Interaction Graph, Topical Communities, and Efficient Local Event Detection from Social Streams

Shubham Gupta[1], Suman Kundu[1]

[a]*Department of Computer and Science Engineering, Indian Institute of Technology, Jodhpur, 342001, Rajasthan, India*

Abstract

Social networks have become an essential part of daily life, and hence every real-world activity finds its place in this virtual world. The present paper proposes a methodology to find localized micro-events from the social network stream. The method is named CommunityINDICATOR. A concept of 'separation of concerns' from the software design principle is incorporated in the methodology to reduce the execution time drastically from existing state-of-the-art methods of event detection. In order to reduce the execution time, the algorithm first generates an interaction graph from the social stream and applies community detection followed by a clustering algorithm onto it to detect micro-level events. Experiments have been conducted on Twitter data stream of 5 different cities on three different continents with the size of 2 million tweets. We have used well known quality metrics such as precision, recall, F1-score, accuracy, and execution time to compare performance with other state-of-the-art methodologies. The proposed CommunityINDICATOR provides up to 30% higher accuracy than EvenTweet and SEDTWik in similar execution times. An improvement of 11% to 51% and 17% to 57% in execution time is observed for the proposed algorithm in comparison with TwiiterNews and EventX, respectively, for different datasets.

Keywords: event detection, event extraction, data stream mining, unsupervised learning

1. Introduction

Social network platforms such as Facebook, Twitter etc., become an essential part of modern society. Any real-world activities or events, such as concerts, gatherings, conferences, etc. are now had their virtual presence in the Internet. Sometimes their presence is observed even before the occurrence of the actual event or mainstream media publications, for examples hashtag #FIFAWorldCupQatar2022 was available on the social network from starting of the year 2022, while the actual event occurred in December 2022. Similarly, in case of emergency, hashtag or information related to the event appears in social media immediately, e.g., *"Four people shot outside of Nationals Park. #TerroristAttack #Emergency"*, *"Taliban is attacking on Afghanistan #TalibanAttack #PleaseSave"*, etc. This observation leads to a question: Can we detect or find any real-world events, crises, and incidents from the social streams?

According to (McMinn et al., 2013), an event is a significant thing that occurs at some specific place and time, for example, an award function in a city, violent attacks in local areas, a strike against some local issue etc. Events can be classified as harmful (e.g., fake news, spammer, riots, mugging, etc.), neutral (e.g., football matches, religious activities, etc.), or positive (e.g., awareness campaign, polio drop, COVID vaccination). Event detection in social network is to find different events from it (Garg and Kumar, 2016). Detection of different events beforehand or real time from social network has several benefits in safety & security (Roldán et al., 2020), environment monitoring (Kerman et al., 2008), and health monitoring (Luo et al., 2022). It may also help local municipal or law enforcement to take appropriate action when required.

Event detection methods, having no prior knowledge of the target event are called open domain event detection (Deng et al., 2015; Atefeh and Khreich, 2015). Many methods (Li et al., 2012; Morabia et al., 2019; Chen et al., 2012; Fedoryszak et al., 2019; Hasan et al., 2016b; Walther and Kaisser, 2013; Li et al., 2020) for open domain event detection have been proposed in last decades. In Twevent (Li et al., 2012) and SEDTWik (Morabia et al., 2019), events were identified from Tweets by segmentation of text. On the other hand, Chen et al. (2012) finds an event based on the community structure of the graph. This approach considers all maximal cliques in the graph as a community. Fedoryszak et al. (2019) proposed a method based on a similarity graph for finding the weighted bipartite matching, which helps to identify events in a real-time environment. In (Hasan et al., 2016b), bucketing of similar tweets based on hashing is used for event detection. Many times it is important to identify events that are from a particular geolocation, and one such work is proposed by Walther and Kaisser (2013) which uses machine learning. Events are also identified based on the geographical and semantic similarity of tweet (Abdelhaq et al., 2013; Zhang et al., 2016, 2018). They create the clusters based on those that have the same geographical and semantic similarity. The aforementioned methods have certain limitations e.g. segmentation-based methods require time for similarity calculations, clique-based event detection methods fail to recognise

URL: gupta.37@iitj.ac.in (Shubham Gupta), suman@iitj.ac.in (Suman Kundu)

loosely connected groups in graphs, and the remaining algorithms take time to cluster large data. Further, these methods report events only on a macroscopic level, except for a few.

In this paper, we proposed an event detection methodology, CommunityINDICATOR, where we identify community structure from the interaction graph of the social stream before running the unsupervised machine learning. Identifying communities beforehand reduces the execution time many folds as compared to state-of-the-art methodologies. First, the algorithm constructs interaction graphs of social communications such as like, comment, news feed, user mentions, hashtags, etc., from the social data streams of a fixed time duration. Second, a community detection algorithm is then used to identify different topical communities in each interaction graph. From each individual topical community, we have identified micro clusters that can be considered as a good candidate of local event. Third, the micro clusters found from topical communities of subsequent interaction graphs are compared based on the similarity with the next time slice cluster. Latent Dirichlet Allocation (LDA) (Blei et al., 2003) topic modelling is used on micro clusters to extract the context which helps to find out similarity. Fourth, we build a chain of clusters based on these similar clusters, called as topical chain. At last, events are then identified based on the size of the topical chain. Figure 1 illustrates the method of identification of topical chains. Extensive experiments are conducted on Twitter for five different cities of three different continents. These are Delhi, Mumbai, and Bangalore in India, London in United Kingdom, and New York in United State of America. The proposed algorithm identifies 10% more events than the nearest state-of-the-art method for all the cities we have experimented with. In addition, CommunityINDICATOR takes less time for majority of the experiments and in some cases it takes more than 40% less time than the nearest comparing method. Experimentally, the proposed method is capable of identifying very localized events such as local satsang (religious activities), university exam and violent attacks in local areas. In particular, we sum up contributions of this study as follows:

- This research proposes an end-to-end event detection methodology CommunityINDICATOR, which identify the micro level localized events from the social streams.
- The concept of 'separation of concerns' is introduced to improve the performance which provides massive execution time benefit over exiting event detection methodologies in the existing unsupervised machine learning pipeline. The 'separation of concerns' introduced via topical community detection over interaction graph.
- Effectiveness of proposed methodology is demonstrated through extensive experiments on extracted Twitter dataset. The proposed methodology outperforms other state-of-the-art methodologies in the terms of precision, F1-score and accuracy.

The paper is organized as follows: Section 2 reports the related work. Section 3 defines the problem statement formally. Sections 4 and 5 describe our proposed algorithm and experimental results. Finally, Section 6 and 7 concludes the research outcome with future work.

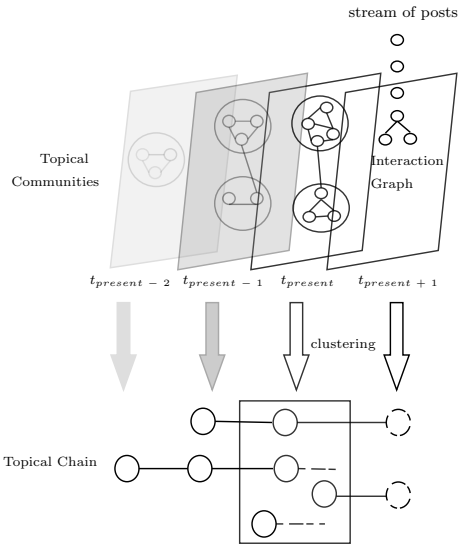

Figure 1: An illustration of how our proposed methodology works in streaming data. The diagram shows the formation of topical chains from the topical micro clusters of the interaction graph of time slice $t_{present}$. The shade of the color in the process indicates the removal of components from the memory space.

2. Related Work

Event detection was first explored by Allan (2002) in his book on topic detection and tracking. His work provided methods that later became the basis of several event detection problems. Here we will present a literature review of open domain event detection under the category of global and local event detection.

Global event detection targets those events that are having impact on global audience and are more bursty in the text streams. Allan et al. (1998) proposed one single-pass clustering approach where a stream is grouped based on similarity threshold. Similarity threshold is used to determine whether a new document will form a new topic or merge into an existing one. Sankaranarayanan et al. (2009) used the TF-IDF vector to find out the similarity to cluster the stream and used the Naive Bayes filter to identify the newsworthy events. A method for continuous clustering of the tweet stream for detecting events is proposed in (Aggarwal and Subbian, 2012), and the similarity measure used is based on content relevance and user proximity. TwitterNews proposed in (Hasan et al., 2016b) is a local sensitive hashing-based approach to identify the uniqueness of tweets and then cluster them based on the similarity threshold. TwitterNews addressed the problem of cluster fragmentation commonly encountered in incremental clustering. After generating a set of potential events, TwitterNews filters them and presents significant events along with a tweet selected automatically to represent each event cluster. Authors further proposed TwitterNews+ (Hasan et al., 2019), which uses term-eventIDs inverted index to cluster similar tweets. In the similar context, Latent Dirichlet Allocation (LDA) is used for event

detection (Li and Huang, 2011; Turgeman et al., 2022) problem. In the improvement of LDA process, Srijith et al. (2017) proposed one sub-story detection method based on hierarchical Dirichelt process. In contrast to the document-based methods used in the aforementioned methods, there are feature-based methods that first identify the keywords from the stream and then cluster bursty keywords to detect the events. Bursty information is the information that rapidly spreads in the network. In (Fung et al., 2005), binomial distribution is used to extract the bursty features, while wavelet transformation to find out the high energy words is used in (Weng and Lee, 2011). Further, (Mathioudakis and Koudas, 2010) reports a trend detection technique for Twitter by analyzing bursty information from tweets. Similarly, (Morabia et al., 2019; Li et al., 2012) used bursty segmentation to cluster keywords and identify the events. In SEDITWik (Morabia et al., 2019), all tweets received from the Twitter stream within the current time window are divided into segments and their details are stored for later use during the tweet segmentation stage. Hashtags are considered more important as they typically contain more information. The probability distribution of segments, retweet counts, user diversity, and user popularity are analyzed to identify abnormally active segments, which are then grouped into clusters in the next two stages. In the final step, these clusters are summarized for event identification. On the other hand, Chen and Roy (2009) converted spatiotemporal distribution into 3D signals to find out events based on wavelet transformation. Another methodology of event detection is structure-based methods (Silva and Willett, 2008; Akoglu and Faloutsos, 2009; Gao et al., 2015) . A bipartite graph is used to form a cluster chain from the text features (Akoglu and Faloutsos, 2009) and a recursive event detection method was designed by observing the graph evolving over time. These structure-based methods are designed to find global events that are bursty in the entire text stream.

In contrast, local event detection targets to extract events more related to a particular small location. In order to geolocating events, Abdelhaq et al. (2013) proposed one framework called EvenTweet, which detects the local event by computing spatial entropy of each words and cluster localized words based on spatial distribution from several previous time windows. GeoBurst proposed in (Zhang et al., 2016) finds out location-specific events based on the geographical and semantic impact. They further proposed GeoBurst+ (Zhang et al., 2018), in which they replaced the manual ranking function with the keyword embedding to capture the subtle semantic of tweet messages to improve the performance of GeoBurst. Recently ML/DL methods are also proposed in the event detection of social streams. An event detection method MLEM (Liu et al., 2020b) works on the multi-lingual event mining model by generating the evolution graph of the streams. Liu et al. (2020a) proposed a two-way clustering method called EventX to find out the events from the news articles. EventX uses a classic machine learning classifier to classify the valuable features of the text. After that, it applies the graph-based clustering method to find out the events. However, EventX takes high computation time as it generates a large no of clusters. In 2021 a knowledge-preserving incremental heterogeneous graph neural network (KPGNN) (Cao et al., 2021) for incremental social event detection is proposed. Semi-supervised machine learning solutions are also use to solve event detection problems, e.g., recently, Marullo et al. (2023) proposed a multitask convolution network for event detection with the aid of semantic segmentation. Amirkhani et al. (2021) proposed semantic segmentation with multi-teacher knowledge distillation, in which five different CNNs are trained as teachers on various datasets, and the student gathers knowledge from various sources to perform numerous downstream tasks. Khosravian et al. (2021) used generative adversarial networks (GAN) and weather modelling to handle the data distribution disparity for intra-domain data. The majority of these methods work on the batch mode and do not dynamically find the events. Further, these methods need to deal with the information losses due to converting long text into other forms. These ML/DL based methods require some guided data to find out the events.

3. Problem Statement

Given a set of social media posts or interactions over time, we would like to mine localized micro-events. Before defining the problem mathematically, let us define Post.

Definition 1 (Post). *A post is a piece of text streamed on social media, which may contain keywords like hashtags and user mentions. A post $p_i = (auth, t, M, S)$ is a quadruple of author auth, time t, a set of mentions M, and a set of sentences S.*

In our work, we extend the definition of an event that an event is a significant thing that occurs at some specific place and time as well as it has long-term engagement on a topic. In other words, if a topic is active for an unusually longer time without any break, then we define the topic as an event. Based on this the problem statement is as follows.

Let $P = \{p_0, p_1, p_2, \cdots\}$ be a set of social media posts of a particular geographical region where we would like to mine a set of events $\{e_0, e_1, e_2, \cdots\}$ out of it.

4. CommunityINDICATOR: Community based event detection in social network

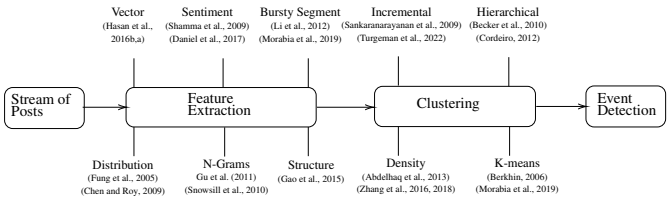

Figure 2: General working model of existing methodologies.

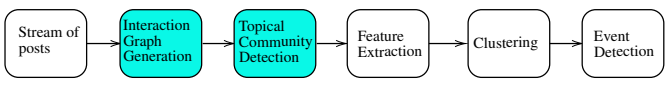

Figure 3: Working model of the proposed methodology.

In this section, we will present the proposed methodology for localized event detection from social media stream. Most of the existing unsupervised event detection methodologies work in four phases as shown in Figure 2. Methods used in each phase of existing methodologies are also shown in the figure. First, it collects the stream of posts and extracts the features using different feature extraction techniques. After extracting the features, they apply different clustering algorithms such as incremental, hierarchical, K-means, and density-based to detect the events. These existing unsupervised event detection methods are time-consuming. During the feature extraction, similarities of a large number of documents or texts are calculated, which costs a huge amount of time. In order to reduce the execution time of existing unsupervised methodologies, we bring in the concept of 'separation of concerns' from the software design principle. Here, we consider each event/topic as a separate concern. A similarity measure should be performed only within the same concern. Hence we incorporated the generation of an interaction graph. Each concern will form a different community in the interaction graph. A community detection algorithm is used to separate the concerns. We called all these communities topical communities as it is formed due to discussion on some topic.

An interaction graph is formed based on the interaction between people through social posts, e.g., in a post, one author mentions another user depicting an interaction between them. There will be an edge between the author of the post and the user mentioned in the post. On this interaction graph, we extracted the communities. Each community is the subgraph of an interaction graph called as topical community which depicts at least one topic in discussion. Before presenting the detailed algorithm, let us define the interaction graph, topical community, and related mathematical theories. List of symbols used here is shown in Table 1 for the reader's reference.

Table 1: List of symbols used in this manuscript.

Symbol	Description
p	Post
P	Set of posts
d	Time duration
τ	Set of time slices
k	Number of time slice
$\mathbb{C}$	Topical community
$S_{\mathbb{C}_i}$	Set of topics discussed under $\mathbb{C}_i$
C	Set of topical communities
c	Micro topical cluster
C	Set of micro topical clusters
r_{c_i}	Cluster representative of c_i
$\|c_i(r_{c_i})\|$	Number of posts present in cluster c_i
TC	Set of topical chains
δ	Similarity threshold
γ	Event threshold
$\mathcal{E}$	Set of events
$\Im_G(V,E,A)$	Interaction graph

Definition 2 (Interaction Graph). *Given a set of post $P = \{p(auth_i, t_i, M_i, S_i)\}$, an interaction graph $\Im_G(V, E, A)$ is the social graph where*

$$V = \bigcup_{p_i(auth_i,t_i,M_i,S_i)\in P} \{auth_i\} \cup M_i \tag{1}$$

$$E = \{(u,v) \mid \exists p_i(auth_i, t_i, M_i, S_i) \in P \; s.t.\; auth_i = u \text{ , and } v \in M_i\} \tag{2}$$

$$A = \{(u, S_u) \mid S_u = \bigcup_{p_i(u,t_i,M_i,S_i)\in P} S_i\} \tag{3}$$

Here A is stored as a node attribute in the interaction graph.

Given a set of post within a time starting from t_{start} till t_{end}, and a fixed time duration d, let $\tau = \{\tau_0, \tau_1, \tau_2, \cdots, \tau_{(k-1)}\}$ is the order set of time slices where $\tau_0 = (t_{start}, t_{start}+d)$, $\tau_1 = (t_{start}+d+1, t_{start}+2d)$, ..., $\tau_{(k-1)} = (t_{start}+(k-1)d+1, t_{start}+kd)$, and $k = \left\lceil \frac{t_{end}-t_{start}}{d} \right\rceil$ is the number of time slice. $\Im_G^{(\tau_i)}(V^{(\tau_i)}, E^{(\tau_i)}, A^{(\tau_i)})$ is an interaction graph formed by the set of posts $P^{(\tau_i)} \in P$ where $mint(P^{(\tau_i)}) \geq (t_{start}+id+1)$, and $maxt(P^{(\tau_i)}) \leq (t_{start}+(i+1)d)$. The functions $mint(\cdot)$ and $maxt(\cdot)$ give a minimum and maximum time of posts.

According to Girvan and Newman (2002), community is the dense subgraph where a group of vertices within which the connections are more denser than between different groups. Here we will define topical community.

Definition 3 (Topical Community). *A topical community $\mathbb{C}_i^{(\tau_i)}(G, S_{\mathbb{C}_i}^{(\tau_i)})$, is a densely connected subgraph G of an interaction graph $\Im_G^{(\tau_i)}$, where $S_{\mathbb{C}_i}^{(\tau_i)}$ is the topic under discussion and defined as $S_{\mathbb{C}_i}^{(\tau_i)} = \{s \in A_{\mathbb{C}_i}^{(\tau_i)} \mid A_{\mathbb{C}_i}^{(\tau_i)} = \bigcup_{u\in\mathbb{C}_i^{(\tau_i)}} A[u]\}$*

Definition 4 (Size of the Topical Community). *The size of topical community $\mathbb{C}_i^{(\tau_i)}(G, S_{\mathbb{C}_i}^{(\tau_i)})$ is the number of nodes having at least one topic in discussion i.e. $|\mathbb{C}^{(\tau_i)}| = |\{u \mid A[u] \neq \phi\}|$*

Given a topical community $\mathbb{C}_i^{(\tau_i)}(G, S_{\mathbb{C}_i}^{(\tau_i)})$, we extract the features in the vector form to find out the micro topic clusters. For converting one topical community into a vector, we have used the term frequency-inverse document frequency (TF-IDF). Given a TF-IDF vector of one topical community $\mathbb{C}^{(\cdot)}$, we are interested to find out micro topical clusters $\{c_1, c_2, ..., c_k\}$ discussed under one large topical community. We employed DBSCAN (Ester et al., 1996) clustering algorithm for the same. Since the number of micro topical clusters is unknown at the outset, it varies for each topical community and is strongly depends by the clustering parameter ϵ. Different experiments on clustering parameters setting are shown in Experiments 5.2.4 and 5.2.5.

Definition 5 (Successor ($\sqsupset$)). *Given two subsequent topical communities $\mathbb{C}^{(\tau_i)}$, $\mathbb{C}^{(\tau_{i+1})}$ and corresponding micro topical clusters $C^{(\tau_i)} = \{c_1^{(\tau_i)}, c_2^{(\tau_i)}, \cdots, c_k^{(\tau_i)}\}$, $C^{(\tau_{i+1})} = \{c_1^{(\tau_{i+1})}, c_2^{(\tau_{i+1})}, \cdots, c_j^{(\tau_{i+1})}\}$, let $\{r_{c_1}^{(\tau_i)}, r_{c_2}^{(\tau_i)}, \cdots, r_{c_k}^{(\tau_i)}\}$, $\{r_{c_1}^{(\tau_{i+1})}, r_{c_2}^{(\tau_{i+1})}, \cdots, r_{c_j}^{(\tau_{i+1})}\}$ be the set of cluster representative. We called $c_j^{(\tau_{i+1})}(r_{c_j})$ is the successor cluster of $c_k^{(\tau_i)}(r_{c_k})$ if r_{c_j} and r_{c_k} are similar, i.e., $c_k^{(\tau_i)}(r_{c_k}) \sqsupset c_j^{(\tau_{i+1})}(r_{c_j})$ if Similarity$(r_{c_j}, r_{c_k}) > \delta$. The representative with the highest score and larger than δ will be taken into consideration if more than one cluster representative exhibits similarity over the threshold δ.*

Definition 6 (Topical Chain). *A topical chain is order set of $\{c_{j_1}^{(\tau_i)}(r_{c_{j_1}}), c_{j_2}^{(\tau_{i+1})}(r_{c_{j_2}}), \cdots, c_{j_k}^{(\tau_{i+(k-1)})}(r_{c_{j_k}})\}$, where $c_{j_1}^{(\tau_i)} \sqsupset c_{j_2}^{(\tau_{i+1})} \sqsupset$*

$c_{j_3}^{(\tau_{i+2})}, \cdots \sqsupset c_{j_k}^{(\tau_{i+(k-1)})}$. *Size of the topical chain is the cardinality of the order set.*

δ is the similarity threshold that lies between 0 to 1 and controls the length of the event. A lower value of δ will allow a longer chain even if the topical clusters are not highly similar. Higher value may restrict longer chain even though they are similar. For example, if $\delta = 1$, then only the same sentence representative of two clusters will form a chain where as $\delta = 0$ will allow dissimilar topics to get connected.

Definition 7 (Event). *An event is defined by a topical chain that has a size greater than γ. We called γ as the Event Threshold that controls the size of an event or signifies the length of time duration for which a chain is labeled as an event.*

Properties of Event

1. *Active period of an event*: Active period is the period when a topic is in discussion. It is measured using the size of topical chain as follows:

$$Active\ period = d \times |\{c_{j_1}^{(\tau_i)}(r_{c_{j_1}}), c_{j_2}^{(\tau_{i+1})}(r_{c_{j_2}}), \cdots, c_{j_k}^{(\tau_{i+k})}(r_{c_{j_k}})\}| \quad (4)$$

2. *Volume*: Total number of posts during the active period of an event and it defined as:

$$Volume = |c_{j_1}^{(\tau_i)}(r_{c_{j_1}})| + |c_{j_2}^{(\tau_{i+1})}(r_{c_{j_2}})| + \cdots + |c_{j_k}^{(\tau_{i+k})}(r_{c_{j_k}})| \quad (5)$$

3. *Peak of the event*: We define peak of an event is the maximum number of posts in the active period of an event.

$$Peak = max(|c_{j_1}^{(\tau_i)}(r_{c_{j_1}})|, |c_{j_2}^{(\tau_{i+1})}(r_{c_{j_2}})|, \cdots, |c_{j_k}^{(\tau_{i+k})}(r_{c_{j_k}})|) \quad (6)$$

4. *Spike in the event*: A spike in an event is the point of the topical chain where the number of post is greater than the twice of the median of the whole chain. That is, an event has a spike at τ_{i+m} if

$$|c_{j_{m-1}}^{(\tau_{i+m})}(r_{c_{j_{m-1}}})| > 2 \times median(|c_{j_1}^{(\tau_i)}(r_{c_{j_1}})|, \cdots, |c_{j_k}^{(\tau_{i+k})}(r_{c_{j_k}})|) \quad (7)$$

5. For an event, if we fit a regression line considering each data point of the topical chain, then the value of slope can further categorize an event.
   - *Growing Event*: If the slope is positive then it is called as growing event.
   - *Decaying Event*: If the slope is negative then it is called as decaying event.
   - *Steady Event*: If the slope is 0 then it is called as steady event.

4.1. Algortihm

CommunityINDICATOR is an end-to-end solution to detect events from social media streams. Figure 3 shows the block diagram of the CommunityINDICATOR. The algorithm works in phases and it has six different phases for the detecting events from the social stream. In the initial phase an interaction graph is generated from the social posts. This interaction graphs is then split into communities. Each community represents larger events, discussing several micro-level activities. For example, in Figure 4, one large community is shown which consists three micro-level activities. The posts shared within this communities are clustered using clustering algorithm. Each of such clusters are potential micro-level events. Event detection is done by looking into the patterns of such clusters with time. The detail algorithms are explained in this Section.

Batch of posts of a fixed duration d, is extracted from input streams and an interaction graph of user mentions is created for it. The detail steps are shown in Algorithm 1. The input to the algorithm is the stream of posts $P^{(\tau_i)}$ of a single time window τ_i. We then extracted post author (*auth*), user mentions (M), and sentences (S) from the post p_i. V is the set of vertices where each node $v \in V$ represents a user id. For each user mentions, an edge is created between the post author and mentioned user present in p_i. Sentence of the post will be stored in the node attribute A. Finally, an interaction graph $\Im_G^{\tau_i}(V, E, A)$ is generated for a time window τ_i.

Algorithm 1 Interaction Graph Generation

Input: $P^{(\tau_i)}$, a stream of posts in time window τ_i
Output: $\Im_G^{\tau_i}(V, E, A)$, an interaction graph for a time window τ_i

$V, E, A \leftarrow NULL$
for each $p_i \in P^{\tau_i}$ **do**
  $M, S \leftarrow p_i$
  $auth \leftarrow$ Creator(p_i) ▷ Creator() returns the post creator
  $V \leftarrow V \cup M$
  $A(auth) \leftarrow A(auth) \cup S$ ▷ A is the node attribute in the graph
  for each $v_j \in M$ **do**
    $E \leftarrow E \cup \{(auth, v_j)\}$
  end for
  $\Im_G^{\tau_i} \leftarrow (V, E, A)$
end for

Algorithm 2 Macro Topical Community & Micro Topical Cluster Identification

Input: $\Im_G^{\tau_i}(V, E, A)$, an interaction graph for a time window τ_i
Output: $C^{\tau_i} = \{c_1, c_2, ..., c_k\}$, where c_i is micro topical cluster of posts in $\Im_G^{\tau_i}(V, E, A)$.

$C \leftarrow$ GetTopicalCommunities(G^{τ_i})
$C^{\tau_i} \leftarrow NULL$
for each $\mathbb{C}_i(G, S_{\mathbb{C}_i}) \in C$ **do**
  if $|\mathbb{C}_i| \geq threshold$ **then**
    $tfIdf \leftarrow$ TF-IDF(A)
    $C^{\tau_i} \leftarrow C^{\tau_i} \cup clustering(tfIdf)$
  end if
end for

Once we have the interaction graph, Algorithm 2, splits it into communities and clusters it. The graph $\Im_G^{\tau_i}$ generated from

Algorithm 3 Topical Chain Identification

Input: $C^{\tau_i} = \{c_1^{\tau_i}, c_2^{\tau_i}, ..., c_k^{\tau_i}\}$, **where** c_i **is micro topical cluster of posts, and** TC**, a set of existing topical chains**
Output: TC**, a set of updated topical chains**

1: **for each** $tc_i \in$ ActiveChains(TC) **do**
2: $c_i^{(\tau_{i-1})} \leftarrow$ GetLastNodeOfChain(tc_i)
3: flag = false
4: **for each** $c_i \in C^{\tau_i}$ **do**
5: $sim \leftarrow$ Similarity(LDA($c_i^{(\tau_{i-1})}$),LDA(c_i)))
6: **if** $sim \geq \delta$ **then**
7: AppendToChain(tc_i, c_i)
8: $C^{\tau_i} \leftarrow C^{\tau_i} \setminus c_i$
9: flag = true
10: break
11: **end if**
12: **end for**
13: **if** (!flag) **then**
14: Status(tc_i)$\leftarrow$ inactive
15: **end if**
16: **end for**
17: $TC \leftarrow C^{\tau_i} \cup TC$

the previous algorithm is the input for this algorithm. The function *GetTopicalCommunities(·)* returns topical communities of the interaction graph $\Im_G^{\tau_i}$. Any community detection algorithm will work for the purpose. However, we choose Label Propagation (Cordasco and Gargano, 2010) in our experiment over Greedy Modulairty (Clauset et al., 2004), Louvian (Blondel et al., 2008), and Asynchronous Label Propagation (Raghavan et al., 2007). The output of Line 1 is a list of communities C. Each $\mathbb{C}_i(G, S_{\mathbb{C}_i}) \in C$ represents the topical communities in the interaction graph $\Im_G^{\tau_i}$. As we are interested in finding out the micro-level communities, we generate the word embedding using *TF-IDF(·)* function for the sentences stored in $S_{\mathbb{C}_i}$ of a topical community $\mathbb{C}_i(G, S_{\mathbb{C}_i})$. We used a threshold to filter out these topical communities which do not have sufficient number of people discussing on one topic shown in Line 4. Next, we are passing word embedding vector to a clustering algorithm. It cluster out the topics $C^{\tau_i} = \{c_1, c_2, ..., c_k\}$ discussed in the large topical community $\mathbb{C}_i(G, S_{\mathbb{C}_i})$.

A set of micro topical clusters generated from the previous algorithm will be passed as input to the Algorithm 3. In the algorithm, set of micro topical clusters $C^{(\tau_i)}$ generated from the graph $\Im_G^{\tau_i}$, and a set of existing topical chains TC are passed as input. Now, the algorithm will check the cosine similarity between each LDA summarized $tc_i \in$ *ActiveChains(TC)* to $C^{(\tau_i)}$. Instead of constructing a embedding of the entire micro topical cluster, we have constructed an embedding on the LDA-based summarized text. *ActiveChain(·)* gives the list of topical chains from the existing topical chain set TC which are active for the queried time duration. Further, if micro cluster found then add to the topical chain tc_i and remove that micro cluster from original cluster set $C^{(\tau_i)}$. If no micro cluster is found then mark the existing topical chain with inactive flag. Remaining clusters are considered as new chain formed in this particular time slice and add to the list of topical chain set TC.

In our methodology, we keep data only for micro topical clusters generated from recent interaction graphs and the set of topical chains found so far. Events can be found with the query at any point in time as follows:

```
Query: for each tc in TC :
            if |tc| ≥ γ :
                E = E.add(tc)
       return E
```

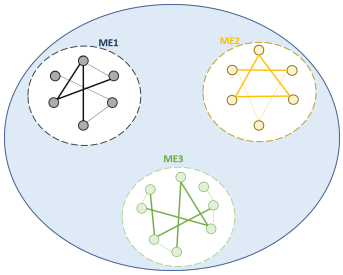

Figure 4: Schematic representation of the community based event which consists micro-level events ME1, ME2, and ME3.

4.2. Time Complexity

Algorithm 1 runs in the order of $O(|P^{(\tau_i)}| \times \exp(|M|))$, where $|P^{(\tau_i)}|$ is the size of posts and $\exp(|M|)$ is the expected size of user mentions present in the posts at time slice τ_i. Algorithm 2 completes in the order of $O(\exp(|C|))$ where C is the set of topical communities identified in $\Im_G^{\tau_i}$ and Algorithm 3 runs in the order of $O(No.\,of\,active\,chains \times |C^{\tau_i}|)$. The overall complexity of the proposed methodology for detecting events in a time slice τ_i is $O(|P^{\tau_i}| \times \exp(|M|) + \exp(|C|) + No.\,of\,active\,chains \times |C^{\tau_i}|)$.

4.3. Space Complexity

We only store the set of topical chains and set of micro topical clusters from the recent interaction graph. The overall space required for the proposed methodology for detecting events in a time slice τ_i is $O(|TC| + |C^{\tau_i}|)$. where TC is the set of topical chains, and C^{τ_i} is the set of micro topical clusters identified in $\Im_G^{\tau_i}$.

5. Experiments and Results

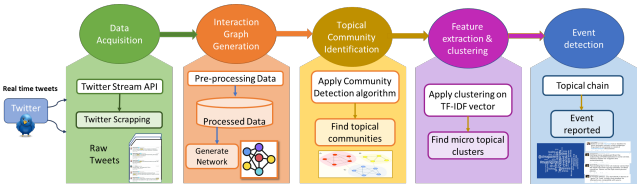

Figure 5: Structural outline of proposed methodology with Twitter social platform.

Extensive experiments have been performed on Twitter data of five cities. Different experiments were conducted to see how

the proposed methodology performs as compared to the state-of-the-art methodologies. In addition to that, we have conducted experiments for checking the performance of our approach with different parameters, and choices of different community detection algorithms. All the experiments were run on Windows 10 Intel Core i7 and 16 GB system memory support. Figure 5 shows the detailed working methodology of Twitter experiment. This section also includes detailed information about the data, analysis, and results obtained during the experiments.

5.1. Dataset Description

Tweets were collected for one month (July 15th to August 15th, 2021) from five metropolitan cities - Delhi, Mumbai, Bangalore, London, and New York - using Twitter Academic API's full-archive search. These cities were chosen for their high number of social media users, as they are the capital of the country, economy, and information technology hubs. Geolocation-based tweets were collected within a 25km radius of each city's center. Table 2 displays the number of tweets collected for each city. Four features were extracted from the data: post author, post creation time, user mentions, and sentences. Basic text preprocessing was performed to remove URLs, ASCII characters, emoticons, and stopwords to better understand the context of the discussion. Stemming was also applied to bring similar words to their stem form.

Table 2: Distribution of tweets scrapped from Twitter API

City	Number of tweets	Total No. of sentences	Avg. No. of sentences per tweet	Total No. of words	Avg. No. of words per tweet
Delhi	433305	72518	2	7038941	20
Mumbai	311711	56377	2	4418772	17
Bangalore	152069	34815	1	2373120	15
London	528846	175974	2	7871537	15
New York	610993	153067	1	7707244	13

5.2. Results

This section provides the experimental results. For all the experiments we have used the DBSCAN clustering technique with the parameters $\delta = 0.5$ and $\gamma = 2$.

5.2.1. Events identified in proposed CommunityINDICATOR

We have identified a significant number of events in each city. Figure 6 and 7 show the top five longest events identified for Delhi, Mumbai, Bangalore, London, and New York city. X-axis of the graph shows the time interval based on one day and Y-axis of the graph represents the number of tweets in each time interval. The label of each event is the representative sentence of the event topic. These labels are made human-readable for better understanding. It is visible from the figures that the top events identified contain localized events. For example, one event showed that a person requesting the government to arrange a bed in the city hospital, water and electricity problems in local due to heavy rain etc., are identified for Delhi city. Based on the event properties discussed in earlier section, graph shows the active period of one particular event, e.g., when it is started and for how many days it was active. For the 'water electricity problem' event it started on day 7 with 10 tweets followed by 15, 52, 12, 38, and so on till the last day of one month period. The volume of the event is 492 tweets which also signifies that more number of people are discussing about this topic. Peak represents the maximum number in the volume of an event such as for 'water electricity problem' event has peak of 52, and 'hospital bed problem' event has peak of 88 etc. Further, we have identified 167 growing, 187 decaying and 11 steady events for Delhi. For example, 'satsang' event is a type of growing event, 'SCST change act' event is a type of decaying event, and hashtag '#HarGharAnn' is a type of stedy event.

Figure 6: Time span of top five events in the Delhi, Mumbai, and Bangalore city.

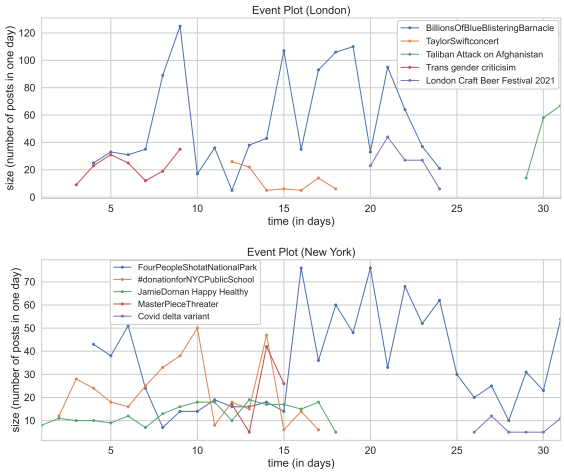

Figure 7: Time span of top five events in the London and NewYork city.

'SCST change act' event shows spike in the topical chain, visible in Figure 6.

Similarly, we have identified local events in Mumbai, for example, no vaccination is available at local hospital, one illegal project is going on near malad (place in the city). Identified events in Bangalore include bad road issue at local, pegasus attack on one local minister, P.V. Sindhu won the medal in hocky in Tokyo Olympic, etc. Based on the properties of events, We have identified 74 growing, 99 decaying, and 8 steady events for Mumbai and 25 growing, 35 decaying, and 6 steady events for Bangalore city. For example, 'no vaccine', 'water problem at local' is a decaying event, 'PMCM meeting' is a steady event with volume of 116 tweets and peak of 10, and 'actor marriage' is a growing event for Mumbai city with volume of 539 tweets and peak of 135. Same we have identified for Bangalore city. 'Sindhu badminton' event is an example of spike present in topical chain.

We got some good results for London, for example, there is one movement going on regarding blue blistering barnacle, london-carft beer festival 2021 is happening, transgender criticism is happening at one local street, etc. Similarly, some good local events are found for New York, e.g., four persons are shot at national park, film actor Jamie Dorman is admitted to hospital, covid delta variant case is found in neighboured, etc. We have identified 72 growing, 73 decaying, and 7 steady events for London and 48 growing, 56 decaying, and 10 steady events for New York. From the results, we can observe that our methodology detects large number of local events that can be reported to local municipal or law enforcement authorities to take appropriate action when required.

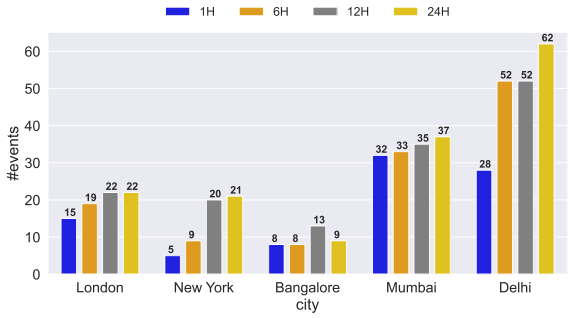

Figure 8: Effect of time duration d on number of events identified.

5.2.2. *Effect of different time duration*

In order to understand the effect of time duration on the result, we performed experiment with different time duration d. We varied d from 1, 6, 12, and 24 hours on five days (22nd October 2021 - 27th October 2021) of data. These five days were chosen because they showed a higher number of local events. Figure 8 shows the results for the number of events detected with d. It is found that when we are decreasing the time from 1 day to 12 hours to 6 hours to 1 hour, then the number of events decreases with respect to time. It is because when we varied d from 1 day to 1 hour, the chain breaks down in the middle due to unavailability of tweets and the length of the chain is becoming shorter and sometimes less meaningful. These shorter chains are then rejected by our algorithm based on the threshold γ. Thus lesser number of events are reported. For Delhi, the number of events reported is 28, 52, 52 and 62 for 1 hour, 6 hours, 12 hours and 1 day respectively. Similar results can be observed for Mumbai, London, and New York cities. The only exception was found with Bangalore city, where we are getting more number of events in 6 hours time duration. This is happening because some of the events are double counted due to the chain size just above the threshold.

5.2.3. *Effect of different community detection algorithms*

To check the effect of different community detection algorithm in the proposed methodology, we perform different experiments with four classical community detection algorithms namely Asynchronous Label Propagation (Raghavan et al., 2007), Label Propagation (Cordasco and Gargano, 2010), Modulairty Optimization (Clauset et al., 2004), and Louvian (Blondel et al., 2008). Detailed explanation of each method is as follows.

- Asynchronous label propagation algorithm assigns label to each node based on the label that appears most frequently among its neighbours. This algorithm is asynchronous because it updates the label of each node without waiting the updates on the remaining nodes.
- Label propagation algorithm is a semi-synchronous approach which removes the problem of randomization of asynchronous label propagation.
- Modularity Optimization algorithm calculates the modularity value on each node in community and joins the pair of communities that increases the modularity value till no such pair exists in the network. Modularity of each node can be calculated by below equation:

$$Q = \frac{1}{2m}\sum_{ij}\left[w_{ij} - \frac{d_i d_j}{2m}\right]\Delta\left(c_i, c_j\right) \tag{8}$$

  where c_i is the group to which vertex v_i belongs, Δ is the Kronecker delta function and m is total edge number.
- Louvian algorithm is based on hierarchical clustering algorithm, that merges the communities in a single node in a recursive manner and it executes the modularity based clustering on the condensed networks.

We used data of two days, from 22nd October 2021 to 24th October 2021 for two cities (London, and Delhi) in our experiment. We have checked the performance of each community detection algorithm based on evaluation parameters as precision, recall, F1-measure, accuracy, and execution time. For reference, these terms are defined as follows :

$$Precision = \frac{TP}{(TP + FP)} \tag{9}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{10}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{11}$$

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{12}$$

$$Execution\ time = \text{Total time required to run the algorithm} \tag{13}$$

Here TP represents true positive, TN denotes true negative, FP refers to false positive, and FN stands for false negative.

We created the ground truth by summing up all the verified events from all the community detection algorithm. Later in the text 5.3.1, a thorough explanation of how ground truth is produced is provided. Table 3 shows number of events detected in Delhi and London for each community detection algorithm. For Delhi, LPA and asynchronous LPA giving almost same amount of events but modularity is not able to identify some events. For example, 'to kill the terrorists which are harming the society'. On the contrary, in some cases modulairty and louvian merged some of the events although they were geographically apart. For example, the case of event 'complaints of water shortage and electricity cut in three different places', LPA and asynchronous LPA both are able to identify all three places as an individual event but modularity and louvian are making a big cluster and merging all three events. Louvian community detection algorithm is also giving good results but not identify some events in comparison with LPA ,e.g., 'oneplus strike in the city', 'SCST act is change', and 'there is some riot happening against one religious community'. Louvian community detection algorithm giving two extra events which are not present in any other community detection algorithm, these are 'fraud happened at local bank', and 'strike against Zomato'. Same results seen for the London dataset. Table 4 shows the precision, recall, F1 and accuracy for each algorithm and it is found that LPA and asynchronous LPA outperforms others in terms of precision. Modularity and Louvain algorithms suffer from the 'resolution limit problem'; it refers to the inability of the algorithm to detect small and densely connected communities within larger communities. The modularity optimization function used in community detection algorithms would measure the quality of a partition of nodes into communities based on the difference between the number of observed edges within a community and the expected number of edges if the nodes were randomly connected. However, when the community sizes become very small, or the subgraph becomes very dense, the expected number of edges becomes comparable to the observed number of edges, making it difficult for the algorithm to distinguish between true community structure and random fluctuations in edge density. As a result, the algorithm tends to merge small communities into larger ones, resulting in a loss of resolution and a bias towards larger communities. Because of this problem, Louvain and Modularity are not able to identify micro-level events present inside macro-level communities. LPA and LPA-asyn, on the other hand, both are fast and simple algorithm that can be easily scale to large networks and identify small clusters, which can also be the potential candidate for an event. Note that, we have experimented with classical community detection algorithms because the purpose was to show the comparing performance with different community detection algorithms. One can use any community detection algorithm with the proposed pipeline.

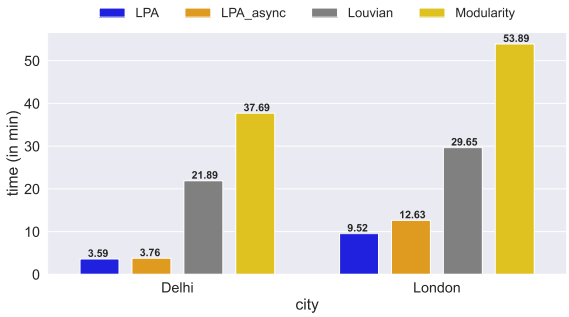

Figure 9: Execution time of different community detection algorithms.

Table 3: Number of events identified in each community detection algorithm

Index	Method	Delhi	London
1	LPA	22	9
2	LPA_asyn	21	9
3	Modulairty	18	6
4	Louvian	15	7

Figure 9 presents a bar graph showing the comparison of execution time among all four community detection algorithms. LPA outperforms all the other community detection algorithms. Modularity-based community detection algorithm is taking longest time to find the communities. Although Louvian is further optimized, it is still taking more time than LPA but performing better than modularity based community detection algorithm.

Table 4: Evaluation parameter analysis of different community detection algorithms.

City : Delhi				
Algorithm	**Recall**	**Precision**	**F1-Score**	**Overall Accuracy**
LPA	0.65	**0.88**	0.75	0.60
LPA_asyn	0.65	**0.88**	0.75	0.60
Modularity	**0.83**	0.59	0.69	0.64
Louvian	0.78	0.82	**0.80**	**0.72**
City : London				
Algorithm	**Recall**	**Precision**	**F1-Score**	**Overall Accuracy**
LPA	**0.90**	**1.0**	**0.95**	**0.90**
LPA_asyn	**0.90**	**1.0**	**0.95**	**0.90**
Modularity	0.85	0.66	0.74	0.60
Louvian	0.88	0.78	0.83	0.70

5.2.4. Effects of clustering parameter ϵ

As we have used DBSCAN in the experiment for the proposed method, we wanted to see how the proposed method performs with the different values of ϵ parameter of DBSCAN. Figure 10 shows the effect of ϵ values on number of events. Graph shows that with the increase of epsilon value, the number of events increases. Hence we vary ϵ from 0.2 to 1.0. For ϵ = 0.9, the number of events has the peak. Therefore, we have selected epsilon value 0.9 for all of the experiments we conducted. We have also observed that when we take a small ϵ

value, we get events that share almost similar types of tweets. That is tweets which are tightly coupled come in one cluster. But when we are increasing ϵ value then it makes the cluster loosely coupled. It helps to find out more local events.

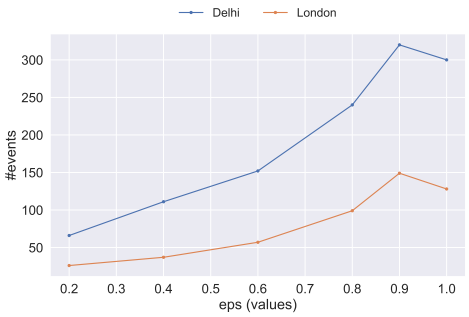

Figure 10: Effect of ϵ value on number of events identified.

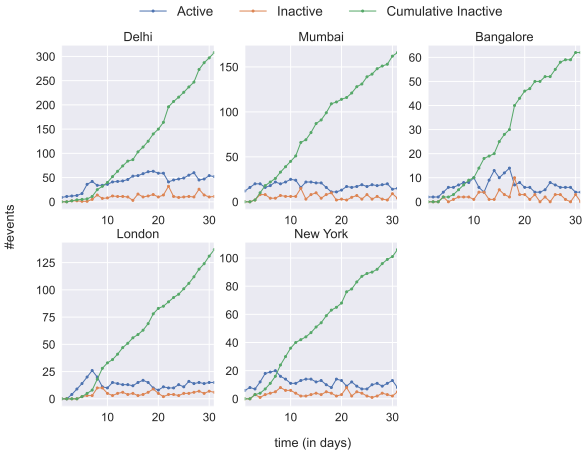

Figure 11: Number of active/inactive events identified in the stream.

5.2.5. *Effect of 'separation of concerns'*

As the major contribution of our proposed methodology is the introduction of 'separation of concerns' before feature extraction, we wanted to see the execution gain of our methodology provided. Hence we run the DBSCAN on TF-IDF vector of all the tweets of each time slice for two days without the 'separation of concerns'. We have selected two days data to visualize the results in better way because for more days of data, DBSCAN+TF-IDF model was not giving output in less than 10 hours. The execution time of this is compared with the proposed methodology. All DBSCAN parameters are same for both the execution. The results are shown in Table 5 and it is evident from the results that the proposed CommunityINDICATOR is more than hundred times faster. Both the methods provided almost same number and same type of events.

Table 5: Execution time of proposed CommunityINDICATOR and DBSCAN.

Index	City	Execution Time
1	CommunityINDICATOR(Proposed)	**155.32 sec**
2	DBSCAN+TF-IDF	20532.74 sec

5.2.6. *Number of active/inactive events in the stream*

As we have used *ActiveChains(·)* function in Algorithm 3, we wanted to see how many active and inactive chains we get from each day. Figure 11 shows the graph of active and inactive events present in one particular time duration for all the five cities. In the graph, active shows that topical chain of an event is active on that particular day. Inactive shows that topical chain of an event is not present in that particular day. Cumulative inactive shows that total topical chains closed before that particular day. It is evident from the figure that the variation of the number of active/inactive events per days is not much throughout the month of the study.

5.3. *Comparative Study*

This section provides comparative study of the proposed method with state-of-the-art methodologies for event detection.

5.3.1. *Ground Truth Creation*

In Twitter dataset, we do not have any label of the event. Therefore we have taken the help of two human annotators to check the authenticity of the events manually and created the ground truth. In the evaluation measures, a true positive is taken into account if an event has genuinely occurred and was recognised by the proposed algorithm, whereas a false positive is taken into account if the event has actually occurred but the proposed algorithm was unable to recognise it. True negative refers to an event that does not occur in reality and is also not identified by the proposed algorithm, whereas false negative refers to an event that does not occur in reality but is identified by the proposed algorithm. We have compiled all the events recognised by all the algorithms into a single set for the ground truth set. We conducted online searches for each events and utilised them as true positive to verify their veracity.

5.3.2. *Comparative Algorithms*

We have compared our method with three other algorithm as follows:

- **EvenTweet** (Abdelhaq et al., 2013) works on entities in the tweet and extracts bursty and localized entities as features, and apply clustering technique to make the cluster of similar words based on spatial distribution. EvenTweet partition the whole space based on N *x* N grid, and we used $N = 10$ for our experiment after tuning.

- **SEDTWik** (Morabia et al., 2019) identifies the event based on the bursty segment in the tweet. It calculates the probability of each segment with user frequency and cluster the similar tweet. SEDTWik has two parameters hashtag weight H and threshold for deciding candidate event cluster T and we used the same values ($H = 3$, $T = 4$) used in the original work

- **TwitterNews** (Hasan et al., 2016b) is a local sensitive hashing based approach to identify the uniqueness of a tweet and then used incremental clustering to cluster the

tweets for identifying events. In order to have a fair comparison in the comparative study, we used the time duration of cluster to 1 day instead of 15 minutes used in the original paper. This value is taken according to our proposed method. Similarly, the threshold for cluster size is also taken equal to the proposed method.

- **EventX** (Liu et al., 2020a) is a two layered clustering algorithm which creates the graph between the most common words of the tweet and apply the clustering algorithm to filter out the events out of it. All the parameters are considered as per the original paper.

5.3.3. Number of events identified

Table 6 shows the number of events generated by all the algorithms and Table 7 is showing the performance of our proposed method with other comparing methodologies in terms of evaluation parameters defined in Equations 9 to 13. For all the cities CommunityINDICATOR is identifying more real world events than other methodologies. From the Table 7, it is clear that our proposed methodology, CommunityINDICATOR, is giving comparable results with TwitterNews, EventX and higher results than other methodologies in terms of precision and recall. In terms of quality, our proposed one is identifying more real life localized events compare to other methods. In terms of accuracy also our proposed methodology outperforms the other. Even though TwitterNews and EventX were giving almost same number of events and same performance but both are very time consuming shown in Table 8.

Table 6: Number of events identified in proposed CommunityINDICATOR and other state-of-the-art algorithms

Index	Method	Delhi	Mumbai	Bangalore	London	New York
1	CommunityINDICATOR(proposed)	360	**181**	**66**	**152**	**114**
2	EvenTweet	97	82	35	96	78
3	SEDTWik	22	18	13	34	36
4	TwitterNews	340	168	60	143	110
5	EventX	**362**	176	64	150	113

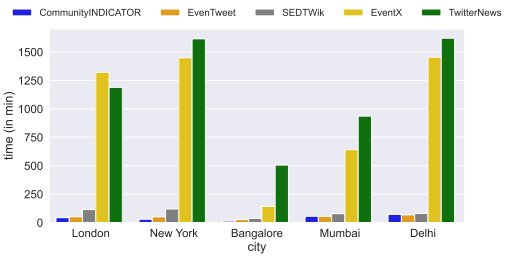

Figure 12: Execution time comparison with state-of-the-art algorithms.

5.3.4. Execution time

Figure 12 and Table 8 show the running time of each method for each city. CommunitINDICATOR and EvenTweet are outperforming all the other methods with significant amount of margin. For three cities viz. London, New York and Bangalore, CommunityINDICATOR is better performing than EvenTweet

Table 7: Evaluation parameter analysis with different state-of-the-art methodologies

City : Delhi				
Algorithm	**Recall**	**Precision**	**F1-Score**	**Overall Accuracy**
CommunityINDICATOR	0.80	**0.97**	0.87	0.78
EvenTweet	**0.88**	0.29	0.44	0.40
SEDTWik	0.81	0.06	0.12	0.24
TwitterNews	0.80	0.92	0.86	0.76
EventX	0.81	**0.97**	**0.88**	**0.79**
City : Mumbai				
Algorithm	**Recall**	**Precision**	**F1-Score**	**Overall Accuracy**
CommunityINDICATOR	0.72	**0.96**	**0.83**	**0.70**
EvenTweet	**0.89**	0.52	0.65	0.60
SEDTWik	**0.89**	0.15	0.20	0.34
TwitterNews	0.73	0.87	0.79	0.66
EventX	0.72	0.95	0.82	**0.70**
City : Bangalore				
Algorithm	**Recall**	**Precision**	**F1-Score**	**Overall Accuracy**
CommunityINDICATOR	0.85	**0.92**	**0.88**	**0.79**
EvenTweet	0.86	0.49	0.63	0.49
SEDTWik	**0.92**	0.20	0.32	0.3
TwitterNews	0.88	0.86	0.87	0.78
EventX	0.86	0.90	0.88	0.78
City : New York				
Algorithm	**Recall**	**Precision**	**F1-Score**	**Overall Accuracy**
CommunityINDICATOR	0.90	**0.93**	**0.92**	**0.85**
EvenTweet	**0.94**	0.66	0.78	0.65
SEDTWik	**0.94**	0.30	0.47	0.36
TwitterNews	0.91	0.91	0.91	0.83
EventX	0.91	0.92	0.91	**0.85**
City : London				
Algorithm	**Recall**	**Precision**	**F1-Score**	**Overall Accuracy**
CommunityINDICATOR	0.92	**0.93**	**0.92**	**0.86**
EvenTweet	0.93	0.59	0.73	0.59
SEDTWik	**0.94**	0.21	0.35	0.26
TwitterNews	0.95	0.90	**0.92**	0.85
EventX	0.92	0.91	0.91	0.85

Table 8: Execution time (in minutes) comparison with state-of-the-art algorithms.

Index	Method	Delhi	Mumbai	Bangalore	London	New York
1	CommunityINDICATOR(proposed)	71.22	**54.57**	**12.03**	**42.52**	**28.63**
2	EvenTweet	**70.11**	54.59	25.54	50.47	49.72
3	SEDTWik	79.43	77.69	35.86	113.73	118.81
4	TwitterNews	1453.99	641.13	141.54	1321.16	1449.05
5	EventX	1620.77	935.31	504.95	1188.52	1615.71

and for Delhi and Mumbai, our proposed method is having similar running time compared to EvenTweet. The major overhead in EvenTweet is because of the space partitioning strategy for computing spatial entropy to select the localized keywords, SEDTWik is taking time because to find the bursty segment which also can be large in numbers. TwitterNews takes more time due to similarity mechanism. The reason behind the poor execution time of EventX is due to the formation of graph in the early stage. Unlike the proposed method where the interaction graph is generated based on the social communication on a topic, EventX generates a graph based on the words+entity similarity. This choice leads to different graph structures, while the proposed method creates a scale-free graph with a largest connected component, EventX creates a disconnected graph with many components. Hence the proposed algorithm shows a significant amount of execution time gain over EventX. EventX has other drawbacks as well. It is not implemented for streams and ignores any new events if that is not part of the label data used in feature extraction.

6. Discussion and implications

Social platforms such as Twitter help to provide the activities happening around the world. Many event can be identified from twitter and similar platforms that are currently happening

in real life. Finding these events from the social streams requires a large amount of space and high computational power. The model should be swift enough to recognise the events when the continuous stream is coming with a huge amount of data. Considering these challenges, we have used the concept of 'separation of concerns' in the existing unsupervised machine learning pipeline by introducing interaction graph generation and identification of topical communities. The outcome of CommunityINDICATOR reduces the query time of identifying an event by 100 times. The proposed methodology identifies the micro-level events (gold medal in hockey, local satsang (religious activities), shortage of medicine in local hospitals etc.) with the macro-level events (Olympics sports, Covid-19 effects on nation). It helps journalists and local municipalities to react promptly and help to track the origin of an event.

After determining the macro-level topical communities, we take into account the topical clusters. Due to the fact that we are unsure of the precise number of topics discussed in a single macro-level topical community, we used the DBScan algorithm in our pipeline. K-means and K-median clustering algorithms can work more effectively if we can visualize or determine the number of topics in advance. The proposed CommunityINDICATOR algorithm offers a higher accuracy of up to 30% compared to EvenTweet and SEDTWik, with similar execution times. Furthermore, the proposed algorithm shows an execution time improvement ranging from 11% to 51% compared to TwiiterNews, and from 17% to 57% compared to EventX, on the different datasets used. TwitterNews and EventX produce nearly the same amount of events in the process, as seen in Table 6. However, the reason for not producing results in terms of execution time is because EventX is creating the graph based on word and entity similarity, which results in more edges, whereas TwitterNews is utilizing the similarity mechanism for one tweet to every other tweet. In order to ensure that a topical community is found, and it will be a potential candidate to become an event in the future, the proposed CommunityINDICATOR is creating the interaction graph based on the interaction between the users.

7. Conclusion

We presented an end-to-end solution to detect micro-level events from social data streams. We have added the concept of 'separation of concerns' in the existing unsupervised machine learning pipeline. We showed through experiments that the introduction of community detection on interaction graph reduces the execution time by 100 times.

Our algorithm was compared to state-of-the-art methodologies over a large dataset, and our proposed methodology was found to outperform others in terms of precision and accuracy while also being faster in the majority of experiments.

Use of interaction graph and community detection therein produces execution time benefit over existing methods. In this purpose different community detection algorithm detected similar no of events, however, it is found that LPA is taking less time compare to others.

One of the drawback of the proposed methods is that the detection of the events very much depends on the value of the time duration d. With change in value of d the chances of getting appropriate events will change. Further, micro-topical cluster detection can be improved based on better sentence representation, such as BERT-based contextual representation.

Acknowledgement

We would like to express our sincere gratitude to all the anonymous reviewers who provided valuable feedback and constructive criticism during the peer-review process. We would also like to acknowledge the Ministry of Education (MoE) for their financial support of this work.

Code

Code of the proposed methodology and data collection is available at link.

References

Abdelhaq, H., Sengstock, C., and Gertz, M. (2013). Eventweet: Online localized event detection from twitter. *Proc. of the VLDB Endowment*, 6(12):1326–1329.

Aggarwal, C. C. and Subbian, K. (2012). Event detection in social streams. In *Proc. of the 2012 SIAM international conf. on data mining*, pages 624–635, University of Auckland, NZ. SIAM, SIAM.

Akoglu, L. and Faloutsos, C. (2009). Rtg: a recursive realistic graph generator using random typing. In *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*, pages 13–28, Berlin, Heidelberg. Springer.

Allan, J. (2002). Introduction to topic detection and tracking. In *Topic detection and tracking*, pages 1–16. Springer, Boston, MA.

Allan, J., Papka, R., and Lavrenko, V. (1998). On-line new event detection and tracking. In *Proc. of the 21st annual international ACM SIGIR conf. on Research and development in information retrieval*, pages 37–45, Melbourne, Australia. ACM.

Amirkhani, A., Khosravian, A., Masih-Tehrani, M., and Kashiani, H. (2021). Robust semantic segmentation with multi-teacher knowledge distillation. *IEEE Access*, 9:119049–119066.

Atefeh, F. and Khreich, W. (2015). A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164.

Becker, H., Naaman, M., and Gravano, L. (2010). Learning similarity metrics for event identification in social media. In *Proc. of the third ACM international conf. on Web search and data mining*, pages 291–300, NY. ACM.

Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, Berlin, Heidelberg.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Cao, Y., Peng, H., Wu, J., Dou, Y., Li, J., and Yu, P. S. (2021). Knowledge-preserving incremental social event detection via heterogeneous gnns. In *Proc. of the Web Conf. 2021*, pages 3383–3395.

Chen, L. and Roy, A. (2009). Event detection from flickr data through wavelet-based spatial analysis. In *Proc. of the 18th ACM conf. on Information and knowledge management*, pages 523–532, Hong Kong China. Association for Computing Machinery.

Chen, Z., Hendrix, W., and Samatova, N. F. (2012). Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems*, 39(1):59–85.

Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6):066111.

Cordasco, G. and Gargano, L. (2010). Community detection via semi-synchronous label propagation algorithms. In *2010 IEEE international workshop on: business applications of social network analysis (BASNA)*, pages 1–8, Bangalore, India. IEEE.

Cordeiro, M. (2012). Twitter event detection: combining wavelet analysis and topic inference summarization. In *Doctoral symposium on informatics engineering*, volume 1, pages 11–16, Porto, Portugal.

Daniel, M., Neves, R. F., and Horta, N. (2017). Company event popularity for financial markets using twitter and sentiment analysis. *Expert Systems with Applications*, 71:111–124.

Deng, J., Qiao, F., Li, H., Zhang, X., and Wang, H. (2015). An overview of event extraction from twitter. In *2015 International Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 251–256, Xi'an, China. IEEE.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, Portland Oregon. AAAI Press.

Fedoryszak, M., Frederick, B., Rajaram, V., and Zhong, C. (2019). Real-time event detection on social data streams. In *Proc. of the 25th ACM SIGKDD International Conf. on Knowledge Discovery & Data Mining*, pages 2774–2782, Anchorage. ACM.

Fung, G. P. C., Yu, J. X., Yu, P. S., and Lu, H. (2005). Parameter free bursty events detection in text streams. In *Proc. of the 31st international conf. on Very large data bases*, pages 181–192, Trondheim Norway. Citeseer, VLDB Endowment.

Gao, Y., Zhao, S., Yang, Y., and Chua, T.-S. (2015). Multimedia social event detection in microblog. In *International Conf. on Multimedia Modeling*, pages 269–281, Cham. Springer.

Garg, M. and Kumar, M. (2016). Review on event detection techniques in social multimedia. *Online Inf. Rev.*, 40:347–361.

Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks. *Proc. of the national academy of sciences*, 99(12):7821–7826.

Gu, H., Xie, X., Lv, Q., Ruan, Y., and Shang, L. (2011). Etree: Effective and efficient event modeling for real-time online social media networks. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 300–307, Lyon, France. IEEE.

Hasan, M., Orgun, M. A., and Schwitter, R. (2016a). Twitternews+: a framework for real time event detection from the twitter data stream. In *International conf. on social informatics*, pages 224–239, Cham. Springer.

Hasan, M., Orgun, M. A., and Schwitter, R. (2016b). Twitternews: Real time event detection from the twitter data stream. *PeerJ PrePrints*, 4:e2297v1.

Hasan, M., Orgun, M. A., and Schwitter, R. (2019). Real-time event detection from the twitter data stream using the twitternews+ framework. *Information Processing & Management*, 56(3):1146–1165.

Kerman, M., Jiang, W., Blumberg, A., and Buttrey, S. (2008). A comparison of robust metamodels for the uncertainty quantification (uq) of new york harbor oceanographic data. *Journal of operational oceanography*, 1(2):3–13.

Khosravian, A., Amirkhani, A., Kashiani, H., and Masih-Tehrani, M. (2021). Generalizing state-of-the-art object detectors for autonomous vehicles in unseen environments. *Expert Systems with Applications*, 183:115417.

Li, C., Sun, A., and Datta, A. (2012). Twevent: segment-based event detection from tweets. In *Proc. of the 21st ACM international conf. on Information and knowledge management*, pages 155–164, Maui. ACM.

Li, W. and Huang, Y. (2011). New event detect based on lda and correlation of subject terms. In *2011 International Conf. on Internet Technology and Applications*, pages 1–4. IEEE.

Li, X., Sun, C., and Zia, M. A. (2020). Social influence based community detection in event-based social networks. *Information Processing & Management*, 57(6):102353.

Liu, B., Han, F. X., Niu, D., Kong, L., Lai, K., and Xu, Y. (2020a). Story forest: Extracting events and telling stories from breaking news. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(3):1–28.

Liu, Y., Peng, H., Li, J., Song, Y., and Li, X. (2020b). Event detection and evolution in multi-lingual social streams. *Frontiers of Computer Science*, 14(5):1–15.

Luo, L., Wang, Y., and Liu, H. (2022). Covid-19 personal health mention detection from tweets using dual convolutional neural network. *Expert Systems with Applications*, 200:117139.

Marullo, G., Tanzi, L., Ulrich, L., Porpiglia, F., and Vezzetti, E. (2023). A multi-task convolutional neural network for semantic segmentation and event detection in laparoscopic surgery. *Journal of Personalized Medicine*, 13(3).

Mathioudakis, M. and Koudas, N. (2010). Twittermonitor: trend detection over the twitter stream. In *Proc. of the 2010 ACM SIGMOD International Conf. on Management of data*, pages 1155–1158, Indianapolis Indiana USA. Association for Computing Machinery.

McMinn, A. J., Moshfeghi, Y., and Jose, J. M. (2013). Building a large-scale corpus for evaluating event detection on twitter. In *Proc. of the 22nd ACM international conf. on Information & Knowledge Management*, pages 409–418, San Francisco. ACM.

Morabia, K., Bhanu Murthy, N. L., Malapati, A., and Samant, S. (2019). SEDTWik: Segmentation-based event detection from tweets using Wikipedia. In *Proc. of 2019 Conf. of the North American Chapter of the ACL: Student Research Workshop*, pages 77–85, Minneapolis.

Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106.

Roldán, J., Boubeta-Puig, J., Luis Martínez, J., and Ortiz, G. (2020). Integrating complex event processing and machine learning: An intelligent architecture for detecting iot security attacks. *Expert Systems with Applications*, 149:113251.

Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., and Sperling, J. (2009). Twitterstand: News in tweets. In *Proc. of the 17th ACM SIGSPATIAL International Conf. on Advances in Geographic Information Systems*, GIS '09, page 42–51, New York, NY, USA. Association for Computing Machinery.

Shamma, D. A., Kennedy, L., and Churchill, E. F. (2009). Tweet the debates: Understanding community annotation of uncollected sources. In *Proc. of the First SIGMM Workshop on Social Media*, WSM '09, page 3–10, New York. Association for Computing Machinery.

Silva, J. and Willett, R. (2008). Hypergraph-based anomaly detection of high-dimensional co-occurrences. *IEEE transactions on pattern analysis and machine intelligence*, 31(3):563–569.

Snowsill, T., Nicart, F., Stefani, M., De Bie, T., and Cristianini, N. (2010). Finding surprising patterns in textual data streams. In *2010 2nd International workshop on cognitive information processing*, pages 405–410, Elba Island, Italy. IEEE.

Srijith, P., Hepple, M., Bontcheva, K., and Preotiuc-Pietro, D. (2017). Sub-story detection in twitter with hierarchical dirichlet processes. *Information Processing & Management*, 53(4):989–1003.

Turgeman, L., Avrashi, Y., Vagner, G., Azaizah, N., and Katkar, S. (2022). Context-aware incremental clustering of alerts in monitoring systems. *Expert Systems with Applications*, 210:118489.

Walther, M. and Kaisser, M. (2013). Geo-spatial event detection in the twitter stream. In *European conf. on information retrieval*, pages 356–367, Berlin, Heidelberg. Springer.

Weng, J. and Lee, B.-S. (2011). Event detection in twitter. In *Proc. of the International AAAI Conf. on Web and Social Media*, volume 5, pages 401–408, Barcelona, Catalonia, Spain. AAAI Press.

Zhang, C., Lei, D., Yuan, Q., Zhuang, H., Kaplan, L., Wang, S., and Han, J. (2018). Geoburst+ effective and real-time local event detection in geo-tagged tweet streams. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(3):1–24.

Zhang, C., Zhou, G., Yuan, Q., Zhuang, H., Zheng, Y., Kaplan, L., Wang, S., and Han, J. (2016). Geoburst: Real-time local event detection in geo-tagged tweet streams. In *Proc. of the 39th International ACM SIGIR conf. on Research and Development in Information Retrieval*, pages 513–522, Pisa Italy. SIGIR 2016.