Extreme Event Detection and Management using Twitter Data Analysis

Girish K K[1], Jeni Moni[2], Joel Gee Roy[3], Afreed C P[4], S Harikrishnan[5], Gokul G Kumar[6]

[1,2] Assistant Professor, Computer Science Department, Providence College of Engineering, Chengannur, Kerala, India

[3,4,5,6] UG Scholar, Computer Science Department, Providence College of Engineering, Chengannur, Kerala, India

email: [1]girish.k@providence.edu.in

Abstract— **Events like the natural disasters are impossible to predict and are inevitable. The best course of action in that scenario is to effectively detect the event and respond to it accordingly. With quick detection and tangible information retrieval, casualties can be greatly reduced. Traditionally, this has been a tiring task as the data was siphoned from human observers or expensive equipment. In countries with scarce economies, the acquisition of these equipment can prove herculean and relying on individual human observers may lead to the spread of misinformation. An inexpensive and reliable solution to this problem can be found in the form of a system which facilitates the detection of extreme real-world events through the reactions of people on online social networks like Twitter. Unusual bursts in traffic from Twitter can be detected and this can be associated with an event. Moreover, the system can monitor the Twittersphere for rescue requests that appear on Twitter at the aftermath of an extreme event. Thanks to the containerized designed and hence its scalability, this model can undergo worldwide deployment with ease.**

Keywords*:* Data Mining; Text Mining; Twitter Analysis; Streaming data

I. INTRODUCTION

An extreme event is an occurrence that can cause significant damage to both lives and property. Conventionally, these events are reported manually over local communication lines at their occurrence. With this method, there is a great delay in the traversal of information and also allows room for false information to be spread. Similarly, a lot of spin-off events occur as the direct consequence of the initial extreme event. These isolated events need to be also handled to ensure efficient rescue operations and management. Twitter is a microblogging platform where users can post short messages termed as 'tweets' which can be of a maximum length of 140 characters. For years, extensive research has been done on the data generated by the Twitter platform and it has been found that the sentiments and opinions generated by the users on the platform stands true to the opinions that are present in real life. Twitter provides open APIs which can be used to access these tweets and analysis can be done on those tweets. Text analysis of tweets have been implemented for multiple applications ranging from sentiment analysis to stock market predictions. Furthermore, Twitter provides a robust platform to detect the emergence of news and other relevant events [1]. Aggregation algorithms are being used to summarize news topics and generate tangible headlines. 'Trends' provide a basic overview of the topics that are popular among the users and gives a good idea as to where the attention of an average user is. Our project aims to use the data generated by Twitter to improve the detection of extreme events and improve rescue efforts.

The default system used to inform authorities is the process of manually calling them up over a network. The people on ground zero of the disaster must call up the authorities and convey the information manually. Other systems include flood and earthquake sensors that detect these events upon occurrence and sends the notification to the authorities. There have been systems which facilitates scanning social media for the presence of specific keywords, but these provide high false positives. Spam accounts could easy repost the same tweet and this model would find it hard to detect them. The disadvantages of these methods include:

- Communication hierarchies involved causes great delay for the traversal of information.
- Messages may get disrupted and false information might get spread while information passes through multiple communication channels.
- Devices like earthquake sensors are expensive and hence is not scalable.

We propose a model that can effectively tackle these challenges and even goes beyond conventional methods by providing additional features that will boost the rescue efforts. The objectives of the proposed system are:

- To design a scalable extreme event detection and management system using Twitter data
- To model an effective early warning system to effectively manage an extreme event.

The rest of the paper is organized as follows. The analysis of the existing works is discussed in section 2. Section 3 and 4 discusses the system design and implementation methodologies respectively.

II. RELATED WORKS

Previous approaches were focused on probabilistic temporal models, decision support systems, long-term short-term averages, and the classification of shared common characteristics.

Michael et al [2] proposed a device that performs trend detection over the Twitter stream. One of the first challenges that this device tackles is to identify and evaluate the emerging topics that occur in the stream automatically and to do so in real time. Usually, developments are driven by emerging events, breaking news that draw broad users' interest. Twitter Monitor discusses the real-time trend detection problem, as well as its architecture, and ends with a demonstration scenario summary. Twitter Monitor performs trend detection in two steps and analyzes trends in a third step. Its advantage is Usage of recent event detection techniques for detecting events of global and local interest from the Twitter data stream. Marco et al [3] proposed a system that conducts real-time event detection in a timely manner. In order to gain greater situational awareness and to effectively alert authorities or check information gathered from other means, updated information on emerging situations of danger can be collected through social media. The consequences of the events are also analyzed to give a complete idea of the event ecosystem. In general, the method proposed is adequate and easily adaptable to deal with other types of events. The advantages of the system include its robustness and reliability, while its disadvantage is its susceptibility to fake accounts.

Wenwen et al [4] explores and summarizes common tasks in relation to event detection and proposed multiple new tasks to further improve the process. Social Media is reviewed to identify major events as well as subevent which may have arisen as a direct consequence of the major event. With this data, a timeline is constructed, which gives a basic overview of the events. Through further study of the responses to each particular event over time, information about the event itself, the views of people about the event, and causal associations between events (subevents) and responses can be inferred. Its advantages are the added importance given to the identification of subevents and the importance given to people's outlook of a specific event. Phuvipadawat et al [5] proposed a Twitter method for gathering, grouping, rating, and monitoring breaking news. Since short length messages make comparison of similarity difficult, to enhance the grouping outcomes, and increase of scores on proper nouns is done. Based on popularity and reliability variables, each category is rated. The existing method of identification is limited to information that are part of the messages. This model provides a ranking view of current news events and that is considered as it's advantage.

Jianshu et al [6] proposed a system that, by analyzing the Twitter text stream, detects certain events. The high flood of pointless terms typically overwhelms tweets describing such occurrences. In addition, given the sheer number of tweets, the event detection algorithm needs to be scalable. By applying wavelet analysis on the frequency based raw signals of the words, this model creates signals for individual words. By looking at their corresponding signal autocorrelations, it then cleans the trivial words away. The usage of wavelet analysis and the advanced filtration methods are this model's advantages.

Takeshi et al [7] proposed a system which investigates the real-time interaction of events such as earthquakes in Twitter and propose an algorithm to monitor tweets and to detect a target event. To detect a target case, a tweet classifier was developed based on features such as tweet keywords, word count, and background. Subsequently, a spatiotemporal probabilistic model was produced in order to locate the center and the location of the case. Every Twitter user was viewed as a sensor and Kalman filtering and particle filtering were applied. This model assumes that there are no two events at the same time. This can be valid in some cases, but not in the whole spectrum of serious occurrences. Earle et al [8] proposed an earthquake detection procedure that relies solely on Twitter data. A time series of tweets consisting of the word earthquake clearly indicates significant peaks linked to the time of origin of widespread felt events. A short-term, long-run average algorithm was used to classify potential earthquakes. The detections are normally triggered by widespread, more urgent incidents than those without any human effects. The detections are also quick; approximately 75 per cent take place within two minutes. While this model offers some concrete results and recognizes important events, it is not equivalent to modern seismic instruments. However, the detections are normally due to events that are more specifically significant than those without a human effect.

III. SYSTEM DESIGN

The proposed emergency event detection and management system consists of the following 2 modules: Web application module, and a cloud computing module. The Twitter data stream provides the system with real-time tweets and these tweets are acquired by the cloud computing instance. Tweet bursts are detected, and text analysis is done on the data stream. The inferences from the analysis phase are sent to the web dashboard to be viewed by the users. The block diagram of proposed system is shown in Fig. 1.

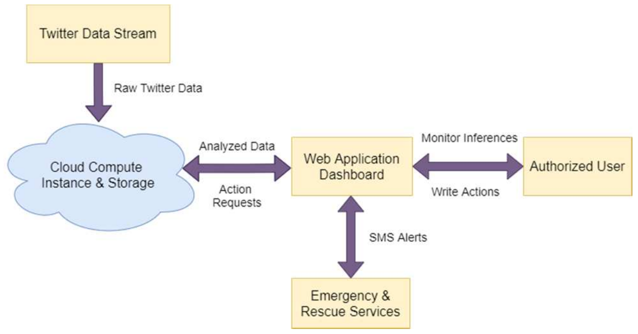


Fig. 1. Block diagram of proposed system

The architecture of the proposed system is designed with scalability, availability, and cost efficiency in mind. The functioning logic is expected to be mounted on the cloud to allow efficient data transfer fluidity and give room for future expansions. Fig. 2. shows the working of the proposed system. Twitter Streaming API is initialized which provides a continuous flow of real-time tweets in the cloud architecture. The cloud architecture holds the main logic of the system. Bursts are detected on the cloud architecture, and it does text analysis on the received tweet data to obtain event information. The web app can send SMS alerts to emergency services as needed. The web app is connected to the cloud, siphoning the data generated by it to be shown to the end user.

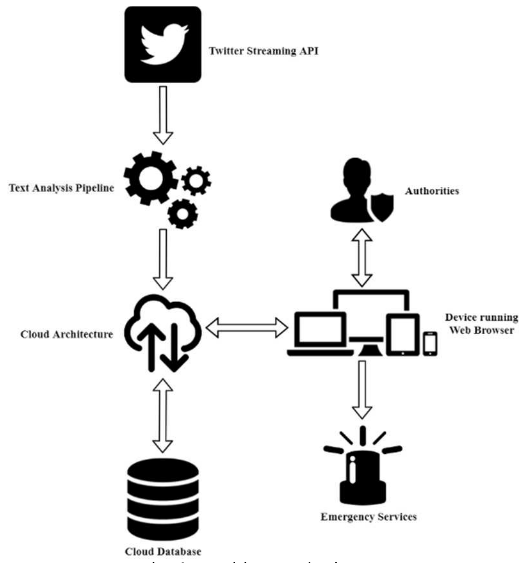


Fig. 2. Architectural Diagram

IV. METHODOLOGY & IMPLEMENTATION

The system is constructed using multiple programming languages and frameworks working together through compatible API endpoints and their corresponding query and request mechanisms. The tweets are analyzed on the text analysis pipeline, and the extreme event information is sent to the backend where MongoDB acts as the central database. The web app built using React and Next.js queries the database and shows all the relevant information to the user. A GraphQL server is used to process the queries from the backend and frontend. The Twitter API is polled frequently to gain access to live tweets as they come. The main components of the system work together to achieve the required objectives and improve the performance of one another by implementing compatible methods of interoperability. The GraphQL server acts as the central point of data communications as it controls and filters the flow of data from one component to another. Both the text analysis pipeline and the frontend of the system makes sure that the data is requested using queries that are compatible with the datatypes and schema setup at the backend to enable the smooth functioning of the entire system. The core of the application consists of the text analysis pipeline. The primary function of the text analysis pipeline is to analyze incoming tweets and produce inferences on the occurrence of an extreme event and its related information. The text data is passed through multiple workflows which involve the cleaning of the data, analysis of the data and the conversion of the data into compatible networking formats. All the steps involved in the text analysis pipeline work together to achieve the ultimate output which is to extract the required information from the tweets. This portion of the system is built mainly using the programming language Python. Multiple libraries and frameworks are used to perform linguistic processes on the text data to ensure the best possible results. The workflows involved in the pipeline includes removal of special characters and links, removal of stop words, extracting high frequency words using frequency count and LDA, entity extraction using NER, and data parsing for optimal data transfer.

A. Tweet Extraction using Twitter API

The first workflow of the text analysis pipeline involves the extraction of tweets using the Twitter API. Twitter API is a rate limited API that allows users to stream, search and filter tweets according to their requirements. The retrieved tweets are bootstrapped with a set of properties such as the tweet text, tweet author, creation time, location, etc. The properties also provide multiple indicators which can specify information such as whether the retrieved tweet is a retweet or not, whether the tweet was posted as a reply to another tweet etc. The tweets are received as a stream set and is processed accordingly. Each set of tweets is passed through the analysis workflow and the generated inferences are stored in the datastore. The tweet filters can be updated periodically to change the relevant keywords or structural patterns. There is a limit to how much tweets can be retrieved every second and measures are taken to ensure that the limits are respected.

B. Tweet Cleaning using Regular Expression

Raw tweets contain a lot of characters that are irrelevant to our extreme event detection workflow [9]. These include characters like ‘@’, ‘#’, etc. Furthermore, Twitter users sometime tend to include links in tweets, which are also not useful for the analysis chain. These characters and patterns need to be removed to ensure efficient and reliable detection results. Regular Expression is used to achieve this removal process. Multiple patterns pertaining to the occurrence of irrelevant characters are identified and appropriate filters are constructed. The initial tweet set is then replaced with the new content which contains the filtered data.

C. Stop Words Removal

Stop words consists of words or phrases that contribute to a sentence in a grammatical context but is irrelevant to analysis workflows. These are common words in a language which are usually regularly found in linguistic texts. Examples of stop words include "the", "is", "on", etc. Search engines and crawlers usually avoid stop words to ensure useful search results. Stop words need to be filtered out so that efficient language processing can be performed. There are multiple ways to perform stop word filtering. One of the methods is to construct a list of stop words and perform phrase matching to identify and remove stop words using this list. In this method, the stop words list is created by the user and hence has to be continuously curated to include new words as they become apparent. In this system, a stop word list curated by Spacy is used to perform the stop words removal. Spacy is a software library that aids natural language processing, and it provides a solid list of stop words that is used by multiple industrial entities.

D. Burst Detection

After the tweet set is thoroughly cleaned, processing is done on the tweet set to determine whether a 'burst' has occurred. A 'burst' refers to a sudden increase in tweet traffic in a particular timeframe. An occurrence of 'burst' indicates that Twitter users are suddenly talking about a topic, and in context of the event detection system, this means an event has occurred. It is vital for the system to detect these bursts so that the required information can be extracted from the corresponding tweet set. To perform burst detection, the incoming tweet set has to be analyzed to find whether the majority of the arrived tweets are mentioning an extreme event or not. For this, the most important topics of the tweet set has to be extracted. This topic set can be compared with phrases relating to extreme events and hence confirm the occurrence of a burst [10]. To retrieve the most important topics, the system uses a high frequency words extractor and Latent Dirichlet Allocation. Both methods work hand-in-hand to generate the list of most mentioned topics.

The high frequency words extractor tokenizes the words in the tweet set and counts their frequency. A dictionary of words is constructed which also contains the corresponding frequency of each word. Later, this dictionary is sorted to obtain the words with the highest frequency and are labelled as such. This method gives us a rough idea about the topic of conversation of the tweets; but that is not enough to construct a reliable topic model [11]. To ensure reliability, Latent Dirichlet Allocation or LDA is also used. LDA is a generative statistical model that is prominently used for topic modelling. LDA analyses a document and retrieves the most relevant topics or words from that document. In the context of the burst detection system, the document is the tweet set and LDA can be used to extract the topic of conversation from that tweet set. From the LDA model, a list of relevant keywords is obtained which can be used to construct the most important words list [12].

E. Location Extraction and Help Tweet Detection

Each tweet is analyzed to detect the location associated with the event and also to detect any help or rescue request tweets that may have been posted. To extract locations and help tweets, Named Entity Recognition is used. Named Entity Recognition or NER is the process of identifying and labelling key information from a given text. Each entity extracted is associated with a corresponding category and is labelled as such.

Training of the custom NER model involves the identification and setting up of the training data [13]. This meant that sample tweets had to be created and their entities had to be manually labelled to enable the nominal training of the model. Help tweets from past events can be used as reference and the training data with all the required notations will ensure the production of an efficient NER model. The custom NER model is expected to identify and label help tweets according to the structure and words of the tweet text. The custom NER model which was developed labels the request wordings as 'HLP' and the default spacy NER model labels the locations as 'GPE'. The tweet text is first analyzed to find whether if it is a help tweet or not. If so, the tweet is stored in an array as an object. The properties of the tweet object include the tweet text, timestamp, and location, which will be added later on in the workflow. After undergoing the help tweet categorization, the tweet text is analyzed to detect the locations associated with it.

If a location is detected, the location is stored to a location dictionary which holds all the locations discovered and their corresponding frequency. Each time a location is encountered in the tweet set, the corresponding entity in the dictionary is incremented to reflect the tweet count. If the tweet that is being analyzed was detected as a help tweet, the location that was just detected is added to the appropriate location field in the help tweet array. An abstract event list is constructed using the detected event type and the extracted locations. This data can now be parsed into a format that is compatible for data transfer using the GraphQL query system.

F. Data Parsing and Query Construction

Unlike REST APIs, GraphQL queries have their own datatypes and corresponding formats. Python uses a dictionary-based format to store its objects while GraphQL requires the data to be in JSON format. Furthermore, data cannot be sent plainly in a GraphQL system, it has to be neatly constructed into a GraphQL query that is compatible with the corresponding backend of the system. The abstract event list is used to construct the GraphQL query. The format function is used to populate the query string with the appropriate values. The requests library of the Python programming language is used to send the data as a POST request to the backend of the system.

V. RESULTS

The web application of the system is initiated as soon as the user accesses the URL of the app. First, they are directed to the home page. An authenticated user will be directed to the extreme event detection and management dashboard where they will be able to perform multiple actions in relation to the monitoring and the management of extreme events. The dashboard is further divided into multiple sections. The overview section gives the user exactly what the name suggests – an overview of all the events detected by the system. A map view (Refer Fig. 3.) is used to achieve this feature. Furthermore, the main events section and the subevent section gives the tabular views.

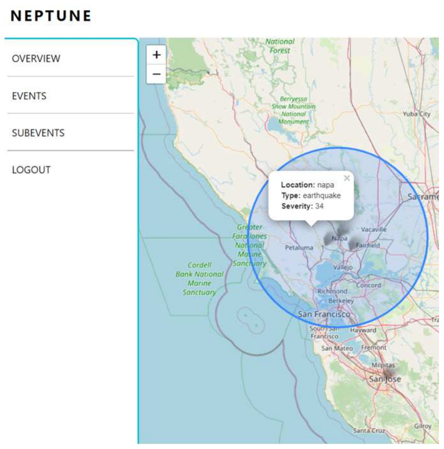
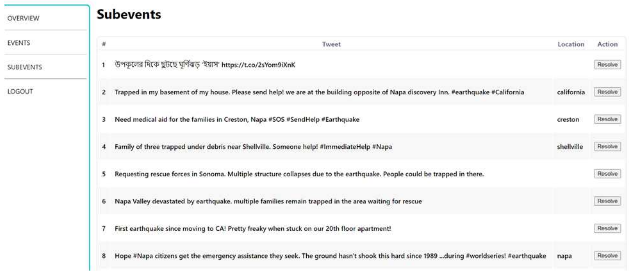


Fig. 3. Event detection



Subevents

#	Tweet	Location	Action
1	উপকূলের দিকে ছুটছে ঘূর্ণিঝড় 'ইয়াস' https://t.co/2sYom9iXnK		Resolve
2	Trapped in my basement of my house. Please send help! we are at the building opposite of Napa discovery Inn. #earthquake #California	california	Resolve
3	Need medical aid for the families in Creston, Napa #SOS #SendHelp #Earthquake	creston	Resolve
4	Family of three trapped under debris near Shellville. Someone help! #ImmediateHelp #Napa	shellville	Resolve
5	Requesting rescue forces in Sonoma. Multiple structure collapses due to the earthquake. People could be trapped in there.		Resolve
6	Napa Valley devastated by earthquake. multiple families remain trapped in the area waiting for rescue		Resolve
7	First earthquake since moving to CA! Pretty freaky when stuck on our 20th floor apartment!		Resolve
8	Hope #Napa citizens get the emergency assistance they seek. The ground hasn't shook this hard since 1989 ...during #worldseries! #earthquake	napa	Resolve

Fig. 4. Subevents table

The help request tweets are marked according to their locations on the map-view. On clicking the markers, user will get to know the contents of the help tweet text, the location and also the timestamp as to when that tweet was tweeted. The details of the major events that are detected are consolidated into a table in the 'Events' section of the web app. The event name, location and the tweet frequency are displayed on the table. A 'Send Alert' button is associated with each event that allows the user to initiate the SMS alert workflow. The system automatically fills in the event data into the autogenerated message, but this can be edited by the user. As seen in Fig. 4, help request tweets are also consolidated into a table on the 'Subevents' section of the web app. The tweet text is listed out along with any extracted locations associated with the tweet.

VI. CONCLUSION

Our Extreme event detection and management system can accept a stream of tweets, analyze it, and notify its users whether an extreme event has occurred or not. Furthermore, the users can have a degree of control over the extreme event's management by getting a bird's eye view of the area of occurrence and also are capable of responding to the event and alerting the rescue forces promptly. This means that events that can cause harm to the normal way of life can be detected early, and the damage can be greatly reduced. Rescue requests get directly registered into a central system and people who can help those in need will have direct access to that information. The system is mounted on a cloud architecture and thus allows for highly scalable workloads and provides great reliability.

REFERENCES

[1] B. Poblete, J. Guzmán, J. Maldonado and F. Tobar, "Robust Detection of Extreme Events Using Twitter: Worldwide Earthquake Monitoring," IEEE Transactions on Multimedia, vol. 20, no. 10, pp. 2551-2561, 2018.

[2] M. Mathioudakis and N. Koudas, "TwitterMonitor: trend detection over the Twitter stream," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, New York, 2010.

[3] M. Avvenuti, M. G.C.A. Cimino, S. Cresci, A. Marchetti and M. Tesconi, "Earthquake emergency management by social sensing," in Proc. of IEEE PerCom, 2014, pp. 587–592.

[4] W. Dou, W. Xiijuan, W. Ribarsky and M. Zhou, "Event detection in social media data," in Proceedings of the IEEE VisWeek Workshop on Interactive Visual Text Analytics-Task Driven Analytics of Social Media Content. 971-980., 2012.

[5] S. Phuvipadawat and T. Murata, "Breaking News Detection and Tracking in Twitter," in 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, ON, 2010, pp. 120-123.

[6] J. Weng and B. S. Lee, "Event Detection in Twitter," in Proceedings of the Fifth International Conference on Weblogs and social media, Barcelona, Catalonia, Spain, 2011.

[7] T. Sakaki, M. Okazaki and Y. Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors," in Proceedings of the 19th international conference on World wide web, 2010.

[8] P. S. Earle, D. C. Bowden and M. Guy, "Twitter earthquake detection: Earthquake monitoring in a social world," Annals of Geophysics, vol. 54, no. 6, 2012.

[9] Rukhma Qasim, Waqas Haider Bangyal, Mohammed A. Alqarni, Abdulwahab Ali Almazroi, "A Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification", Journal of Healthcare Engineering, vol. 2022, Article ID 3498123, 17 pages, 2022.

[10] Waqas Haider Bangyal, Rukhma Qasim, Najeeb ur Rehman, Zeeshan Ahmad, Hafsa Dar, Laiqa Rukhsar, Zahra Aman, Jamil Ahmad, "Detection of Fake News Text Classification on COVID-19 Using Deep Learning Approaches", Computational and Mathematical Methods in Medicine, vol. 2021, 2021.

[11] P. M. Jacob, Priyadarsini, R. Rachel and Sumisha, "A Comparative analysis on software testing tools and strategies," International Journal of Scientific & Technology Research, vol. 9, no. 4, pp. 3510-3515, 2020.

[12] Jacob, P.M., Mani, P., "A performance estimation model for software testing tools", International Journal of Engineering and Advanced Technology, 8 (4), pp. 248-253, 2019.

[13] R. R. Varghese, P. Mathew Jacob, S. S, D. M. Ranjan, J. Cherian Varughese and H. Raju, "Detection and Grading of Multiple Fruits and Vegetables Using Machine Vision," 2021 8th International Conference on Smart Computing and Communications (ICSCC), 2021, pp. 85-89