Jarvis-Patrick algorithm to identify events. For our experiments, we applied this method to each time window and only considered the events with high newsworthiness than a defined threshold. If any window contained such event(s), we marked it as an event window.
- **LDA-based Sub-Event Tracking (LDA-SET)** [35]: This method identifies topics per window using Latent Dirichlet Allocation (LDA) and then analyses their temporal evolution to capture event transitions. The temporal evolution is mainly assessed using topic similarity (common keyword count-based measure) and topic word count. Among the captured transitions, we only considered *form* events (or new topics) for the experiments because we only evaluate such events in this research.

The results obtained for MUNLIV and BrexitVote data sets using WhatsUp and SOTA methods are summarised in Table 13 and 14. For WhatsUp, we experimented with both similarity-based (s) and count-based (m) clustering approaches as they showed different qualities useful for different domains and users. For event cluster extraction, also we applied pruning based on novelty (η) and keyword count (κ) thresholds to evaluate their effectiveness. Combining both clustering and cluster pruning techniques, in total, there are four combinations for WhatsUp, as shown in the results tables. The bellow section of each table lists the experimented SOTA methods and obtained results.

According to the results, for MUNLIV, all four combinations of WhatsUp outperformed the SOTA methods returning 14.93% higher Time Window F1 than the best SOTA method. Similarly, for BrexitVote, all combinations of WhatsUp obtained 9.09% higher Time Window F1 than the best SOTA method, proving that our approach can effectively extract temporal event information in diverse domains than the available methods. We cannot involve event-based and keyword-based metrics for comparisons between methods because they are calculated based on the event windows identified by each method. Differ

Table 13
Evaluation results of event detection for MUNLIV. TW stands for Time Window and Average Time indicates the processing time for a 2-min time window. The best result is in bold and the best SOTA method result is in italics.

Method	TW Recall	TW Precision	TW F1	Event Recall	Event Relevance	Keyword Recall	Average Time(s)
WhatsUp(s,κ)	0.7200	0.7500	**0.7347**	**0.7742**	**0.4400**	**0.7813**	3.6140
WhatsUp(m,κ)	0.7200	0.7500	0.7347	0.7742	0.4231	0.8125	3.7193
WhatsUp(s,η)	0.7200	0.7500	0.7347	0.7742	0.2836	0.7619	3.8596
WhatsUp(m,η)	0.7200	0.7500	0.7347	0.7742	0.3743	0.8125	3.7368
MABED	0.5200	0.3095	0.3881	0.5161	0.2867	0.6444	3.9825
SEDTWiK	0.4800	0.2308	0.3117	0.5161	0.3148	0.4889	11.5263
LDA-SET	0.9600	0.4211	*0.5854*	*0.9677*	*0.2391*	*0.8442*	3.7807

Table 14
Evaluation results of event detection for BrexitVote. TW stands for Time Window and Average Time indicates the processing time for a 30-min time window. The best result is in bold and the best SOTA method result is in italics.

Method	TW Recall	TW Precision	TW F1	Event Recall	Event Relevance	Keyword Recall	Average Time(s)
WhatsUp(s,κ)	1.0000	1.0000	**1.0000**	**1.0000**	**0.6585**	**0.7396**	32.1818
WhatsUp(m,κ)	1.0000	1.0000	1.0000	1.0000	0.2463	0.5464	35.4545
WhatsUp(s,η)	1.0000	1.0000	1.0000	1.0000	0.5152	0.7917	32.4545
WhatsUp(m,η)	1.0000	1.0000	1.0000	1.0000	0.2671	0.6489	33.6364
MABED	0.9091	0.9091	*0.9091*	*0.8889*	*0.6067*	*0.5682*	51.2727
SEDTWiK	0.7273	0.7273	0.7273	0.7407	0.7500	0.3784	42.7273
LDA-SET	0.8182	0.8182	0.8182	0.8889	0.5962	0.4471	17.5000

the identified event windows, these computations happen on different bases (Section 5.2.2). Using WhatsUp, for both data sets, we received the same Event Recall for all combinations. However, we noticed variations in Event Relevance and Keyword Recall depending on the used combination. Overall, κ-based pruning performed better than η-based pruning in both data sets. Following the insights in Section 5.3.2, for MUNLIV, both s- and m-based clustering performed well with κ, but for BrexitVote, only s-based clustering performed well in terms of Event Relevance and Keyword Recall. In summary, these experiments reveal that WhatsUp with (s,κ) combination performs effective event detection in diverse domains, specifically, sports and politics than other combinations and SOTA methods. Further, we report qualitative analysis of sample events in A focusing on word coverage and novelty measure, which further emphasise the effectiveness of event details detected by WhatsUp.

Additionally, the processing time is also a critical factor in performing event detection in (near) real-time. Targeting this requirement, we involved efficient computation techniques for our approach and also incorporated parallel processing for all time-consuming operations. To measure the reported processing times in Table 13 and 14, we used parallel processing with 8 workers. According to the measured times, on average, WhatsUp took 3.72 s to process a 2-min time window ($\approx 1,724$ tweets) and 33.43 s to process a 30-min time window ($\approx 14,542$ tweets). Compared to the majority of the SOTA methods, our approach performed faster in both data sets. Considering the specifications of the used machine and measured processing times, we can state that our approach is sufficiently fast for (near) real-time detection. Also, parallel processing provides the capability to handle increasing data amounts successfully. Further, in B, we report a detailed analysis conducted on intermediate processing times comparing the sequential and parallel processing.

Hyper-parameters: For all methods, we used the optimal hyper-parameter settings to generate comparable results. Table 15 summarises the optimal hyper-parameter values obtained. For WhatsUp, we optimised the change threshold (α) and frequency threshold (β), considering all possible values to identify event windows. To detect event clusters, four parameter combinations of cluster word similarity (s) or cluster word count (m) and novelty (η) or keyword count (κ) were optimised following all possibilities. For m, we defined a maximum value of 25 for MUNLIV and 35 for BrexitVote, considering the GT event word counts. For MABED, we optimised the number of events (k), maximum number of words describing each event (p), weight threshold for selecting relevant words (θ) and overlap threshold (σ). For k and p, incremental values have been experimented with until we receive the optimal result. For θ and σ, we experimented with the values around the original values used in the initial experiments [12]. For SEDTWik, we optimised the number of subwindows (M), number of cluster neighbours (k) and newsworthiness threshold (τ). Values for M were picked based on the time window length of each data set. For k and τ, incremental values were experimented with until receiving the optimal result. For LDA-SET, we optimised the number of topics (k), minimum probability to filter topic words and threshold match, which compares the events temporally. We experimented with incremental values for all of them until we received the optimal result.

Table 15
Best hyper-parameter settings.

Method	Hyper-parameters	
	MUNLIV	**BrexitVote**
WhatsUp	$\alpha = 0.12$ $\beta = 20$ $(s, \kappa) = (0.4, 15)$ $(m, \kappa) = (25, 15)$ $(s, \eta) = (0.4, 0.1)$ $(m, \eta) = (25, 0.12)$	$\alpha = 0.12$ $\beta = 10$ $(s, \kappa) = (0.6, 130)$ $(m, \kappa) = (35, 50)$ $(s, \eta) = (0.6, 0.08)$ $(m, \eta) = (35, 0.2)$
MABED	$k = 150$ $p = 25$ $\theta = 0.7$ $\sigma = 0.5$	$k = 150$ $p = 35$ $\theta = 0.6$ $\sigma = 0.5$
SEDTWik	$M = 2$ $k = 5$ $\tau = 0.6$	$M = 3$ $k = 3$ $\tau = 0.3$
LDA-SET	$k = 15$ *min. probability*= 0.01 *threshold match*= 0.7	$k = 30$ *min. probability*= 0.01 *threshold match*= 0.6

5.4. Parameter sensitivity analysis

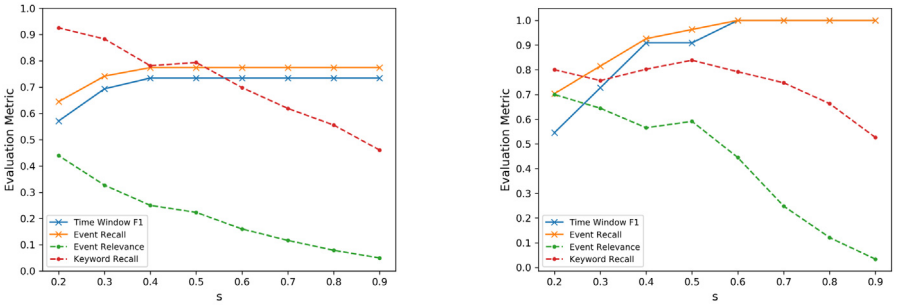
WhatsUp requires hyper-parameters during word embedding learning, event window identification and event cluster detection. They should be mainly picked depending on the characteristics of the targeted domain and user preferences on resulting events. Different parameters make different levels of impact on the output, as described below.

Word embedding learning: To train Word2Vec models during embedding learning, three hyper-parameters: minimum word count, context size, and vector dimension are required. The minimum word count facilitates the removal of tokens with less total frequency than the count. The context size defines the number of words around the target word to consider during learning. The vector dimension represents the number of dimensions in the final word embeddings. We fixed the minimum word count to 1, considering the limited data availability per window. The other parameters generally improve the text similarity accuracy, higher their values [42]. However, such effect was not notably seen with event detection due to its comparatively low sensitivity to the underlying linguistics [8]. Yet, the processing time increase proportional to these parameters. Following these facts, we fixed the context size to 5 and vector dimensions to 100 to provide sufficient knowledge for efficient embedding learning.

Event window identification: To identify event windows, two hyper-parameters: change threshold (α) and frequency threshold (β) are required. α mainly indicates the significance level of targeted events, and β is used as a frequency threshold to remove outlier tokens. Typically, more fluctuations in overall change are expected in highly evolving domains like sports, and such fluctuations will be reduced along with the reduction of evolution rate, varying the sensitivity of α [8]. Also, event significance is a personal preference. Thus, α should be picked based on the domain and personal event interests. The behaviour of a few past time windows can be used to support this decision. Similarly, β is also a domain-dependent parameter since the inclusion and magnitude of outlier tokens vary with the domain. It is important to ensure that β is only sufficient to remove outliers, as an unnecessarily large β can also remove event tokens as well [8].

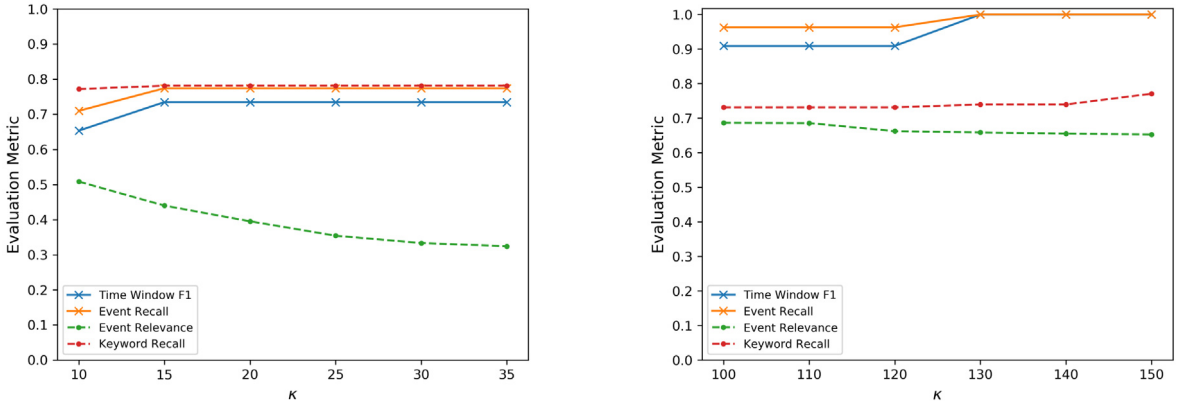
Event cluster detection: To generate clusters, we proposed to use either a similarity-based threshold (s) or a word count-based threshold (m). According to the overall performance, s-based clustering performed best for both data sets. Any value between 0–1 can be picked for s as it considers the LDL similarity between tokens. To provide a clearer idea, we plotted the variations of evaluation metrics of both data sets with varying s in Fig. 5. According to the obtained results, for both data sets, Time Window F1 and Event Recall increase with increasing s until they get their maxima. $s \approx 0$ generates huge clusters since it also groups tokens with less similarity, failing to identify event clusters separately. Thus, optimal results are obtained when s is sufficiently large for the cluster separation. Contrarily, Event Relevance and Keyword Recall decrease with increasing s. When $s \approx 1$, very small clusters will be generated due to the consideration of very similar tokens. Thus, unnecessary separation of event clusters occurs, reducing the keyword coverage per cluster. Therefore it is important to pick a middle value for s. For MUNLIV, a value between 0.4–0.6 and for BrexitVote, a value between 0.6–0.8 is appropriate. Following these insights, we can summarise that if the targeted domain is highly evolving and short time windows with fewer data amounts are considered, a comparatively low value will be the optimal s. A relatively high value will be optimum for a domain with opposite characteristics. As described in Section 4.3, if the targeted domain has events with a similar structure, m-based clustering can be used, and the average number of keywords per event can be set as m straightforwardly.

After generating the clusters, they need to be pruned to extract important events. We followed two approaches using novelty (η) and keyword count (κ) for pruning, and, among them, κ-based pruning performed best in both domains. As κ, the possible keyword count per time window should be set. Relatively, MUNLIV prefers a low value than BrexitVote because

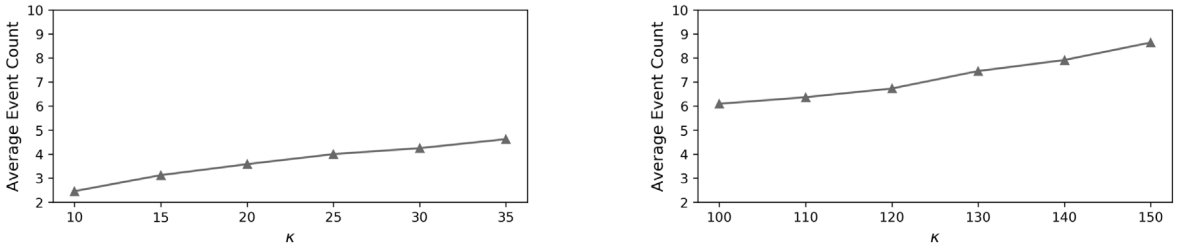


(a) Evaluation results- MUNLIV (b) Evaluation results- BrexitVote

Fig. 5. Evaluation results for varying similarity threshold (s) values.



(a) Evaluation results-MUNLIV($s = 0.4$) (b) Evaluation results-BrexitVote($s = 0.6$)



(c) Average event counts- MUNLIV (d) Average event counts- BrexitVote

Fig. 6. Evaluation results and average event counts per event windows for varying keyword count (κ) values.

MUNLIV has shorter time windows with fewer possible events. To provide a clearer insight, we plotted evaluation results and resulted in cluster counts with varying κ values in Fig. 6. As can be seen, for both data sets, Time Window F1 and Event Recall achieved their optimal state with sufficiently large κ to capture all important events. Lower the κ; important events can be missed. However, Keyword Recall shows a near-constant behaviour, and Event Relevance shows a slight drop with increasing κ. Since more events detect increasing the κ (Fig. 6c, 6d), it is possible to capture non-relevant events too. Thus, when picking a value for κ, the focus should be on the lowest possible keyword count, which is sufficiently large to capture relevant events.

6. Conclusions and future work

In this paper, we proposed a novel event resolution method named *WhatsUp* for co-occurring event detection in social media. Our approach focuses on the automatic detection of both temporal and fine-grained textual (text of co-occurred

events) event details involving both statistical and linguistical features of underlying data. To capture statistics, token frequency-based measures were used and to capture linguistics, self-learned word embeddings and their hierarchical relationships in dendrograms were used. In summary, WhatsUp considers all the important features in textual data needed for effective event detection. Also, to the best of our knowledge, no prior work involved both statistics and linguistics in detecting both temporal and textual event details together. Further, the usage of unsupervised techniques makes our approach expandable to any language, platform and domain. The involvement of self-learned word embeddings supports expandability as well as understanding data-specific linguistics.

WhatsUp returned promising results on conducted experiments. We used several state-of-the-art methods from competitive areas to make stronger baselines for comparisons. Also, we designed metrics covering both temporal and textual event aspects to conduct comprehensive evaluations and used data from two diverse domains to assess the methods' universality. Overall, WhatsUp outperformed all the baselines regarding event time and text detection. Following these results, it is safe to assume that our approach has low information loss than available approaches, which happens due to insufficient involvement of linguistics. Considering the processing time, WhatsUp took comparatively low time for both data sets, proving its efficiency appropriate for (near) real-time processing.

In future work, we plan to integrate a text summarisation technique to return a summary of each event cluster. When a similarity-based threshold is used for clustering, it is possible to grow clusters massively in some instances. A summary will be helpful in such situations to provide the user with a quick insight into the event. Also, we plan to extend our approach to detect the evolution of events over time to understand the event progress and analyse the impact of time window length for event resolution. Further, it is useful to mine sentiments of events to enhance their informativeness because knowing public opinion would be helpful in situations such as crises, political debates and product launches to take necessary immediate actions. Also, there is a tendency to use multimedia data such as images and videos to report events on social media in addition to text [49,50]. Thus, it is worth investigating how WhatsUp can be extended for multi-model event detection in future.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT authorship contribution statement

Hansi Hettiarachchi: Conceptualization, Methodology, Software, Validation, Data curation, Writing – original draft, Writing – review & editing. **Mariam Adedoyin-Olowe:** Writing – review & editing, Supervision. **Jagdev Bhogal:** Writing – review & editing, Supervision. **Mohamed Medhat Gaber:** Conceptualization, Writing – review & editing, Supervision.

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.

Appendix A. Event clusters

We further analysed the quality of events detected by WhatsUp, considering the word coverage and novelty measures. Table A.16 summarises a few event windows with detected events that match with GT events. Due to the space limitations, we only report the events in the MUNLIV data set, considering their shorter length than the BrexitVote events. For MUNLIV,

Table A.16
Sample events detected by WhatsUp for MUNLIV. TW, Nov. and R stand for time window, novelty and rank, respectively. Rank indicates the event position within the corresponding time window based on the descending order of event novelties.

TW	GT Event	Detected Event	Nov.	R
16:52–16:54	Marcus Rashford is one-on-one with Virgil van Dijk on the right touchline and gets fortunate as the LFC defender slips.	bullied, see, mufc, love, marcus, rashford, 21, dijk, puts, van, floor, mulive, promising, start, old, trafford, chances, far, rt, counter-attack, either, followers, side, 😂, applying	0.4013	1
16:56–16:58	Scott McTominay (MUFC) right footed shot from outside the box is saved in the centre of the goal by Alisson.	hosts, mctominay, 0–0, distance, effort, alisson, fires, goalwards, saves, afternoon, old, trafford, 😡 fixture, like, smells, 👏, love, marcus, rashford, see, van, man, mufc, reds	0.3024	1
17:06-17:08	Goal by Marcus Rashford (MUFC)! Manchester United 1, Liverpool 0.	goooaaalll, @marcusrashford, munliv, mufc, man, marcus, rashford, 1–0, 36, goal, utd, 😱, gives, lead, 0, 1, liverpool, manchester, united, ⚪, ⚽, 🔴, 0–0, buuut	0.4430	1
	VAR review is in progress.	var, f**king, great, love, see, atkinson, back, game, win, going, martin, origi, daniel, dijk, range, van, today, far, players, @manutd, come, front, f**k, get, ggmu	0.2908	6
17:14–17:16	Sadio Mané (LFC) scores but the goal is ruled out after a VAR review.	44, disallows, let, mulive, var, equalises, mane, ruled, sadio, fans, loving, man, utd, goal, handball, overturned, disallowed, mane's, 1–0, golazo, hosts, remains, every, f**k, get	0.3914	1

both s- and m-based clustering performed well with κ (Section 5.3.3). Among them, we used (m, κ) combination for the predictions because m fixes the number of words in a cluster easing the analysis and reporting.

As can be seen in Table A.16, detected events effectively cover the majority of keywords in GT events. Additionally, event-related misspelt words, other/background words, and emojis are also recognised by WhatsUp, highlighting the potential of the involved linguistical features. Further, the novelty measure provides a positive insight into the newness of the event. However, it is less effective to compare novelty values across different event windows because they highly depend on the underlying data. In the given samples, for top-ranked events within the windows, novelty varied from 0.30 to 0.44. Similar nature of word coverage and novelty measure was also found in BrexitVote event details, except for the differences in cluster size and less emoji usage, which should be expected with the complexities and audience in the political domain.

Appendix B. Intermediate processing time

Additionally, we did an intermediate analysis to understand the complexity of each component. The obtained results are summarised in Table B.17 and B.18. Total time indicates the processing time taken by the whole corpus (57 2-min windows of MUNLIV and 11 30-min windows of BrexitVote), and average time indicates the time taken by a single time window. We only integrate parallel processing for time-consuming operations (i.e. word embedding learning and event window identifying) to faster the process to support (near) real-time processing.

Table B.17
WhatsUp(s, κ) intermediate processing time – MUNLIV.

Step	Time(s): 1 worker		Time(s): 8 workers	
	Total	Average	Total	Average
Stream Chunker	66	1.1579	69	1.2105
Word Embedding Learner	114	2.0000	87	1.5263
Statistical Information Extractor	1	0.0175	1	0.0175
Event Window Identifier	33	0.5789	31	0.5439
Event Cluster Detector	8	0.1404	8	0.1404
Full Process	235	4.1228	206	3.6140

Table B.18
WhatsUp (s, κ) intermediate processing time – BrexitVote.

Step	Time(s): 1 worker		Time(s): 8 workers	
	Total	Average	Total	Average
Stream Chunker	31	2.8182	32	2.9091
Word Embedding Learner	192	17.4545	73	6.6364
Statistical Information Extractor	3	0.2727	3	0.2727
Event Window Identifier	573	52.0909	152	13.8182
Event Cluster Detector	72	6.5455	65	5.9091
Full Process	899	81.7273	354	32.1818

References

[1] D. Chaffey, Global social media statistics research summary 2021 — smart insights, https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/, 2021.
[2] C. Castillo, M. Mendoza, B. Poblete, Information credibility on twitter, in: Proceedings of the 20th International Conference on World Wide Web (WWW), WWW '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 675–684. doi: 10.1145/1963405.1963500.
[3] Ç.Ç. Karaman, S. Yalıman, S.A. Oto, Event detection from social media: 5w1h analysis on big data, in: 2017 25th Signal Processing and Communications Applications Conference (SIU), 2017, pp. 1–4, https://doi.org/10.1109/SIU.2017.7960211.
[4] H. Kwak, C. Lee, H. Park, S. Moon, What is twitter, a social network or a news media?, in: Proceedings of the 19th International Conference on World Wide Web (WWW), WWW '10, Association for Computing Machinery, New York, NY, USA, 2010, pp 591–600, 10.1145/1772690.1772751. DOI: 10.1145/1772690.1772751.
[5] T.G. van der Meer, P. Verhoeven, Public framing organizational crisis situations: Social media versus news media, Public Relations Rev. 39 (2013) 229–231.
[6] J.A. Gottfried, E. Shearer, News use across social media platforms 2017, https://www.journalism.org/2017/09/07/news-use-across-social-media-platforms-2017/, 2017.
[7] D.T. Hoang, V.C. Tran, V.D. Nguyen, N.T. Nguyen, D. Hwang, Improving academic event recommendation using research similarity and interaction strength between authors, Cybern. Syst. 48 (2017) 210–230.
[8] H. Hettiarachchi, M. Adedoyin-Olowe, J. Bhogal, M.M. Gaber, Embed2Detect: Temporally clustered embedded words for event detection in social media, Mach. Learn. 111 (2022) 49–87.
[9] S.G. Small, L. Medsker, Review of information extraction technologies and applications, Neural Comput. Appl. 25 (2014) 533–548.
[10] C. Li, A. Sun, A. Datta, Twevent: Segment-based event detection from tweets, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM), CIKM '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 155–164. doi: 10.1145/2396761.2396785.

[11] K. Morabia, N.L. Bhanu Murthy, A. Malapati, S. Samant, SEDTWik: Segmentation-based event detection from tweets using Wikipedia, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Student Research Workshop, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 77–85. 10.18653/v1/N19-3011.
[12] A. Guille, C. Favre, Event detection, tracking, and visualization in twitter: a mention-anomaly-based approach, Soc. Network Anal. Min. 5 (2015) 18.
[13] C. Comito, A. Forestiero, C. Pizzuti, Bursty event detection in twitter streams, ACM Trans. Knowl. Discov. Data 13 (2019).
[14] M. Adedoyin-Olowe, M.M. Gaber, C.M. Dancausa, F. Stahl, J.B. Gomes, A rule dynamics approach to event detection in twitter with its application to sports and politics, Expert Syst. Appl. 55 (2016) 351–360.
[15] Q. Li, A. Nourbakhsh, S. Shah, X. Liu, Real-time novel event detection from social media, in: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), 2017, pp. 1129–1139, https://doi.org/10.1109/ICDE.2017.157.
[16] S. Nguyen, B. Ngo, C. Vo, T. Cao, Hot topic detection on twitter data streams with incremental clustering using named entities and central centroids, 2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF) (2019) 1–6, https://doi.org/10.1109/RIVF.2019.8713730.
[17] A.M. Ertugrul, B. Velioglu, P. Karagoz, Word embedding based event detection on social media, in: F.J. Martínez de Pisón, R. Urraca, H. Quintián, E. Corchado (Eds.), Hybrid Artificial Intelligent Systems, Springer International Publishing, Cham, 2017, pp. 3–14.
[18] C. Comito, A. Forestiero, C. Pizzuti, Word embedding based clustering to detect topics in social media, in: IEEE/WIC/ ACM International Conference on Web Intelligence (WI), WI '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 192–199. doi: 10.1145/3350546.3352518.
[19] Z. Saeed, R.A. Abbasi, O. Maqbool, A. Sadaf, I. Razzak, A. Daud, N.R. Aljohani, G. Xu, What's happening around the world? a survey and framework on event detection techniques on twitter, J. Grid Comput. 17 (2019) 279–312.
[20] Y. Yang, T. Pierce, J. Carbonell, A study of retrospective and on-line event detection, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98, Association for Computing Machinery, New York, NY, USA, 1998, pp. 28–36. doi: 10.1145/290941.290953.
[21] T. Sutanto, R. Nayak, Fine-grained document clustering via ranking and its application to social media analytics, Soc. Network Anal. Min. 8 (2018) 1–19.
[22] M. Hasan, M.A. Orgun, R. Schwitter, Real-time event detection from the twitter data stream using the TwitterNews+ framework, Inf. Process. Manage. 56 (2019) 1146–1165.
[23] J. Chen, Z. Gong, W. Liu, A nonparametric model for online topic discovery with word embeddings, Inf. Sci. 504 (2019) 32–47.
[24] D. Corney, C. Martin, A. Göker, Spot the ball: Detecting sports events on twitter, in: M. de Rijke, T. Kenter, A.P. de Vries, C. Zhai, F. de Jong, K. Radinsky, K. Hofmann (Eds.), Advances in Information Retrieval, Springer International Publishing, Cham, 2014, pp. 449–454.
[25] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, K. Wang, TopicSketch: Real-time bursty topic detection from twitter, IEEE Trans. Knowl. Data Eng. 28 (2016) 2216–2229.
[26] M. Peng, S. Ouyang, J. Zhu, J. Huang, H. Wang, J. Yong, Emerging topic detection from microblog streams based on emerging pattern mining, in: 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2018, pp. 259–264, https://doi.org/10.1109/CSCWD.2018.8465166.
[27] H.-J. Choi, C.H. Park, Emerging topic detection in twitter stream based on high utility pattern mining, Expert Syst. Appl. 115 (2019) 27–36.
[28] H. Sayyadi, M. Hurst, A. Maykov, Event detection and tracking in social streams, in: 3rd International AAAI Conference on Weblogs and Social Media (ICWSM), 2009, pp. 311–314.
[29] M. Takaffoli, F. Sangi, J. Fagnan, O.R. Zaïane, MODEC – modeling and detecting evolutions of communities, in: L.A. Adamic, R. Baeza-Yates, S. Counts (Eds.), 5th International AAAI Conference on Weblogs and Social Media (ICWSM), The AAAI Press, 2011, pp. 626–629. http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2853.
[30] L. Mu, P. Jin, L. Zheng, E.-H. Chen, L. Yue, Lifecycle-based event detection from microblogs, in: Companion Proceedings of the The Web Conference 2018, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, p. 283–290. doi: 10.1145/3184558.3186338.
[31] X. Liu, J. Fu, Y. Chen, Event evolution model for cybersecurity event mining in tweet streams, Inf. Sci. 524 (2020) 254–276.
[32] J.H. Lau, N. Collier, T. Baldwin, On-line trend analysis with topic models: #twitter trends detection topic model online, in: Proceedings of the 24th International Conference on Computational Linguistics (COLING), The COLING 2012 Organizing Committee Mumbai, India, 2012, pp. 1519–1534.
[33] X. Chen, X. Zhou, T. Sellis, X. Li, Social event detection with retweeting behavior correlation, Expert Syst. Appl. 114 (2018) 516–523.
[34] J. Huang, M. Peng, H. Wang, J. Cao, W. Gao, X. Zhang, A probabilistic method for emerging topic tracking in microblog stream, World Wide Web 20 (2017) 325–350.
[35] S. Unankard, W. Nadee, Sub-events tracking from social network based on the relationships between topics, in: 2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT NCON), 2020, pp. 1–6. DOI: 10.1109/ECTIDAMTNCON48261.2020.9090732.
[36] R. McCreadie, C. Macdonald, I. Ounis, M. Osborne, S. Petrovic, Scalable distributed event detection for twitter, IEEE International Conference on Big Data 2013 (2013) 543–549, https://doi.org/10.1109/BigData.2013.6691620.
[37] L.M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Göker, I. Kompatsiaris, A. Jaimes, Sensing trending topics in twitter, IEEE Trans. Multimedia 15 (2013) 1268–1282.
[38] P.S.M. Tsai, Mining frequent itemsets in data streams using the weighted sliding window model, Expert Syst. Appl. 36 (2009) 11617–11625.
[39] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. https://aclanthology.org/N19-1423. 10.18653/v1/N19-1423.
[40] T. Ranasinghe, C. Orasan, R. Mitkov, TransQuest: Translation quality estimation with cross-lingual transformers, in: Proceedings of the 28th International Conference on Computational Linguistics (COLING), International Committee on Computational Linguistics, Barcelona, Spain (Online), 2020, pp. 5070–5081. https://aclanthology.org/2020.coling-main.445. 10.18653/v1/2020.coling-main.445.
[41] H. Hettiarachchi, M. Adedoyin-Olowe, J. Bhogal, M.M. Gaber, DAAI at CASE 2021 task 1: Transformer-based multilingual socio-political and crisis event detection, in: Proceedings of the 4th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2021), Association for Computational Linguistics, Online, 2021, pp. 120–130. https://aclanthology.org/2021.case-1.16. 10.18653/v1/2021.case-1.16.
[42] 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, 2013a. http://arxiv.org/abs/1301.3781.
[43] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS) – Volume 2, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013b, p. 3111–3119.
[44] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, USA, 2008.
[45] D. Müllner, Modern hierarchical, agglomerative clustering algorithms, CoRR abs/1109.2378 (2011).
[46] M. Antoniak, D. Mimno, Evaluating the stability of embedding-based word similarities, Trans. Assoc. Comput. Linguist. 6 (2018) 107–119.
[47] A.M.J. Schakel, B.J. Wilson, Measuring word significance using distributed representations of words, CoRR abs/1508.02297 (2015).
[48] J. Munkres, Algorithms for the assignment and transportation problems, J. Soc. Ind. Appl. Math. 5 (1957) 32–38.
[49] S. Zhao, Y. Gao, G. Ding, T.-S. Chua, Real-time multimedia social event detection in microblog, IEEE Trans. Cybern. 48 (2018) 3218–3231.
[50] H. Zhou, H. Yin, H. Zheng, Y. Li, A survey on multi-modal social event detection, Knowl.-Based Syst. 195 (2020) 105695.