Real-time Event Detection on Twitter

A Project Report Submitted in partial fulfilment of the degree of

Master of Computer Applications

By

Shubham Singh (18MCMC52)

Nitesh Rawal (18MCMC12)

School of Computer and Information Sciences,
University of Hyderabad,
Gachibowli, Hyderabad- 500046, India

July 2021

CERTIFICATE

This is to certify that the Project Report entitled "**Real-time Event Detection on Twitter**" submitted by **Shubham Singh** bearing Reg. No. **18MCMC52** and **Nitesh Rawal** bearing Reg. No. **18MCMC12** in partial fulfilment of the requirements for the award of Master of Computer Applications, is a bonafide work carried out by him under my supervision and guidance.

The Project Report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. Rajendra Prasad Lal
Assistant Professor
School of Computer and Information Sciences,
University of Hyderabad

Dean,
Dr. Chakravarthy Bhagvati
School of Computer and Information Sciences, University of Hyderabad

DECLARATION

We, Shubham Singh and Nitesh Rawal hereby declare that this dissertation entitled "**Real-time Event Detection on Twitter**" submitted by me under the guidance and supervision of **Dr. Rajendra Prasad Lal** is a bonafide work. We also declare that it has not been submitted previously in part or in full to this or any other University or Institution for award of any degree or diploma.

Date:

Signature of Student:
Shubham Singh
Reg. No.: **18MCMC52**

Signature of Student:
Nitesh Rawal
Reg. No.: **18MCMC12**

ACKNOWLEDGEMENTS

We would like to take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

- We feel great pleasure to express our deep sense of gratitude to **Dr. Rajendra Prasad Lal**, our supervisor who generously showed his continuous support and guidance throughout our project.

- We would like to thank all the **Faculty of SCIS**, AI Lab staff, and non-teaching staff, the University of Hyderabad for their cooperation.

- I am grateful to **Dr. Chakravarthy Bhagvati**, Dean of the SCIS(UOH) and University of Hyderabad for providing the facilities and extending their cooperation during course.

- Last but not least we place a deep sense of gratitude to our family members and our friends who have been a constant source of inspiration during the preparation of this project work.

Shubham Singh

Nitesh Rawal

Abstract

In the last few years, Twitter has become a popular platform for sharing opinions, experiences, news, and views in real-time. Twitter presents an interesting opportunity for detecting events happening around the real-world events. The content (tweets) published on Twitter are short and pose diverse challenges for detecting and interpreting event-related information. Twitter can produce rich data streams for immediate insights into ongoing matters and the conversations around them to tackle the problem. There are various technique available in previous work for detecting an event from twitter data stream such as feature pivot, document pivot, topic modeling, unspecified and specified event. In our case we have followed a methodology where we have highlighted the key results such as entity extraction and filtering, computing similarity between entity and entity clustering to get the final output as finding the number of events on twitter especially those detected from the twitter data streams. We have performed an evaluation data set in an offline mode.

KEYWORDS: Twitter event detection; Social network analysis; data extraction and filtering; compute similarities; entity clustering;

Contents

List of Tables

List of Figure

Chapter 1

INTRODUCTION

What is Twitter and why should use it?

Twitter is a 'microblogging' system that allows you to send and receive short posts called tweets. Tweets can be up to 280 characters long and can include links to relevant websites and resources. Twitter users follow other users. If you follow someone you can see their tweets in your twitter 'timeline'. You can choose to follow people and organizations with similar academic and personal interests to you.

You can create your own tweets or you can retweet information that has been tweeted by others. Retweeting means that information can be shared quickly and efficiently with a large number of people. Twitter has become increasingly popular with academics as well as students, policymakers, politicians and the general public. Many users struggled to understand what Twitter is and how they could use it, but it has now become the social media platform of choice for many. The snappy nature of tweets means that Twitter is widely used by smartphone users who don't want to read long content items on-screen.

Twitter is perhaps the most popular microblogging platform and Social Networking Service on Which Users Can Post and Read short text messages known as Tweet that are written in a unique conversational style particular to the brevity of the Twitter medium. There are approximately 500 million tweets a day [20] (or 6K tweets per second on average on the complete Twitter Firehouse for stream of all tweets. These Millions of users share information on different aspects of everyday life through these social networking services. Information shared in these platforms range from personal status (i.e., opinions, emotions, pointless babbles) to newsworthy events, as well as updates and discussions on these events. Due to the informal nature of the tweets, and the ease with which they can be posted, Twitter users can be faster in covering an event than the traditional news media which sometimes results that an event may occur on Twitter first then the main stream media.

Social Networks are being increasingly used for news by both journalists and consumers alike. For journalists, they are a key way to distribute news and engage with audiences found that more than half of journalists' stated social media was their preferred mode of communication with the public [1]. Where journalists also use social media frequently for sourcing news stories because they "promise faster access to elites, to the voice of the people, and to regions of the world which are difficult to access" [2]. For consumers, according to a recent survey, social networks have surpassed print media as their primary source of news gathering. Factoring in journalists' predilection for breaking stories on social media, audiences often turn to social networks for discovering what is happening in the world.

Social Network Analysis

Social network analysis (SNA) is the process of investigating social structures through the use of networks and graph-theory [3]. It characterizes networked structures in terms of nodes (individual actors, people, or things within the network) and the ties, edges, or links (relationships or interactions) that connect them.

Examples of social structure commonly visualized through social-network analysis include social media networks, memes spread, information circulation, friendship and acquaintance networks, business networks, knowledge networks, difficult working relationships, social networks, collaboration graphs, and disease transmission. It is visual representation of social link in which nodes are represented as points and ties are represented as lines. These visualizations provide a means of qualitatively assessing networks by varying the visual representation of their nodes and edges to reflect attributes of interest.

SNA is the practice of representing networks of people as graphs and then exploring these graphs. A typical social network representation has nodes for people, and edges connecting two nodes to represent one or more relationships between them. The resulting graph can reveal patterns of connection among people. Small networks can be represented visually, and these visualizations are intuitive and may make apparent patterns of connections, and reveal nodes that are highly connected or which play a critical role in connecting groups together. As the network representation of a community grows, it becomes necessary to apply graph analytic techniques to compute the characteristics of nodes and the graph as a whole [4].

Event

An event as something that happens at some specific time and place, and the unavoidable consequences. Specific elections, accidents, crimes and natural disasters are examples of events [5]. Also define an activity as a connected set of actions that have a common focus or purpose. Specific campaigns, investigations, and disaster relief efforts are examples of activities

A news event as being any event (something happening at a specific time and place) of interest to the (news) media. They also consider any such event as being a single episode in a larger story.

Example: A speech at a rally might be an event, but it is an episode in a larger context: a presidential election. They use the term episode to mean any such event, and saga to refer to the collection of events related within a broader context.

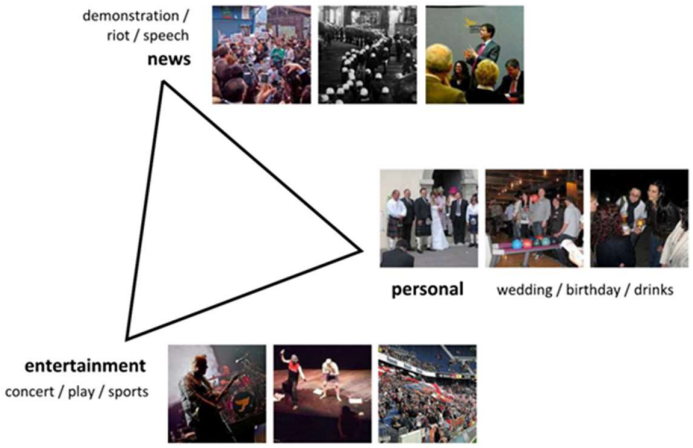

Fig-1 Example of Event categories in above image [6]

Application of Event Detection

In general, in order to detect an event, one must be actively looking for it. In other Words, the correct data must be collected, examined, and analyzed in order for an event to be detected. Given that it is still difficult to detect an event even while actively seeking it, the chances are slim that an event will be detected by happenstance alone. Therefore, it makes sense that the primary purposes of event detection are monitoring, surveillance, and management of systems and processes [6].

This section examines example event detection applications and classifies and organizes these applications according to their problem domains.

1. Identify Particularly Important Plays during Sporting Events such as Cricket, Football, Volleyball, Hockey etc. are detecting using tweets between the games.
2. Identify Earthquake or natural disaster from session data or from social media or from news texts on online but it is also been used in many different area like identify hardware issue or changes in hardware allocation and cloud based system.
3. Financial Markets Crises can be detected on Tweets using financial data for stock prices.
4. Areas like Cyber Security are detected using graph for bandwidth utilization for particular server where graph line can show the bandwidth of that server.

Health Monitoring and Management

In Health-care detection and Prediction of condition or events is also extreme importance in healthcare application. The Center for Disease Control and Prevention (CDC), for instance, continuously monitors medical and public health information from physicians and hospitals across the country. This practice is necessary for the earliest possible detection of viruses and disease like Covid19. Any detected wide-spread illness must be contained by quarantining

and treating the afflicted individuals. The objective is to prevent further spreading of the illness so that it does not result in an epidemic or, worse, a pandemic. The early detection of disease within individual patients presents another health management issue. Discusses screening and monitoring programs for the early detection of diseases such as covid19.

Environmental Monitoring and Prediction

Environmental monitoring and prediction is another common area for the application of Event detection methods. The earth's environment can be extremely violent, and early warnings of impending natural disasters such as tornadoes, hurricanes, tsunamis, earthquakes, floods, and volcanic eruptions are critical for the safety and security of populations within the affected regions. For example, Hurricane Ike recently devastated the city of Galveston, Texas. Due to the influence of early detection and warning systems, the majority of the populace was safely evacuated prior to hurricane landfall [MSNBC News Service, 2008]. Additionally, the contamination of natural resources, whether it be by natural or man-made (e.g., terrorist) causes, is another area of concern. Potable water, for instance, is continuously monitored by water utilities for purity and potential contaminants.

Safety and Security

Safety and security applications are other areas that utilize event detection methods. For instance, physical intrusion detection and fire safety are of critical importance to businesses and homeowners. Automobile, home, and corporate security alarm systems deter potential thefts and mischievous acts. Furthermore, fire, smoke, and carbon monoxide alarm systems increase survivability in the advent of a fire or buildup of toxic gas. Developing a prediction method for 9-1-1 call volumes can aid emergency service Providers in service planning and recognition of anomalous calls.

Business Process Optimization

Manufacturers rely heavily upon event detection methods to reduce overall maintenance costs and ensure compliance with requirements. Manufacturing and condition-based maintenance is one example. In industrial plants, engineers are concerned with identifying machines or processes that are in need of repair or adjustment. Business process compliance is another issue. In food and drug manufacturing, strict regulatory requirements obligate companies to certify that their products do not exceed specific environmental parameters during processing Furthermore, today’s fast-paced and constantly evolving high-tech business environment is extremely demanding on organizations and requires a detailed visibility into real-time business activities. Thus, the ability to efficiently detect correlated business process events that represent opportunities or problems to the organization and require quick action from the decision-maker is paramount.

Novelty of the Proposed Method

This work has resulted in a production application that solves various product use cases and improves metrics related to user exploration of ongoing events.

It involves following novel contributions:

1. **Tracking of event evolution over time -** Online social post streams such as Twitter timelines and forum discussions have emerged as important channels for information dissemination. They are noisy, informal, and surge quickly. Real life events, which may happen and evolve every minute, are perceived and circulated in post streams by social users. Which Representing event as a chain of clusters over time is a powerful abstraction Moreover, we are able to track these cluster chains in real time. Temporal analysis of clusters yields insights about sub-events and audience interest shifts. We highlight a case study of a high profile event on Twitter to demonstrate this.

2. **Differentiated focus on quality of clustering –** As author introduce new metrics for entity clustering quality that we believe can help ground subsequent efforts in the space. First we quantitatively evaluate the quality of detected event. Our model find the cluster of tweet that represent events. We are interested in quality of top event that are the most popular events on Twitter. For each method, we rank the detected events based on the number of tweets assigned to them and then pick the top events for each method.

3. **Novel real-time system design –**As author's contribution [7] is based on the realization that we can decompose burst detection and clustering into separate components that can be scaled independently. Through system profiling, we demonstrate the scalability and resilience of this approach. A large-scale Twitter data in real time combined detection and clustering of bursty terms in a sequential pipeline.

In our system design is based on realization that we can decompose each of the separate component of cluster service design we see that our system perform similarity computation and entity clustering by relying on the output of similar type of entities and shows the final result as number of events detected, where graph shows cluster of similar entities.

Thesis Structure

In the Chapter 2 we have described the related work based on the problem of event detection where most significant event detection technique are categorized as feature-pivot, document-pivot, topics model.

In the chapter 3 we have described about the methodology that are being used as clustering service design in which the mentioned terminologies are Entity extraction, compute similarity, similarity filtering, entity clustering.

In the Chapter 4 we have described about experiments and result of our Evaluation data set.

In the chapter 5 we have described about conclusion and future work of Real Time Event Detection on Twitter. In the last we have mentioned the bibliography that are linked to our projects.

Chapter 2

RELATED WORK

There are different survey papers in the literature focusing on summarizing different types of event detection technique. Where these techniques can be categorized as document pivot methods, where the primary features are tweets and feature pivot methods, where the primary features are the essential keywords, hashtags and other user-level features such as language-specific or user-specific information. In this paper[19] Summarized the event detection models for Twitter in terms of different techniques used, Term interestingness-based approaches, topic modelling-based approaches, incremental clustering-based approaches, and the other miscellaneous approaches. Furthermore, in [19] categorized the different event detection technique in terms of event detection types, i.e., Specified and Unspecified event detection, which use Supervised, unsupervised, and semi-supervised approaches

According to survey [11] the most significant event detection techniques can be broadly categorized as either **feature-pivot (FP) or document-pivot (DP)** methods.

Feature-pivot: These are based on detecting abnormal patterns in the appearance of features, such as words. Once an event happens, the expected frequency of a feature will be abnormal comparing to its historical behavior, indicating a potential new event.

Document-pivot: This category comprises methods that represent documents as items to be clustered using some similarity measure.

Topic modeling: This includes methods that utilize statistical models to identify events as latent variables in the documents.

2.1 Feature-pivot methods

Feature-pivot methods, this event detection method is rely on the detection of text features, which are likely to refer to events. These feature-pivot techniques, proposed originally for the analysis of time stamped document streams, consider an event as a bursty activity that makes some of the text features more prominent. The intuition behind these methods is that once an event emerges, certain features related to it will exhibit a similar abnormal rise in their frequency. The type feature can range from single keywords, named entities and phrase to social interactions. Traditionally, the distributions are detected, and finally, the discovery of events is conducted by grouping features that exhibit a similar behavior. This process is depicted in Figure 2 .According to this, an event is represented as a number of features showing an abnormality in appearance counts. The initial documents are assigned to the

detected events, based mainly on the appearance of the event-related features in them. These techniques cannot work in pure real-time fashion as there is a need for knowledge of the behavior of feature frequencies over a certain period of time. To overcome that limitation, an incremental is approach is usually followed, by detecting events on predefined timeslots. Another drawback of feature-pivot approaches is that as they depend on the detection of a bursty activity, they typically captures trends. Thus events that are not trendy and do no attract a lot of attention are likely to be missed [9].

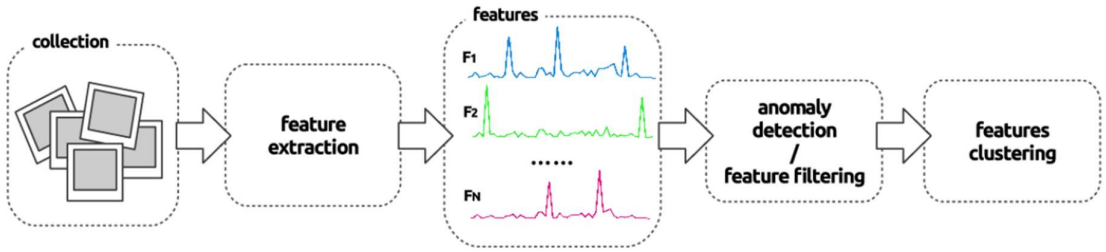

Fig. 2.feature pivot paradigm for event detection [9]

2.2 Document-pivot methods:

In this section discusses method that detect events by clustering documents on the basis of their semantic similarity. Document-pivot approaches, originating from the field to Topic Detection task (TDT) [17] can be seen as a clustering problem. Both RED and FSD approaches have been proposed. In both cases, the underlying principle is quite similar: documents are model in the way that capture their semantic content and then a clustering algorithm is applied to group them into events. What differs between approaches is mainly the characteristics of the clustering approach, the way that are documents are mapped to feature vectors, and the similarity metric used to identify whether two documents are from the same event [9].

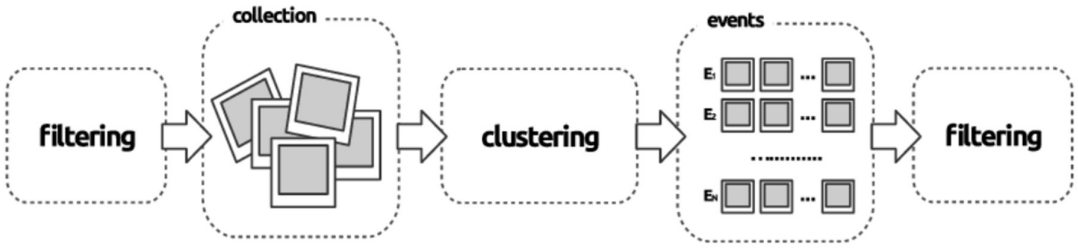

Fig. 3. Document-pivot event detection [9]

2.3 Topic modeling methods:

In this section describes approaches based on probabilistic models that detect events in social media documents in similar way that topic model identify latent topic in text documents. Originally topic models relied on word occurrence in text corpora to model latent topics as mixture of the identified set of topics. Latent Dirichlet Allocation (LDA) [19], which is the most known probabilistic topic modeling technique, is a hierarchical Bayesian model where a topic distribution is assumed to have sparse Dirichlet prior.

2.4 A Visual Backchannel for Large-Scale Events on Twitter:

Concept of a Visual Backchannel [12] as a novel way of following and exploring online conversations about large-scale events. Such as Twitter, are increasingly used as digital backchannels for timely exchange of brief comments and impressions during political speeches, sport competitions, natural disasters, and other large events. Currently, shared updates are typically displayed in the form of a simple list, making it difficult to get an overview of the fast-paced discussions as it happens in the moment and how it evolves over time. In contrast, our Visual Backchannel design provides an evolving, interactive, and multi-faceted visual overview of large-scale ongoing conversations on Twitter. To visualize a continuously updating information stream, in that include visual saliency for what is happening now and what has just happened, set in the context of the evolving conversation. As part of a fully web-based coordinated-view system In that Topic Streams has been introduce, a temporally adjustable stacked graph visualizing topics over time, a People Spiral representing participants and their activity, and an Image Cloud encoding the popularity of event photos by size. Together with a post listing, these mutually linked views support cross-filtering along topics, participants, and time ranges. In that design considerations has been discussed, in particular with respect to evolving visualizations of dynamically changing data.

2.5 Unspecified versus specified Event

Depending on the available information on the event of interest, event detection can be classified into specified and unspecified techniques, Because no prior information is available about the event, the unspecified event detection techniques rely on the temporal signal of Twitter streams to detect the occurrence of a real- world event. These techniques typically require monitoring for bursts or trends in Twitter streams, grouping the features with identical trend into events, and ultimately classifying the events into different categories.

On the other hand, the specified event detection relies on specific information and features that are known about the event, such as a places, time, type, and description, which are provided by the user or from the event context. These features can be exploited by adapting traditional information retrieval and extraction techniques (such as filtering, query generation and expansion, clustering, and information aggregation) to the unique characteristics of tweets. [13]

2.6 Challenges

There are various challenges to the problem of event detection. From twitter data stream in which the first of these is the scale and second one is brevity of the twitter medium (280 characters limit per tweet) and third one is noise where many of the tweets on the platform are unrelated to events and even those that are related can include irreverent terms and fourth one is dynamic nature of what is discussed on the platform that are actually events.

Chapter 3

Methodology

3.1 Framework

Clustering service design described the End-to-End Framework to the output entity clusters from twitter stream of data where each component in our system deals with the operation of specific stage and we have separated each component that can be scale independently at particular part of pipeline in order to improve overall output.
The Key advantage of this end-to-end framework is the modular compositions.to describe this clustering service design we first start with important terminology and then go through each of the component.

3.1.1 Terminology

Entity –

Entities provide metadata and additional contextual information about content posted on Twitter. The entities section provides vector of common things included in Tweets: hashtags, user mentions, links, stickers (symbols), Twitter polls, and attached media.

Examples: used in this work include named entities and hashtags but we can extend to other entity types such as user IDs or URLs.

Cluster -

A set of entities and their associated metadata (e.g., entity frequency count)
Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups.

Cluster Chain -

A list of clusters over time that is related to the same ongoing event.
In cluster chain there is an individual who act as source of message and transmit information to the pre-selected group off individual out of whom few individual again tell the same message to the other selected group of individual.

Event -
An event as something that happens at some specific time and place, and the unavoidable consequences. Specific elections, accidents, crimes and natural disasters are examples of Events. A cluster chain along with any metadata (e.g. detected start time, end time).

3.1.2 Entity Extraction and Filtering

In this context, Entity extraction we have discussed about extracting a tweet, Data preprocessing, and performed Named Entity Reorganization (NER) Within a raw text Data. The categorization system that these entities are sorted into can be unique to each other. Entities can be categorized into groups as broad as People, Organizations, and Places.

Here are some **example** entity types that are extracted from each tweet:
• **Named entities -** e.g. "Jason roy"
• **Hashtags -** e.g. "#indiavseng"

Before it's possible to extract entities from a text, there are several preprocessing tasks that need to be completed.

Extraction of tweet data using Tweepy:
Twitter allows us to mine the data of any user using Twitter API or Tweepy. The data will be tweets extracted from the user. The first thing to do is get the consumer key, consumer secret, access key and access secret from twitter developer available easily for each user. These keys will help the API for authentication.
Steps to obtain keys:
Login to twitter developer section

- Go to "Create an App"
- Fill the details of the application.
- Click on Create your Twitter Application
- Details of your new app will be shown along with consumer key and consumer secret.
- For access token, Click " Create my access token". The page will refresh and generate access token.

Tweepy is one of the library that should be installed using pip. Now in order to authorize our app to access Twitter on our behalf, we need to use the OAuth(Authorization key) Interface. Tweepy provides the convenient Cursor interface to iterate through different types of objects. Twitter allows a maximum of 3200 tweets for extraction.

Data Pre-Processing:

Once gathered the tweets data from twitter you need to prepare your data. Social media data is unstructured and needs to be cleaned before using it.
Preprocessing a Twitter dataset involves a series of tasks like removing all types of irrelevant information like RT, User Mentions, links, special characters, digit and extra blank spaces. It can also involve making format improvements, delete duplicate tweets, or tweets that are shorter than three characters.

Remove 'RT', **User Mentions** and **links:**
In the tweet text, we can usually see that every sentence contains a reference that is a retweet ('RT'), a User mention or a URL. Because it is repeated through a lot of tweets and it doesn't give us any useful information about event, we can remove them.

Convert tweets to lower case:
In order to bring all tweets to a consistent form. By performing this, we can assure that further transformations and classification tasks will not suffer from non-consistency or case sensitive issues in our data.

Remove numbers:
Likewise, numbers do not contain any sentiment, so it is also common practice to remove them from the tweet text.

Remove punctuation marks and special characters:
Because this will generate tokens with a high frequency that will cloud our analysis, it is important to remove them.

Removing stop words:
Stop words are function words that are high frequently present across all tweets. There is no need for analyzing them because they do not provide useful information. We can obtain a list of these words from NLTK stop words function.

After performing preprocessing in tweet data removed all type of irrelevant information and we extracted only Tweet id, user id, created time and tweet text from and store it input text file.

Tweet_id	User_id	Created_at	Tweet_text
1391429449309212	1391429450034724	Sun May 09 16:26:46 +0000 2021	Jason roy depart bowl bhuvi bhuvneshwar kumar england early wicket india #indvseng #indiavsenglan d #bhuvi

Table-1 pre-processed data

Named Entity Recognition (NER):
NER, short for, Named Entity Recognition is a standard Natural Language Processing problem which deals with information extraction. The primary objective is to locate and classify named

entities in text into predefined categories such as the names of persons, organizations, locations, events, expressions of times, quantities, monetary values, percentages, etc.

Example:

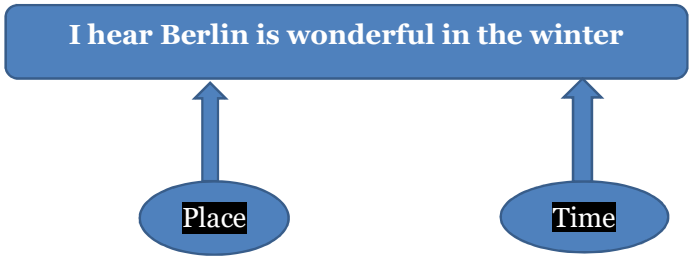

Fig-4 Example of NER

Recognizing named entities in a large corpus can be a challenging task, but NLTK has built-in method **'nltk.ne_chunk()**' that can recognize various entities shown in the table below:

NE Type	Examples
ORGANIZATION	Georgia-Pacific Corp., WHO
PERSON	Eddy Bonte, President Obama
LOCATION	Murray River, Mount Everest
DATE	June, 2020-06-29
TIME	two fifty a.m, 1:30 p.m.
GPE	South-East Asia, Midlothian

Table-2 NER Types and examples

For recognize named entities using NLTK. We have to import nltk library next we tokenize the sentence by using the method work_tokenize(), Also we tag each word with their respective part-of-speech tags using pos_tag(). The next step is to use ne_chunk() to recognize each named entity in the sentence.

3.1.5 Compute Similarities

A commonly used approach to match similar documents is based on counting the maximum number of common words between the documents. But this approach has an inherent flaw. That is, as the size of the document increases, the number of common words tend to increase even if the documents talk about different topics. The cosine similarity helps overcome this fundamental flaw in the 'count-the-common-words' with the remaining filtered entities.
In which we make Inverted index matrix where entities are in rows and tweet text are in column if the entities are matched from the tweet text data add 1 to vector otherwise add 0 in the vector.

Cosine Similarity

Cosine Similarity is a measurement that quantifies the similarity between two or more vectors. The cosine similarity is the cosine of the angle between vectors. The vectors are typically non-zero and are within an inner product space.

The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the Euclidean norms or magnitude of each vector.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Fig-5 Cosine similarity formula

Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the word counts of two documents.

- Cosine Similarity is a value that is bound by a constrained range of 0 and 1.
- The similarity measurement is a measure of the cosine of the angle between the two non-zero vectors A and B.
- Suppose the angle between the two vectors was 90 degrees. In that case, the cosine similarity will have a value of 0; this means that the two vectors are orthogonal or perpendicular to each other.
- As the cosine similarity measurement gets closer to 1, then the angle between the two vectors A and B is smaller.

The vector representations of the documents can then be used within the cosine similarity formula to obtain a quantification of similarity.

In the scenario described above, the cosine similarity of 1 implies that the two documents are exactly alike and a cosine similarity of 0 would point to the conclusion that there are no similarities between the two documents.

Cosine Similarity Example

Step 1: First we obtain a vectorized representation of the texts

Let us take the following example tweets to illustrate further

Tweet ID	Text
1	jason roy has depart bowled by bhuvneshwar england early wicket are Falling now
2	#indiavsengland Jason roy shocked by inswing ball of bhuvneshwar
3	Sanju samson awesome catch upfront Team india in a high pressure game # indiavsengland
4	bhuvneshwar man of the match results India win the game by 20 runs #indiavsengland
5	#indiavsengland looking forward to one more thrilling game as well from india

Table-3 Example Tweet text

We can represent the co-occurrences for entities as seen below:

	Tweet 1	Tweet 2	Tweet 3	Tweet 4	Tweet 5
#indiavsengland	0	1	1	1	1
bhuvneshwar	1	1	0	1	0

Table-4 Encoding of entities

The entity vectors for #indiavsengland and bhuvneshwar are the corresponding rows:

#indiavsengland = [0,1,1,1,1]
bhuvneshwar = [1,1,0,1,0]

Step 2: Find the cosine similarity

- Cosine similarity for two entities X and Y:
- $\cos(X,Y) = \frac{X \cdot Y}{|X| \, |Y|}$

For example: - $X \cdot Y = 0*1 + 1*1 + 1*0 + 1*1 + 1*0 = 2$

$|X| = \sqrt{0^2 + 1^2 + 1^2 + 1^2 + 1^2} = \sqrt{4} = 2$

$|Y| = \sqrt{1^2 + 1^2 + 0^2 + 1^2 + 0^2} = \sqrt{3} = 1.732$

cos(#indiavsengland , bhuvneshwar) = 2/3.464
= 0.5773

(57% similarity between the sentences in both document)

The potential disadvantage of this type of encoding is that it gets extremely sparse as we process more tweets, we avoid this by den-sifying the representation needed to update entity co-occurrences and frequencies. We observe that this type of cosine similarity works well in practice with respect to the final clustering output.

3.1.6 Similarity Filtering

After computing the entity similarities then, we can filter them based on the minimum threshold value S (in the range 0-1) to remove noisy connections between the entities.
Example: Let the threshold value be 0.2 if similarity between entities are greater than 0.2 then we add the edge between them otherwise if the value is less than 0.2 then we will remove noisy connection.

3.1.7 Entity Clustering

Once we get a set of event representative keywords, the next step is to cluster the keywords that are related to the same event. It can be said that two keywords belong to the same topic, if they are associated with similar content they belong to the same event, it is important to see that they follow similar appearance patterns too. Our similarity metric considers both of these aspects [16].
At this stage, we are able to naturally construct a graph consisting of the entities as nodes Where each entities represented by node and their similarities as edge weights. Once we can compute similarities by using cosine similarity, the advantage is that a wide variety of clustering algorithms can be leveraged.

For **example**, **community detection algorithms** have been used in similar settings [14].
One of the most popular algorithms of this type is the **Louvain method** [15], which relies on modularity-based graph partitioning. Some key benefits of Louvain is that it is efficient on even large-scale networks and has a single parameter, resolution R, to tune.

Community detection Algorithms
A community, with respect to graphs, can be defined as a subset of nodes that are densely connected to each other and loosely connected to the nodes in the other communities in the same graph.

Let me break that down using an **example**. Think about social media platforms such as Twitter, where we try to connect with other people. Eventually, after a while, we end up being connected with people belonging to different social circles. These social circles can be a group of relatives, school mates, colleagues, etc.

Louvain's method

Louvain's method for community detection is a method to extract communities from large network created by Blondel et al [18]. The method is greedy optimization method that appears to run in time O(nlog2n) in the number of nodes in the network.

Louvain's algorithm is based on optimizing the Modularity very effectively.

The quality of the communities referred as partitions hereafter is measured by Modularity of the partition. Modularity Q is defined as the formula shown in the below figure.

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

where

Aij is the weight of the edge between i and j.
ki is the sum of weights of the vertex attached to the vertex I, also called as degree of the node
ci is the community to which vertex i is assigned
δ(x,y) is 1 if x = y and 0 otherwise
m = (1/2)$\sum_{ij}$ Aij i.e number of links

Fig-6 Modularity Computation formula

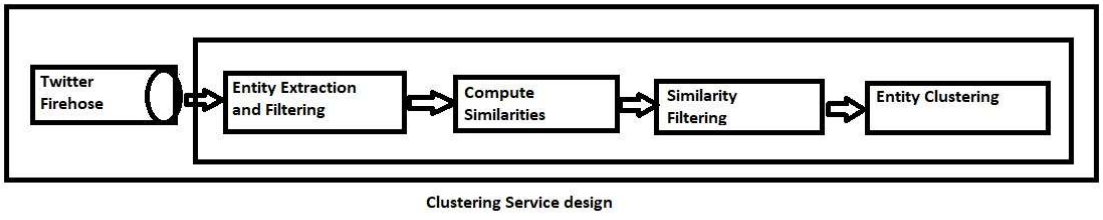

Fig-7 Clustering Service design

3.2 Algorithms

We describe the pseudo code for each of the component in the framework first we start with the entity extraction then moving forward to compute similarities and similarity filtering after that we perform algorithm for entity clustering.

Algorithm-1 Data Processing

Input: Any trending hashtag
Output: **List of tweet in json formate**

1. Procedure ProcessData
2. Consumer _key <- consumer key to twitter
3. Consumer _secret <- consumer secret to twitter
4. Access_token <- access token to twitter
5. Access_secret <- acces secret to twitter
6. Auth <- tweepy.OAuthHandler(Consumer_key,Consumer_secret)
7. Auth.access_token(access_token,access_secret)
8. Api <- tweepy.APT(Auth)
9. Class MyListner <- use to save the data in Json Formate.
10. Twitter_stream <- tweepy.stream(Auth,MyListner())
11. Twitter_stream.filter(track<-['#trending tweet','#'],)

Algorithm- 2 Inverted Index Making

Input: total entity list and tweet id
Output: return Inverted index as a vector with 1 and 0

1. inverted_index <- {}
2. entity_list <- list(set(total_entity_list))
3. for entity in entity_list
4. vector <- []
5. for tweet in keys_list
    a. if entity in tweet_dictionary[tweet]["entities"]
        vec.append(1)
    b. else
        vec.append(0)
6. inverted_index[entity] <- vector

return inverted_index

Algorithm- 3 Cosine Similarity

Input: list of entities
Output: cosine value between 0 and 1

1. cos_sim(entity_one , inv_entity_two)
2. c_sim <- round(dot(a, b)/(norm(a)*norm(b)),2)
3. return c_sim
4. for node in node_list
5. for nodex in entity_list
6. sim = cos_sim(inv_index[node], inv_index[nodex])
7. **End**

Algorithm- 4 Graph Generation between entities

Input: List of entities and inverted index

Output: graph based on entities as node and it similarity edge weight

```
1.  EG <- nx.Graph()
2.  G_node_list <- node_list
3.  List_to_remove <- []
4.  For each node in node_list do
5.          List_to_remove.append(node)
6.          Check_list <- list(set(g_node_list)-set(list_to_remove))
7.  For each node in check_list do
8.      Sim <- cos_sim(inv_index[node],inv_index[nodex])
9.      If(sim>0.2) then
10.      EG.add_edge(node,nodex,weight=sim)
11.     Else then
12. Continue
13. End
```

Chapter 4

Experiment and Results:

In addition to the procedures described above in the methodology, below we have perform all the experiment and steps to get the result.

4.1 Evaluation Data Set

To create an evaluation dataset, we first start with data extraction from Twitter API of #indiavsengland (hashtag) tweets that was played an odi match between india vs England. We got the extracted data in the form of json format.[link]
Extracted data json format as given below image –


```
{
  "created_at":"Sun Mar 28 12:23:43 +0000 2021",
  "id":1376147998860156929,
  "id_str":"1376147998860156929",
  "text":"RT @sunnyvihar1: Man in form J Roy bowled!!!\n@bhuvi gets him after beaten for 14runs.\nENG -14'
  "source":"\u003ca href=\"https:\/\/lol.co.com\" rel=\"nofollow\"\u003eBhurtya1\u003c\/a\u003e",
  "truncated":false,
  "in_reply_to_status_id":null,
  "in_reply_to_status_id_str":null,
  "in_reply_to_user_id":null,
  "in_reply_to_user_id_str":null,
  "in_reply_to_screen_name":null,
  "user":{
    "id":1217871248564224000,
    "id_str":"1217871248564224000",
    "name":"Pune News & Retweets",
    "screen_name":"RetweetsPune",
    "location":"Pune, Maharashtra, India",
    "url":null,
    "description":"#Pune #RT from in & around Pune & are not endorsements, Use #Pune for RT's\nRT's are
    "translator_type":"none",
    "protected":false,
    "verified":false,
    "followers_count":2157,
    "friends_count":30,
    "listed_count":13,
    "favourites_count":101965,
    "statuses_count":140592
  }
```


Fig-8 raw data

4.2 Pre-processing Tweet data

After extracting the data from twitter api for #indiavsengland which was in json format that is unstructured data so we need to clean these data and perform data pre-processing and entity extracting in that we got the data in form of 'tweet_id', 'User_id', 'tweet_text' and 'entities' with Named Entities and hashtag.

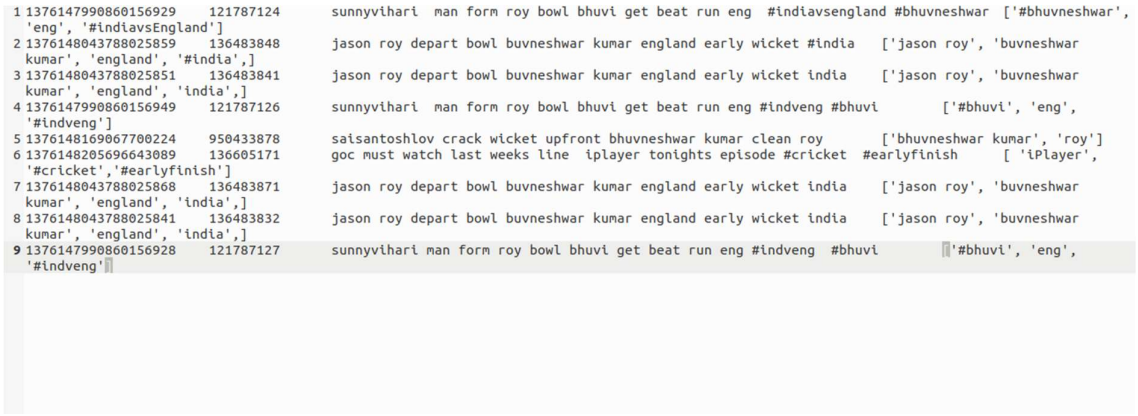

Fig-9 Tweet text with named entities and Hashtag

4.3 Total number of tweet

After getting the pre-processed data in text file then we have calculated total number of tweet using tweet-id for that we have created a list of key for tweet-id then calculate the length of list which gives total number of tweet.

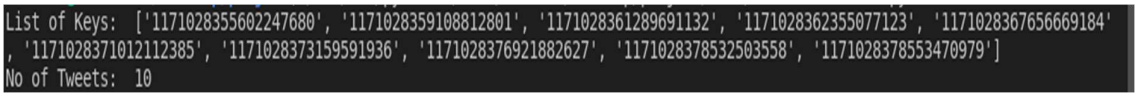

Fig-10 list of key

4.4 Tweet Dictionary

We have stored the pre-processed data from file to dictionary to make the tweet dictionary which contains tweet-id as key and tweet text , user-id , entities as their value for each of the tweet-id.

Fig - 11 tweet dictionary

4.5 Total number of entities

After creating the dictionary we have calculated the total number of entities by storing them as list from dictionary and them we have remove the duplicate entities which are repeated in a list by storing them into set after that we got the unique entity set.

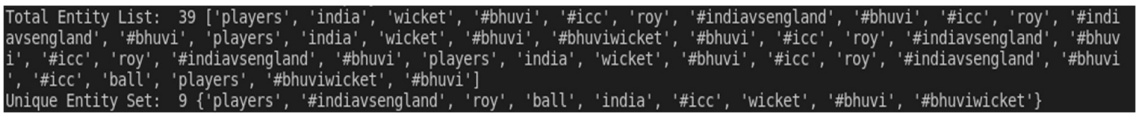

Fig-12 total number of entities

4.6 Inverted Index Making

After getting data in above format we check the similarity between tweet and list of entities and make inverted index if entities match in tweet text data then append vector to 1 other wise append to 0 in the vector.

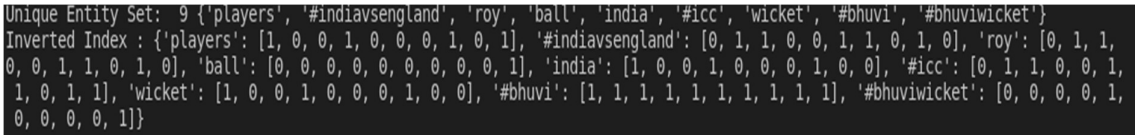

Fig-13 Inverted index between tweet and set of entities

4.7 Cosine Similarity

When we generate the inverted index between entities and tweet id we compute the cosine similarity between entities and generate the graph in graph each entities belong as node and edge are connected with nodes based on its cosine similarity value based on threshold value S if cosine similarity greater than threshold value then edge are connected with another node and its similarity value is its edge weight if cosine similarity value less than threshold value then edge are connected between node.

Fig-14 Cosine similarity between entities

4.8 Graph Generation

After getting cosine similarity between entities now we are at the stage where we can generate the graph between entities we have generated the Graph using networkx library in python.

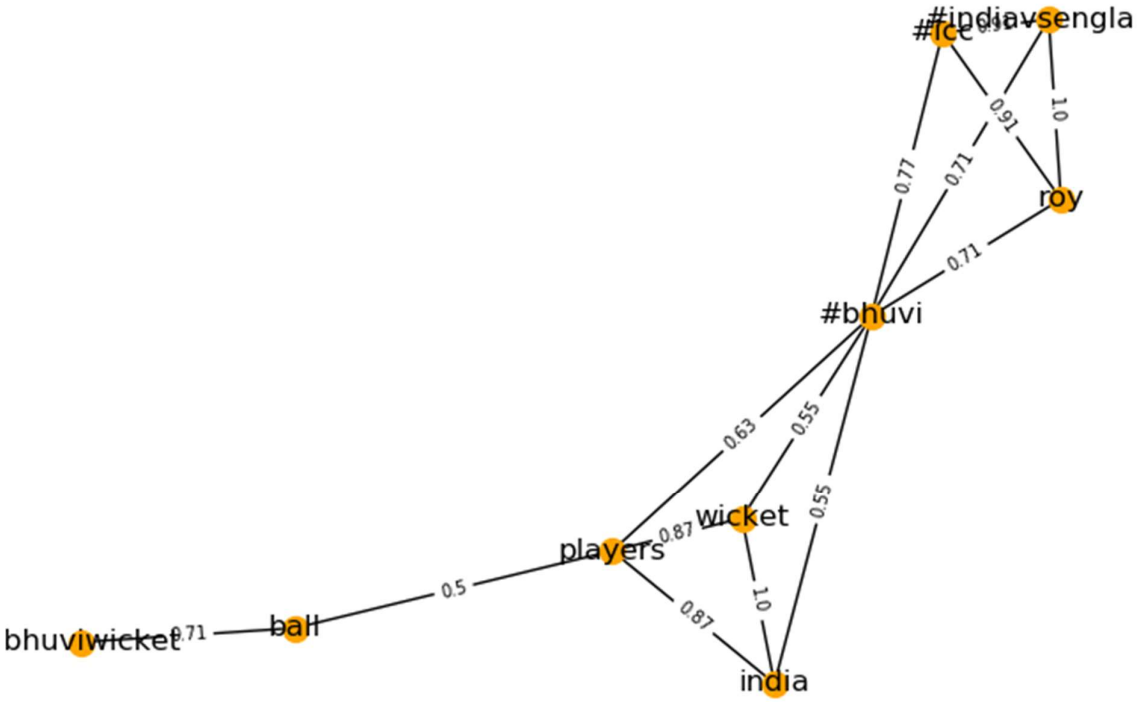

Fig-15 cluster of entities

4.9 Community detection

After generating the graph we have applied community detection algorithm using Louvain method to detect similar type of entities cluster belongs to community in graph where cluster of entities represent different events.

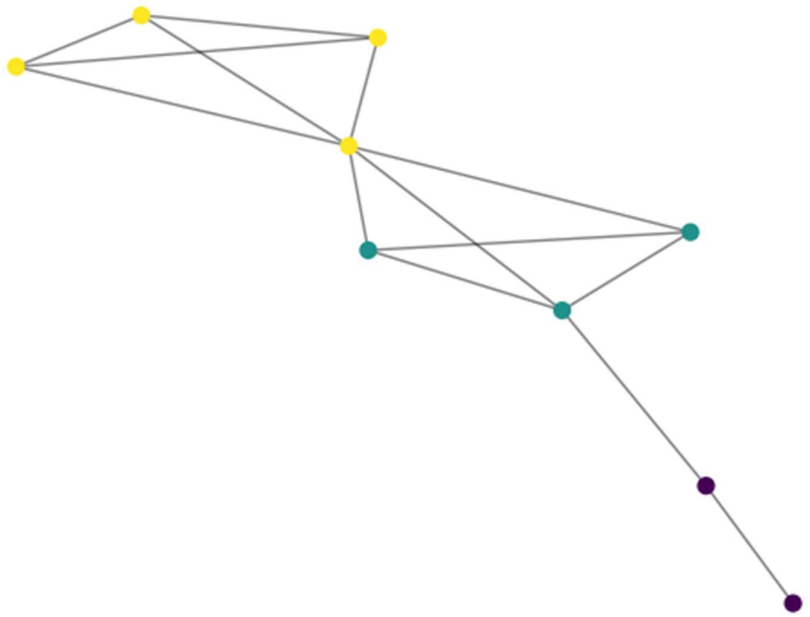

Fig- 16 number of events in graph

4.10 Total Number of Events

After generating the graph we have got our final result as total number of events related to each other**.**

```
Communities extracted using Louvain Method
----------------------------------------
Phase1- {'players': 1, '#indiavsengland': 2, 'roy': 2, 'ball': 0, 'india': 1, '#icc': 2, 'wicket': 1, '#bhuvi': 2, '#bhuviwicket'
: 0}
No_of_events = 3
Extracted events are=  {2: ['#bhuvi', '#icc', '#indiavsengland', 'roy'], 0: ['#bhuviwicket', 'ball'], 1: ['india', 'players', 'wi
cket']}
Phase2- {'players': 2, '#indiavsengland': 1, 'roy': 1, 'ball': 0, 'india': 2, '#icc': 1, 'wicket': 2, '#bhuvi': 1, '#bhuviwicket'
: 0}
No_of_events = 3
Extracted events are=  {1: ['#bhuvi', '#icc', '#indiavsengland', 'roy'], 0: ['#bhuviwicket', 'ball'], 2: ['india', 'players', 'wi
cket']}
```

Fig-17 Total Number of event

Conclusion

In our work, we have followed a Methodology for detecting an Event using Twitter Data Stream and we have evaluated our final output as number of events in an offline mode. First We have started with data extraction from Twitter Firehose using the Twitter API where extracted data was unstructured so we have applied data pre-processing and retain only specified things such as tweet-id, user-id, tweet-text and entities after that we have filter out entities in that to get the name entities and also filter the tweet text with hashtag and text only to do that entity filtering to filter the tweet data we have used NER (name entity recognition), POS(part of speech) tagging. After filtering the entities we have computed the similarities among the entities using Cosine Similarity by making the inverted index then we compute the cosine similarity between the entities. Based on similarity between entities we have made the graph after that we have detect the number of event in that graph. In the end Graph represents the entities with similar type which shows that these event are similar to each other and our final output is the total number of events related to each other and these are the number of event from twitter data stream. We have realized an implementation of such methodology and tested in offline mode. We verified that our approach is able to quickly detect all tested events and that the implementation was able to smoothly run on in our system.

Future work will include a wider evaluation, an additional labelling for the event entities, an optimization of the thresholds in order to improve the performances in terms of event detection speed and reduce the clustering complexity. Additionally, we will test our solution with different similarity metrics and with word embedding techniques to evaluate eventual performance improvements. Finally, we will try to detect the location of the recognized event based on the tweets and to produce a panic map in near real-time. And also the work can be further extended to make it region-centric and generate location-wise events. Also, for future work, different applications of the proposed event detection models such as health-care, emergency detection and disaster prediction, etc. can be explored.

Bibliography

1	2017. 2017 Global Social Journalism Study. https://www.cision.com/us/resources/research-reports/2017-global-social-journalism-study/?sf=false
2	Gerret Von Nordheim, Karin Boczek, and Lars Koppers. 2018. Sourcing the Sources: An analysis of the use of Twitter and Facebook as a journalistic source over 10 years in The New York Times, The Guardian, and Süddeutsche Zeitung. Digital Journalism 6, 7 (2018), 807–828.
3	https://en.wikipedia.org/wiki/Social_network_analysis#:~:text=Social%20network%20analysis%20(SNA)%20is,or%20interactions)%20that%20connect%20them.
4	https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_430
5	https://www.sciencedirect.com/topics/social-sciences/social-network-analysis
6	14th International Command and Control Research and Technology Symposium(ICCRTS), June 15-17, 2009, Washington, D.C.http://hdl.handle.net/10945/37485
7	L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Goker, I.Kompatsiaris, and A. Jaimes, ``Sensing trending topics in Twitter,'' IEEE Trans. Multimedia, vol. 15, no. 6, pp. 12681282, Oct. 2013.
8	Adrien Guille and Cécile Favre. 2015. Event detection, tracking, and visualization in twitter: a mention-anomaly-based approach. Social Network Analysis and Mining 5, 1 (2015), 18.
9	https://arxiv.org/pdf/1807.03675.pdf
10	Pei Lee, Laks VS Lakshmanan, and Evangelos E Milios. 2014. Incremental cluster evolution tracking from highly dynamic network data. In 2014 IEEE 30th International Conference on Data Engineering (ICDE). IEEE, 3–14.
11	Farzindar Atefeh and Wael Khreich. 2015. A survey of techniques for event detection in twitter. Computational Intelligence 31, 1 (2015), 132–164.
12	Marian Dork, Daniel Gruen, Carey Williamson, and Sheelagh Carpendale. 2010.A visual backchannel for large-scale events. IEEE transactions on visualizationand computer graphics 16, 6 (2010), 1129–1138.
13	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1039.8526&rep=rep1&type=pdf
14	Amosse Edouard, Elena Cabrio, Sara Tonelli, and Nhan Le Thanh. 2017. Graphbasedevent extraction from twitter. In RANLP17.
15	Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment 2008, 10 (2008), P10008.
16	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.401.7118&rep=rep1&type=pdf
17	James Allan. Introduction to topic detection and tracking. Topic detection and tracking, pages 1–16, 2002.
18	Fast unfolding of communities in large networks Vincent D. Blondel1;a , Jean-Loup Guillaume1,2;b , Renaud Lambiotte1,3;c and Etienne Lefebvre1
19	Z. Saeed, R. A. Abbasi, O. Maqbool, A. Sadaf, I. Razzak, A. Daud, N. R. Aljohani, and G. Xu, ``What's happening around theworld?Asurvey and framework on event detection techniques. 17, no. 2, pp. 279_312, 20
20	https://www.dsayce.com/social-media/tweets-day/