A dissertation submitted to the **University of Greenwich**
in partial fulfilment of the requirements for the Degree of

Master of Science
in
Data Science

Tracking Strike Action carried out by Healthcare Professionals

Name: Sweejalben Nehal Surti

Student ID: 001127898

School of Computing and Mathematical Science

Supervisor: Dr Mohammad Majid al-Rifaie
Submission Date: 17/01/2022
Word count: 11747

Tracking Strike Action carried out by Healthcare Professionals

Sweejalben Nehal Surti

Computing & Mathematical Sciences, University of Greenwich, 30 Park Row, Greenwich, UK.

(Submitted 17 January 2022)

ABSTRACT For a variety of reasons, healthcare workers have engaged in a range of actions, including strikes, marches, and civil disobedience. There appears to have been an upswing in action during the COVID-19 epidemic, with strike action and protests occurring on a regular basis all around the world. According to preliminary searches of news sources, there are at least a few examples of healthcare protest in news sources per day. It has been noticed by doing research that there has been little to no research on healthcare workers' protest actions. To date, we have not yet monitored social media activity, which is likely to reveal many more examples of protest. Because of its 4.48 billion active users from all over the world (Lambert, 2021), social media has become a large, rich and a vital data source for study in a variety of fields, including healthcare worker strikes. This information can be used to extract a variety of data that will be useful in a variety of applications.

The purpose of this study is to get a solution for the above situation, social media data can provide the needed insights from a global viewpoint to understand how frequent it is and the precise nature of this action. For this, 2.2 Lakhs historical tweets about Health workers protests and its metadata has been collected from Twitter using a third-party Library Snscrape using list of Keywords such as #Doctorprotest.Tweets are then Pre-processed and randomly 500 rows have been sampled to analyse. To automatically extract structured information from the unstructured and/or semi-structured text, Information Extraction technique i.e., Named Entity Recognition of NLP was used to identify the prevalence of protest, along with its features, essentially who is involved, the issues that are being challenged, Location, Hashtags, sentiment analysis and its Impacts etc. From this analysed data, the CSV of the dataset has been prepared and stored in the SQLite Database. Moreover, all above steps from data collection to Data storage have been automated. Finally, this data will be visualised on the Interactive map through the Web application, in which country wise data about protest will be displayed on the map. To analyse further, Topic Modelling (LDA) and Text summarizer has been applied to data. **Keywords**: Health worker protest, NLP, social media, Data extraction, Pre-processing, Named Entity Recognition, Sentiment analysis, Text mining, counter vectorization, Topic Modelling, Web Development, Python.

ACKNOWLEDGEMENTS

I would especially like to thank Dr. Mohammad Majid al-Rifaie for agreeing to be my supervisor and for his consistent advice, feedback, guidance, and support throughout the lifecycle of this MSc data Science project. I would like to thank Dr. Ryan Essex for providing the relevant research papers for reference, guidance, and support throughout the meetings. Also, I would like to thank the Institute for Lifecourse Development at the University of Greenwich for proposing such an interesting and useful project.

I want to thank both Dr. Mohammad Majid al-Rifaie and Dr. Hooman Oroojeni Mohamad Javad for agreeing to have the project demonstration on the scheduled day.

Table of Contents

List of Figures

List of Tables

List of Acronyms

NLP	Natural Language Processing
CSV	Comma Separated Values
HCW	Healthcare workers
HTML	Hyper Text Markup Language
AJAX	Asynchronous JavaScript and XML
LDA	latent Dirichlet allocation
TF-IDF	Term frequency-Inverse Document Frequency
NLTK	Natural Language Toolkit
NA	Not Available
OS	Operating System
NER	Named entity recognition

1. Introduction

1.1 Problem definition

Healthcare workers have been remarkably vocal in their protests. Healthcare workers have taken a variety of actions, including strikes, marches, and civil disobedience, for a variety of reasons. During the COVID-19 pandemic, there appears to have been an uptick in action, with strike action and protests taking place on a regular basis all over the world. Even though these situations are diverse in many ways and health care workers have gone on strike (or protested) for a variety of reasons, the common demands underlying nearly all these actions are insufficient responses to Covid-19 and inadequate protections for frontline workers; every group taking action has explicitly demanded more PPE (Ryan & Sharon, 2021). According to preliminary searches of news sources, there are at least a few examples of healthcare protest in news sources per day. It has been noticed by doing research that there has been little to no research on healthcare workers' protest actions. How we get to the root of these problems, as well as who should be held accountable for them and what may be done to solve them, will differ from country to country. Contrasts can be drawn across well-resourced countries, but global inequalities are much more pronounced, especially given Covid-19's projected future influence in low- and middle-income countries. However, in response to warnings concerning long-term consequences on the mental health of health-care employees, some immediate steps might be taken everywhere: help should be provided now and, in the future, (Ryan & Sharon, 2021).

To date, we have not yet monitored social media activity, which is likely to reveal many more examples of protest. As a solution for the above situation, social media data can provide the needed insights from a global viewpoint to understand how frequent it is and the precise nature of this action. Because of its 4.48 billion active users from all over the world (Lambert, 2021), social media has become a large and rich data source for study in a variety of fields, including healthcare worker strikes. So, the first step will be to answer a few basic questions, such as how common such behaviour is, where it occurs, and the types of behaviour that healthcare professionals frequently use – the goal of this project will be to capture these details through social media using Information Extraction technique i.e., Named Entity Recognition to identify the prevalence of protest, along with its features, essentially who is involved, the issues that

are being challenged, Location, Hashtags, sentiment analysis, Impacts and the protest action taken etc. The rapid growth of big data is aided by the growing popularity of social networking sites such as Facebook, Twitter, and Google. In this project I have focused on twitter data. Twitter is one of the most common sources of big data among social networking sites, where individuals from all over the world offer their opinions on a variety of themes and subjects. Tweets, or short messages of less than 140 characters posted over Twitter 1 service, have become a valuable information source for spotting trends and current happenings throughout the world, with a daily active user count of 100 million people. As a result, the job of named entity recognition (NER) for tweets, which tries to detect mentions of rigid designators from tweets belonging to named-entity categories such as people, organisations, and places (2007), has piqued researchers' interest.

What is Protest/Strike?

Protests are an important aspect of a democratic society, and they have the potential to shape the society's destiny. When a group of people protests, the rest of society joins in, either in support or in opposition to the group. This is extremely important for the advancement of society. The health workers received support from all over the world with thousands of people expressing their opinions on social media. Hashtags like #Doctorprotest, #SAVEMYANMAR, #Whatshappeninginmyanmar, #NHSCrisis, #NEETPG and #NurseStrikes were trending on twitter.

1.2 Project aims and objectives

In this project I hope to collect data from social media platform, analyse it using Natural Language Processing, and consolidate it so that it can determine the prevalence of protest, as well as its characteristics, such as who is involved, the issues being challenged, the protest action taken, their demand, the location, size and duration of the protest also the current status of the protest carried out and the impact of the strikes.

What is Natural Language Processing?

Natural language processing (NLP) is a field that combines computational linguistics, computer science, and artificial intelligence to allow machines to interpret, analyse, and generate natural human speech. In the 1950s, the first genuine application of NLP approaches was in a Russian-to-English translation that featured multiple literal transaction ambiguities.

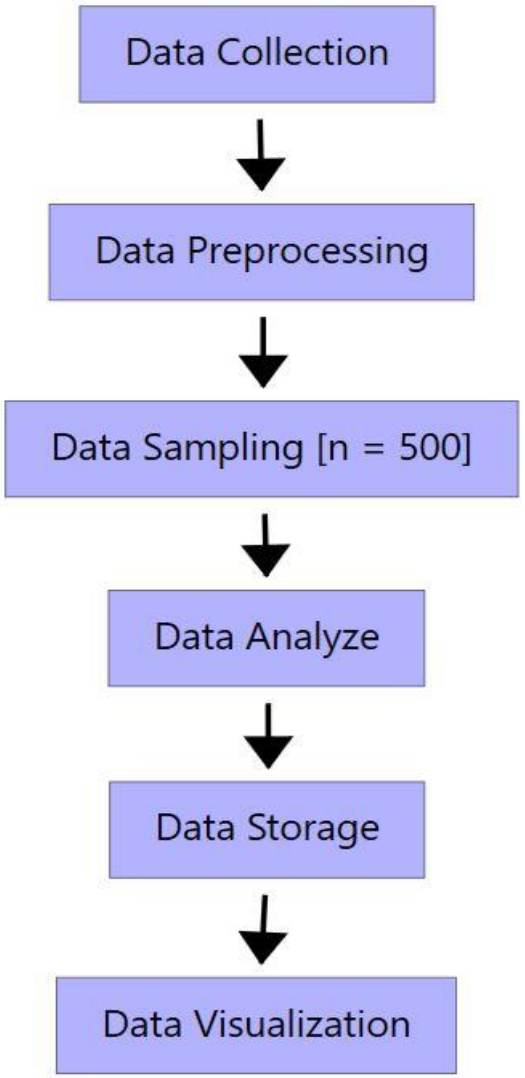


Figure 1 Development Process of the proposed system

This project has accomplished the following objectives as shown in above figure 1:

1. Data Extraction

In this proposed system there are More than 2.2 Lakhs twitter data has been collected and prepared as a dataset for the Date of 2019-01-01 to 2021-12-05 using third-party library Snscrape and twitter's own API tweepy V2 to extract new data. The Tweets related to healthcare worker strikes from Twitter Data have been collected using following list of Keywords#DoctorProtest","DoctorProtest","#DoctorStrikes","DoctorStrikes","#Healthworker Protest","#HealthworkerStrikes","#NurseProtest","#NurseStrikes","#NurseMarch","#Midwif eProtest. While collecting the data, the last collected Date of data will be stored in one text file so to collect data next time this date parameter will be passed to the tweepy API as a parameter (Since) automatically and tweepy will start collecting the tweets from this date onwards. (Refer

Appendix D). So, this method to collect data will be automated as it will pick date automatically from that text file to collect data next time.

2. Data Pre-processing

This dataset contains data such as tweet Content, tweet URL, Date, Tweet Id, User Location, Media URL, Used Hashtags, Number of retweets, Number of Favourites on that post. As a pre-processing of the data, pandas check for duplicates and missing values in the dataset and reduce the size of the dataset rows by removing unnecessary data. Further, these tweets have been pre-processed using one of the trending NLP data Pre-processing pipelines texthero to remove content within brackets and the brackets itself, curly brackets {} and the curly brackets, all diacritics and accents, all digits and replace it with a single space, html tags from the given Pandas Series, all stop words, URLs, Hashtags, Punctuation, Whitespaces.

3. Data Analyse

In this project free version of the libraries has been used such as geopy so it is difficult to analyse large data due to limitation using free versions. So, to analyse the data, randomly 500 rows have been sampled from the dataset to analyse it further. To identify relevant and key terms in social media discussions about the Healthcare worker strikes, context-based natural language processing (NLP) Methods have been used. To Extract the information from that data, the Information Extraction technique of NLP, which is the Named Entity Recognition model by spaCy, has been applied. It's a method for extracting relevant and usable data from an unstructured raw text content. NER locates and categorises entities included in unstructured text into standard categories such as person names, locations, organisations, time expressions, amounts, monetary values, percentages, codes, and so on. To check sentiments of the tweets, a pre-trained model of Huggingface has been used. This data would ideally result in CSV containing the key information extracted of the health worker protest by NLP Methods.

Then to analyse the data further the Count vectorizer has been applied to extract the feature names from the tweet text using the Tfidf vectorizer method of sklearn. The Count Vectorizer returns the number of occurrences per word index. Then this vector of the words is used to get the topics using Topic Modelling technique latent Dirichlet allocation (LDA) (David, et al., 2003), and lastly, Text Summarizer has been applied to summarise all the data by grouping the unique location by the Huggingface Transformer pipeline.

4. Data Consolidation and storage

After data extraction and analysis a CSV file of the dataset has been prepared and this data has been gathered and stored in a database. All required information has been collected from social media and stored in a structured format for further uses. So, for database managing and handling, the SQLite Local database of python has been used. This project has resulted in a database which is updated automatically with each of the above stages daily using the Windows scheduler, and it will be presented online through the webpage. To make every step automated I have created a batch file in the operating system to let OS know to run a python script everyday which will update every step from Data collection to Data storage automatically using the windows scheduler (Refer Appendix B).

5. Webpage Design & Coding

To visualise this data about the Health Protest, this project has been proposed with the visual product as an interactive Web Application of the Django framework containing Graphical views for the data Using table and the Interactive Map. This website has been created using Python programming as a back-end, to manage and retrieve data SQLite has been used, CSS, AJAX, JavaScript, and HTML as a front-end.

However, currently other protest tracker systems set their database manually so this would be a unique approach to have data regarding health workers strikes for further investigation using social media, one of the powerful platforms nowadays and specially Twitter is one of the most popular microblogging platforms.

6. Testing

Testing has been done using new data. The new data from twitter is being collected from the last date of previously extracted data and this new data is processed step by step from data collection to data pre-processing to extract named entity to topic modelling to summarise data to data storage and when Web application runs it will update all analysed data from database to the webpage.

1.3 Structure of report

In this project, the study of automatically extract structured information about the health care professional protests from the unstructured and/or semi-structured text has been performed using Information Extraction technique Named Entity Recognition of NLP to identify the

prevalence of protest, along with its features, essentially who is involved, the issues that are being challenged, Location, Hashtags, sentiment analysis and its Impacts etc.

Chapter 2 shows the Literature about the study of the defined problem.

Chapter 3 focused on the legal, social, and professional issues in developing this system.

Chapter 4 represents the Data collection and qualitative analysis for the proposed system.

Chapter 5 explained the Methodology used to solve the define problem

Chapter 6 visualises the results and testing of the new data.

Chapter 7 shows the development of the software for the defined problem.

Chapter 8 contains the conclusion of study, findings, recommendation, and areas of future work.

2. Literature Review

2.1 Literature Review

According to recent reports, doctor and health-care worker strikes are a global problem that affects both developed and poor countries and have the potential to harm the quality of treatment and the doctor-patient relationship. Strikes are a legal means of breaking a deadlock in collective bargaining when labour negotiations have reached a stalemate (Chima, 2013). Failed employer-employee discussions over fair compensation and working conditions, regulatory challenges, infrastructure limitations in poorer nations, and Healthcare workers (HCW) worries about personal security in the workplace are the main causes of HCW strikes. HCW strikes have a major influence on healthcare delivery, such as cancelled outpatient appointments, hospital admissions, and elective procedures. Except in select circumstances where emergency services were also removed during strikes, there was no clear indication of increased patient mortality during strikes (Chima, 2020).

To compile a complete overview of current surveys, I began by looking at existing studies in the existing system to track the strikes actions and I have found some existing Tracker performing the same task but manually using Hospital database and information from reports and Articles on Health workers Strike action:

After the Myanmar armed forces (known as the Tatmadaw) assumed control of the country on 1 February following a landslide victory by the National League for Democracy party in a general election, mass civil disobedience movement (CDM) protests have erupted across the country. In Myanmar, 109 acts of violence against health care were recorded between February 11 and April 12, 2021.The episodes mentioned are from the dataset Violence Against Health Care in Myanmar Data, which is available on the Humanitarian Data Exchange from 11 February to 12 April 2021 (HDX). Insecurity Insight prepared the map, with support from MapAction, for the Safeguarding Health in Conflict Coalition. Insecurity Insight collated data from multiple public sources, confidential contributions from aid agencies and professional bodies (Insecurity Insight, 2021).

Health professionals in Kenya have gone on strike multiple times in recent years, but the impact on mortality is unknown. They looked at the impact of six health worker strikes in Kilifi,

Kenya, that occurred between 2010 and 2016. They used a negative binomial regression model to daily mortality data from the Kilifi Health and Demographic Surveillance System (KHDSS) to assess the change in mortality during strike periods and the two weeks following strikes The KHDSS updates information on pregnancies, births, deaths, and migrations every four months, and cause of death data is gathered through spoken autopsies. They recorded 1829929 person-years of observation, 6396 deaths, and 128 strike days (median strike duration, 185 days [range 9–42]) between January 1, 2010, and November 30, 2016. In the primary analysis, there was no difference in all-cause mortality during strike times (adjusted rate ratio [RR] 093; 95 percent confidence interval [CI] 081–108; p=034. During the years 2010–16, there were no obvious increases in overall mortality in Kilifi because of health workers' brief strikes. They may have underestimated the effect due to the combined effects of private (and some public) health care during strike periods, a high proportion of out-of-hospital mortality, and a small number of events (Gerald, et al., 2019).

(Giuliano, et al., 2019) Authors analysed the characteristics, frequency, drivers, outcomes, and stakeholders of health workers' strikes in low-income countries for the years 2009 to 2018. For the years 2009 to 2018, they looked at published and grey literature from online sources. They employed four different search methods: (i) utilisation of specialised websites on human resources for health and development; (ii) investigation of major health and social sciences databases; (iii) a customised Google search; and (iv) professional assistance to confirm findings. During that time, they found 116 records reporting on 70 different health worker strikes in 23 low-income countries, totalling 875 days on strike. The year 2018 featured the most events (17), resulting in a loss of 170 working days. Strikes involving multiple professional categories were the most common (32 occurrences), followed by strikes involving solely physicians (22 events). During that time, they found 116 records reporting on 70 different health worker strikes in 23 low-income countries, totalling 875 days on strike. The year 2018 featured the most events (17), resulting in a loss of 170 working days. Strikes involving multiple professional categories were the most common (32 occurrences), followed by strikes involving solely physicians (22 events). Some common characteristics appear to exist in the occurrence of health sector strikes and the players participating in such actions in low-income nations. Future study should concentrate on both individual occurrences and regional patterns to build an evidence base for strike prevention and resolution measures.

I have compiled some literature to find out how data has been collected from twitter to prepare a dataset for the specific topic:

From August 1, 2014, to July 31, 2015, the researchers have gathered Twitter data using 10 terms connected to HPV vaccination. The Twitter Search API was utilised for prospective data collection, and Twitter Firehose was used for retrospective data collection. They have classified a subsample of tweets by hand using a codebook to characterise tweet sentiment and content, then developed classification models to code the complete sample using machine learning processes. They also recorded the words that were most connected with each keyword in the 140-character tweet content. To see if there were any significant differences in tweet characteristics based on sentiment, they employed chi-square tests, analysis of variance, and nonparametric equality of medians. As a result they have concluded that a rising area of public health study is looking at social media to detect health trends and deliver crucial health information. Understanding the substance and implications of discussions on social media about HPV vaccination might help health organisations and health-focused Twitter users have a meaningful exchange of ideas and a major impact on vaccine uptake. This field of study is intrinsically interdisciplinary, and this work contributes to that trend by extending methodology across areas using public health, health communication, and data science approaches. (PM., et al., 2016)

This research shows how the social media data has been mined to evaluate the effect of the specific action globally.

The related example about social media data mining is to evaluate the impact of the COVID-19 epidemic on people around the world. Using social media data, researchers used natural language processing (NLP) and thematic analysis to better understand public perspectives, experiences, and challenges related to the COVID-19 epidemic. To begin, they gathered more than 47 million COVID-19-related comments from Twitter, Facebook, YouTube, and three online discussion forums. Second, they used natural language processing (NLP) techniques to clean and prepare the data for automated key phrase extraction. Third, they used the NLP strategy to extract meaningful key words from over 1 million randomly selected comments, computed a sentiment score for each key phrase, and used a lexicon-based technique to assign sentiment polarity (positive, negative, or neutral) based on the score. Fourth, they categorised or grouped similar negative and positive key phrases into large themes (Oyebode, et al., 2021).

As e-commerce, SaaS solutions, and digital technologies advance, we can clearly observe that sentiment analysis is becoming increasingly popular. There are many uses for sentiment analysis, and many researchers have used it to deduce explanations from Twitter data to investigate an occurrence or address a problem. In this area, there are several interesting projects underway, including (Sarlan, et al., 2014) Over 18 million tweets about new coronavirus were studied by (Soomro., et al., 2020). The tweets were examined to determine if there was a link between public mood and the rise or fall in the number of coronavirus cases.

To extract the information from the Tweets I have used Named Entity recognition (NER) technique using spacy for this project:
Microblogs are possibly the most difficult type of content to handle in terms of Named Entity Recognition (NER) and Information Extraction (IE) in general. To begin with, their brevity (tweets are limited to 140 characters) makes them difficult to comprehend. As a result, ambiguity is a significant issue, as IE techniques cannot simply exploit coreference information. Unlike longer news stories, microblog papers include a small quantity of discourse material, and threaded structure is split among numerous documents, flowing in multiple directions. To address these issues, researchers have concentrated on microblog-specific information extraction techniques (for example, named entity identification for Twitter using CRFs (Alan, et al., 2011), and Wikipedia-based subject and entity disambiguation (van Erp, et al., 2013). Microtext normalisation receives special attention as a means of reducing some of the linguistic noise before part-of-speech labelling and entity recognition (Leon, et al., 2013).
Some alternatives have been offered for Twitter, although they are frequently not openly available. (Alan, et al., 2011) use a pipeline approach to locate named entities, first tokenizing and POS tagging and then utilising topic models to find them.

In this project I have used Topic Modelling of LDA (David, et al., 2003) method to extract the topics from a sentence. Following are some research that have been done using this method:
In the text mining community, topic modelling is gaining a lot of traction. In topic modelling, Latent Dirichlet Allocation (LDA) (David, et al., 2003) is becoming a standard tool. As a result, LDA has been developed in a variety of ways, with several expansions proposed for social networks and social media. In the framework of structural social media data analysis, (Jaffali S., 2020) has offered a summary of social network data analysis, including its basic approaches and applications. They divided social network analysis approaches into two categories:

structural analysis methods (which look at the structure of the social network, such as friendships) and added-content methods (which look at the content that people upload).
(Likhitha, et al., 2019) has presented a comprehensive study of several TM strategies in social media text and described numerous applications, quantitative evaluations of various methods, and numerous datasets that are employed with varied issues in brief content and documents.

2.2 Background Research

To collect the twitter data for this proposed project I have researched many techniques but due to some restrictions and limitations kept by twitter I have used Twitters' third-party library Snscrape to collect historical tweets. Twitter has its own API for Data collection such as Tweepy API v1 and API v2, however all have restrictions on using its API Endpoints. Twitter's standard API allows us to get 7 Days of history Data only. However other API such as API for academic research allows full archive but only 30 Days History Data. Due to this reason, I have used Snscrape to extract the historical tweets. snscrape is a library that allows anyone to scrape tweets without having to use their own API credentials. It features extensive search options that enable very customised searches and can return thousands of tweets in seconds. There is currently a shortage of documentation, particularly regarding scraping tweets by location (Scott, 2020). However, to get the new data I have used Twitter's API tweepy V2 standard version, which has many restrictions to collect the data such as we can extract past 7 days tweets only but to get historical data, I have used Snscrape as mentioned above. Tweepy is a Python package that allows us to interact with the Twitter API. It's ideal for simple automation and Twitter bot creation. Also, to use this Twitter API's it requires the Developer account and API authentication Keys such as consumer key, consumer secret, access token, access token secret. However, it's easy to get That but there are still use case restrictions.
snscrape is a scraper for social media platforms (SNS). It scrapes information such as user profiles, hashtags, and searches and returns the results, such as relevant postings.
Currently, the following services are supported:

Facebook: user profiles, groups, and communities (aka visitor posts)
Instagram: user profiles, hashtags, and locations
Reddit: users, subreddits, and searches (via Pushshift)
Telegram: channels
Twitter: users, user profiles, hashtags, searches, tweets (single or surrounding thread), list posts, and trends

VKontakte: user profiles
Weibo (Sina Weibo): user profiles (PyPI, 2021)

To pre-process the unstructured data, I have analysed many methods such as NLTK's own Pre-processing methods step by step; however, for this project I have used the Texthero library as well as NLTK's library.
Texthero is a Python framework or toolkit for quickly and easily working with text-based datasets. It is simple to learn and is designed to be used on top of Pandas. It is widely reported and possesses similar expressiveness and ferocity as Pandas. It is modern and imagined for multi-decade software engineers with little, if any, etymological information.
Texthero requires only a few lines of code to preprocess text data, map it into vectors, and visualise the resulting vector space. It's open-source, free, and well-documented.

After Cleaning or pre-processing of data, the next step is to Extract the information from that data. I have applied the Information Extraction technique of NLP which is Named Entity Recognition. The task of information extraction (IE) is to automatically extract structured information from unstructured and/or semi-structured machine-readable texts. The first step toward Information Retrieval is named entity recognition. It's a method for extracting relevant and usable data from an unstructured raw text content. NER locates and categorises entities included in unstructured text into standard categories such as person names, locations, organisations, time expressions, amounts, monetary values, percentages, codes, and so on. SpaCy includes a statistical entity recognition system that assigns labels to tokens in continuous spans. SpaCy is a Python module for advanced language processing that is free. It's used to create information extraction systems or to prepare text for deep learning. Tokenization, Parts-of-Speech (PoS) Tagging, Text Classification, and Named Entity Recognition are some of the functionalities offered by SpaCy (Akshay, 2021).

Then to analyse the data more I have applied the Countvectorizer to extract the feature names from the tweet text using Tfidf vectorizer method of sklearn. The Count Vectorizer returns the number of occurrences per word index. After having features names, I have applied an unsupervised Topic modelling method LDA (David, et al., 2003) to get the most frequent topics from the vectors and summaries the same topic using summarizer model of NLP by its unique location to get all tweets related to same topics and location.

After having a vector of the tweet text, I have applied a Topic modelling LDA method to get the topics from the tweet text. In contrast to supervised learning techniques, topic modelling is a type of unsupervised machine learning. This means that we don't need to provide labels (that is, subject names that match each document) during training to train the model. This not only aids in the discovery of potentially intriguing topics, but it also saves the amount of time spent manually tagging texts. On the other hand, evaluating the output of a topic model can be a lot more difficult (Subhashini & Grishma, 2021). This twitter data is unstructured, and it is difficult to get the labels from the unstructured data so this Topic modelling LDA method would be a great approach for this project to get the topic from the sentences.

Then to visualise all analysed data I have made an interactive web application using python web application framework Django, AJAX and to store all data I have used python's local database SQLite.

2.3 Existing System

While there are already existing tools that are like this proposed system, it becomes necessary to highlight what the existing tools do as compared to this proposed system; however they are collecting data traditionally from the news sources and other data sources.

Insecurity Insight: Insecurity Insight prepared the map, with support from Map Action, for the Safeguarding Health in Conflict Coalition. Insecurity Insight collated data from multiple public sources, confidential contributions from aid agencies and professional bodies (Insecurity Insight, 2019).

Crowd Counting Consortium (CCC): The Crowd Counting Consortium (CCC) collects publicly available data on political crowds reported in the United States, including marches, protests, strikes, demonstrations, riots, and other actions (Tommy, n.d.).

Carnegie: Carnegie has developed the Global Protest Tracker to analyse and compare the triggers, motivations, and other aspects of many of the most significant anti-government protests since 2017. Designed for researchers, decisionmakers, and journalists, this comprehensive resource helps illustrate how protests impact today's global politics (Carnegie Endowment, 2022).

3. Analysis of the system

3.1 Legal, Social, Ethical and Professional issues

This project is not eligible for Legal, Social and Ethical approval. This project contains publicly available data and is not dealing with:

- patients,
- people with vulnerabilities,
- children / minors or
- animals, etc.

However, this project required some permission and access keys from twitter to access their data using the developer access account which I have already applied for and got approval to access my account.

4. Data collection and qualitative analysis

4.1 Extraction and Filtering Tweets

In this proposed system there are More than 2.2 Lakhs twitter data has been collected and prepared as a dataset for the Date of 2019-01-01 to 2021-12-05 using third-party API Snscrape. The API can be configured to search Twitter feeds for a certain term or hashtag (#). When Twitter is mined on a large scale to infer the thoughts of the masses on a specific issue, such as understanding people's opinions on a specific prominent person or current events, searching through hashtags can be useful. One of the most significant disadvantages of hashtag, or word, search is the extremely high noise-to-signal ratio. Such inquiries can return millions of tweets. While large amounts of data might be beneficial for inferring public trends, manually classifying such data can be time-consuming. The Tweet post related to healthcare worker strikes from Twitter Data has been collected Using Following list of Keywords#DoctorProtest","DoctorProtest","#DoctorStrikes","DoctorStrikes","#Healthworker Protest","#HealthworkerStrikes","#NurseProtest","#NurseStrikes","#NurseMarch","#Midwif eProtest. However, to extract the new data I have used Twitter's API tweepy V2 standard version, which has many restrictions to collect the data such as we can extract past 7 days tweets only but to get historical data, I have used Snscrape as mentioned above. Tweepy is a Python package that allows us to interact with the Twitter API. It's ideal for simple automation and Twitter bot creation. Also, to use this Twitter API's it requires the Developer account and API authentication Keys such as consumer key, consumer secret, access token, access token secret. However, it's easy to get That but there are still use case restrictions.

This API will pick the date from the data frame's last time extracted date and save it into one text file. So, this method to collect data will be automated as it will pick the date automatically from that text file to collect data next time. Even though the gathered tweets could be in a variety of languages, our algorithm exclusively analyses English tweets. The snscrape package allows us to get tweets in each language. For those tweets with full texts collected in English, a language parameter value such as 'en' is assigned. We also noticed a high number of duplicate tweets when people like to retweet the same text of tweets for other people (called Re-tweet). To avoid repetition, we employed regular expression algorithms based on NLP to look for and remove hyperlinks in the content of tweets. Following that, the duplicate tweets were removed because they appeared to provide no new information to the dataset, proving to be

computationally inefficient. When duplicate tweets are removed from the dataset, the dataset becomes much more meaningful.

4.2 Dataset Description

The below table shows the data field description of the dataset.

Data fields	Description
Datetime	Shows the tweet posted date and time
Tweet Id	Shows the tweet's unique Id used to identify the tweets
Text	Tweet content
Location	Tweet's Users Location
Media	Shows the link of any media (Image or video) attached with post
Tweet URL	Shows the tweet URL
Hashtags	Shows any Hashtags used with tweet post
Retweet Counts	Shows the number of retweets of that post
Like counts	Shows the number of Favourites on that post

Table 1 Data field description of the twitter dataset

The below figures 2 show the raw data collected from twitter using above mentioned details.

	Datetime	Tweet Id	Text	location	Media	
0	2021-11-29 13:17:12+00:00	1465308866808147968	#Karnataka resident #doctors and interns have ...	Hyderabad, India	NaN	https://twitter.com/YTHISNEWS/stat
1	2021-06-06 04:10:25+00:00	1401390999411171331	#GWALIOR - #MadhyaPradesh #doctorprotest \nMP ...	Noida, Uttar Pradesh	[Video(thumbnailUrl='https://pbs.twimg.com/ext...	https://twitter.com/Amritsantlar
2	2021-06-01 11:37:01+00:00	1399691450921603074	@FT @FinancialTimes i think its time too chang...	Ratlam Mp	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/imtusharmeen
3	2021-05-29 16:29:17+00:00	1398677839277289475	@mr_mayank #FarmersProtest is going on since m...	Omnipresent	NaN	https://twitter.com/peppersprae/s
4	2021-03-01 11:54:43+00:00	1366356219418484739	Demanding pay revision, arrears due since 2016...	New Delhi, Delhi	NaN	https://twitter.com/medicaldialo
...	...	...	...	...	...	
229145	2019-12-10 13:43:24+00:00	1204396194610532354	#nursestrikes fully support the nurses having ...	northern ireland	NaN	https://twitter.com/MWsunrunner6
229146	2019-12-06 06:57:58+00:00	1202844613654073344	"Our health service is collapsing as we speak"...	Belfast	NaN	https://twitter.com/BelfastLive/s
229147	2019-01-30 17:33:19+00:00	1090664267257397248	In solidarity with the #Nursestrikes today, th...	Dublin City, Ireland	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/Daviddoylea
229148	2021-11-21 15:00:41+00:00	1462435806388305920	#MidwifeProtest Cambridge https://t.co/b7QHbvQQup	Cambridge, UK	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/bwhitecambs/s
229149	2021-11-20 08:14:09+00:00	1461971111911071745	"This is the most galvanised I've ever seen mi...	United Kingdom	NaN	https://twitter.com/FiLiA_charit

229150 rows × 9 columns

Figure 2 Raw Dataset collected from Twitter

4.3 Exploratory Data Analysis

Exploratory Data Analysis is the crucial process of using descriptive statistics and graphical representations to undertake initial investigations on data to uncover patterns, spot anomalies, test hypotheses, and verify assumptions.

```
In [10]: df.columns

Out[10]: Index(['Datetime', 'Tweet Id', 'Text', 'location', 'Media', 'TweetUrl',
                'Hashtags', 'Retweets Counts', 'Like Counts'],
               dtype='object')
```

Figure 3 Data columns of the twitter dataset

Daily Tweets collected for the period of 2019-01-01 to 2021-12-05:

	date	counts
0	2019-01-01	84
1	2019-01-02	235
2	2019-01-03	273
3	2019-01-04	340
4	2019-01-05	199
...	...	...
1064	2021-11-30	142
1065	2021-12-01	112
1066	2021-12-02	130
1067	2021-12-03	218
1068	2021-12-04	127

1069 rows × 2 columns

Figure 4 Data showing Daily tweets collected for the period

Figure 5 shows the line chart of the daily tweets and figure 6 shows the Line chart of the monthly tweets for the period of 2019 to 2021 of the health worker protest. It can be noticed from the below figures that the most tweets were posted in month of June 2019 about the Healthcare protest.

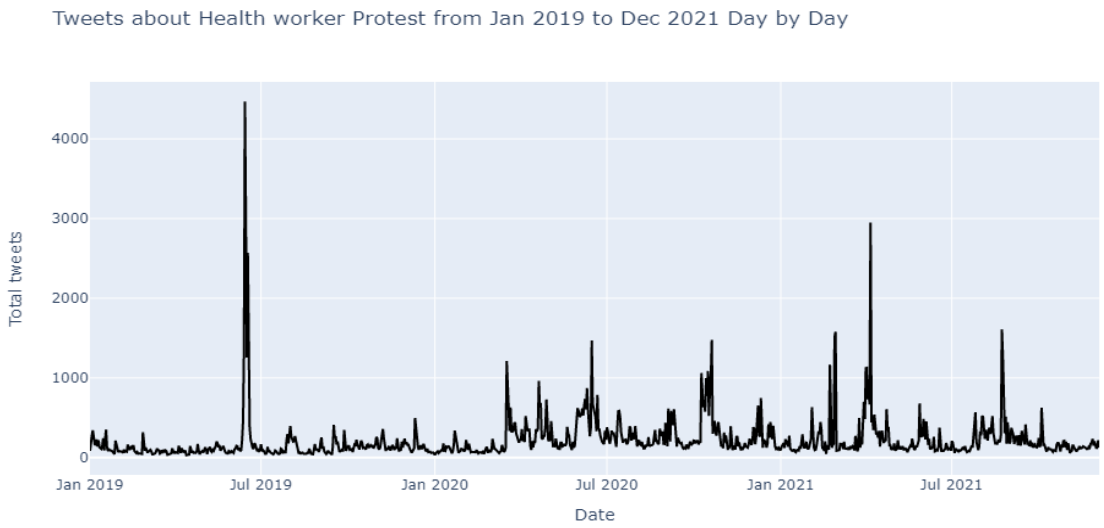


Figure 5 Line chart showing Daily tweets collected for the period

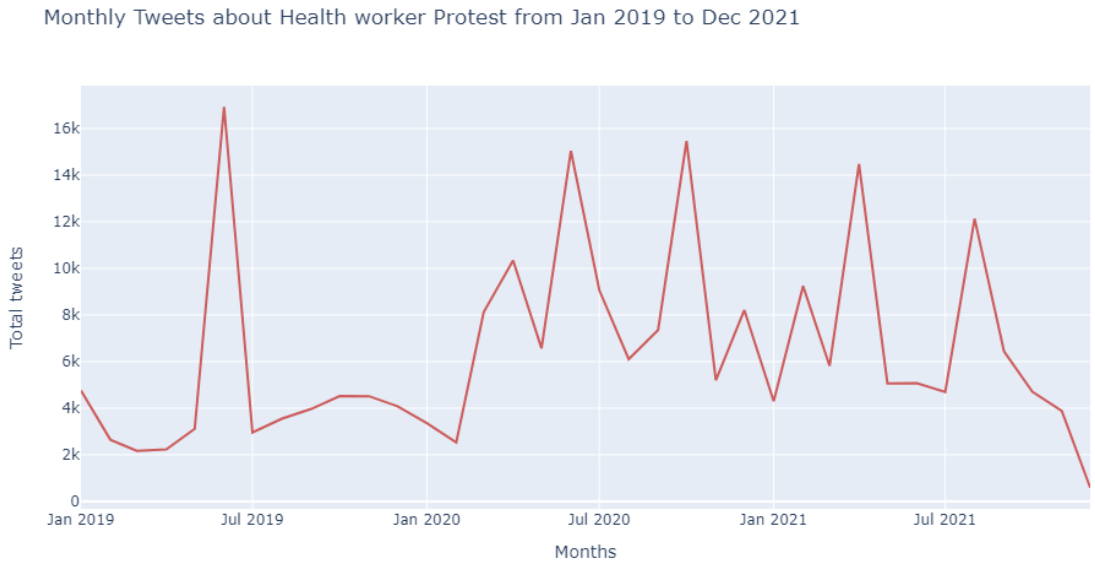


Figure 6 Line chart showing Monthly tweets collected for the period

Word Cloud of Twitter Data: Wordcloud is a cloud of words, as the name implies. It is a graphical representation of the words in a corpus. Each word is depicted in terms of its significance in the context or frequency. The size of the words is determined by the number of times they appear. The more times a term appears in a corpus, the larger it becomes (Suresh, 2020).

Word Cloud of Tweets before pre-processing of the tweet data:

The below figure 7 shows the word cloud of tweet text contains 50 most frequent words such as https, strike, protest need, doctor, Resident Doctor, junior doctor, nurse, country etc for the given period before pre-processing of the data. We can see clearly from the below figure that the word cloud contains http words as well which is no use for our data analysis so for that pre-processing was applied on data.

Figure 7 Word Cloud of Tweet before pre-processing

Word Cloud of Tweets after pre-processing of the tweet data:

We can see from the below figure 8 that the most frequent words used in tweet posts are Doctor Strike, Protest, Junior doctor, resid doctor, indefinite strike etc.

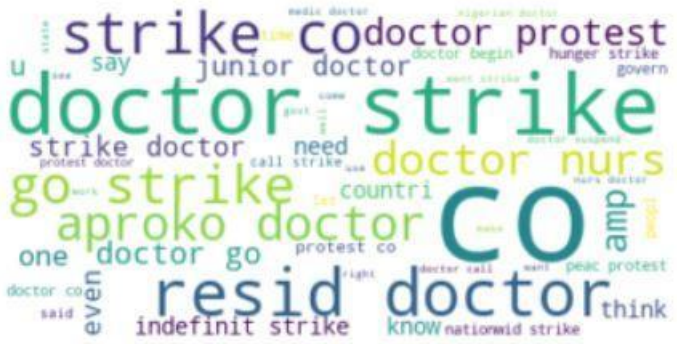


Figure 8 Word Cloud of Tweet after pre-processing

Word Cloud of Hashtags:

We can see from figure 9 that most frequently used hashtags used during posting the strikes post are #WhatsHappeningInMyanmar, #Coronavirus, #covid19 etc.

Figure 9 Word Cloud of Hashtags

Word Cloud of Location:

However, the below figure.8 shows the location of the most frequent tweets sent by that user. We can clearly see that India, Nigeria, United states, United Kingdom, Kenya etc. are the locations where users posted tweets frequently.

Figure 10 Word Cloud of Location

4.3.1 Missing Data

Data in the real world is untidy and frequently contains many missing values. Any analysis is only as good as the data it is based on. When no value is available in one or more of an individual's variables, missing data appears. The statistical power of the study can be reduced because of missing data, which can have an impact on the conclusions' validity (Suvarna, 2021). Here, Location is important for our data to show the detail about the protest data. So, if the tweets do not have location, those rows need to be removed to have only relevant data. Here, we cannot remove the Media and hashtags column's row having NA values because generally not every tweet contains media and hashtags. So, if we remove those rows, there might be chances to lose some useful information from the tweet text.

Column name	Counts
Datetime	0
Tweet Id	0
Text	0
location	**62219**
Media	**183870**
Tweet URL	0
Hashtags	**170160**
Retweets Counts	0
Total Missing Values:	**4,16,249**

Table 2 Missing values of the collected data

5. Methodology

5.1 Data Pre-processing

Pre-processing data is the first and most important stage in every data science project or problem. Any Natural Language Processing, Computer Vision, deep learning, or machine learning task requires pre-processing of the obtained data. Various preparation procedures must be used depending on the type of dataset. The first step in the Natural Language Processing pipeline is pre-processing. Because of noisy or ambiguous data gathered or collected from various sources, pre-processing is becoming increasingly important in NLP. We're using Twitter's unstructured data for our project. In their tweets, individuals can be seen using abbreviated forms, emojis, misspellings, and so on. we should not feed raw data into models without first pre-processing it since text preparation directly increases the model's performance. The constructed models will not learn the true relevance of the input if we feed it without using any text pre-treatment approaches. When we feed raw data into models without using any pre-processing approaches, the models can become confused and produce unpredictable outcomes (harmila, 2020).

Remove Duplicate data: Before pre-processing of the data first I have checked for any duplicates data from more than 2.2 Lakhs tweets, and I have found that there is total 3973 duplicate values in the collected data as shown in table 3. So, I have removed duplicate data from the dataset.

Duplicated Values	
Collected rows	**Duplicates rows**
229150	3973

Table 3 Duplicated values of the collected Twitter data

Text pre-processing techniques of the Text hero library used in this project to clean the data:

Lower Case: Because the machine treats lowercase and uppercase differently, it is easy for a computer to read the words if the text is in the same case. Words like Ball and ball, for example,

are processed differently by machines. To avoid such issues, we must make the text in the same case, with lower case being the most preferable instance (Raghav, 2021).

Remove punctuations: The removal of punctuations is another text processing approach. There are a total of 32 primary punctuations that must be addressed. We may use a regular expression and the string module to replace any punctuation in text with an empty string. The string module provides us with 32 punctuations, which are given below (Raghav, 2021).

```
'!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~'
```

Remove Stop words: Stop words are the most frequently occurring words in a text that offer no useful information. Stop words such as they, there, this, where, and others are examples of stop words (Raghav, 2021). but for this project if stop words get removed then the sentence of meaning gets changed so I have not removed stop words.

Remove Extra Spaces: We need to control this problem since most text data has additional spaces or more than one space is left between the text while applying the preceding preparation processes. This problem is well-solved by the regular expression library (Raghav, 2021).

Also, I have removed diacritics, all types of brackets, URLs, HTML tags to make it clean data use for further analysis. The figure 11 shows the total data after removing duplicates and missing values from the raw dataset and figure 12 shows the clean text after applying pre-processing technique.

Figure 11 Data after removing duplicates and missing values from collected tweets data

```
print(tweets_df['clean_text'])
```

```
0         gwalior    madhyapradesh  doctorprotest mp ju...
1        i think its time too change your editor before...
2        after thousands of farmers are on roads protes...
3        doctors don t want salute they want ppe  when ...
4        why barbaric police action is the solution to ...
                              ...
15367     s new  pilot  healthworkerprotest project is ...
15368    'we're beyond angered'  fed up nurses file law...
15369    this is the first i've seen heard of a  nursep...
15370    in solidarity with the  nursestrikes today  th...
15371                             midwifeprotest cambridge
Name: clean_text, Length: 15372, dtype: object
```

Figure 12 Tweet Text after Pre-processing

5.2 Data Sampling

The process of selecting a random number of records from the dataset is known as Data sampling. In this project, due to some limitation using libraries such as python's package geopy (used to identify and track the location). This service is based on donated servers and has a use limitation in the service. As a result, they ask that we limit our use and follow their usage policy. 1 request per second is the absolute maximum. Bulk geocoding of vast amounts of data is not recommended in general. They recommend that if we have regular geocoding assignments, they suggest, look into the options listed below. Smaller one-time bulk tasks may be permitted if the following extra guidelines are followed:

- Confine your queries to a single thread.
- Scripts can only be run on one machine; no distributed scripts are allowed (including multiple Amazon EC2 instances or similar)
- You must cache the results on your end. Clients who send the same query repeatedly may be flagged as defective and blocked (osmfoundation, n.d.).

The text summarizer by Huggingface Transformer (run on top of PyTorch and TensorFlow) requires a lot of memory because PyTorch or TensorFlow libraries need to be installed to use this transformer. Due to this reason, I have randomly sampled the dataset as 500 rows using below code. See figure 13 for the new sampled dataset.

```
tweets_df=tweets_df.sample(n=500)
tweets_df=tweets_df.reset_index(drop=True)
```

	Datetime	Tweet Id	Text	location	Media	Twee
3211	2021-02-01 05:41:35+00:00	1356115457799979009	Yes still the protest continues\nUntill we get...	srivilliputtur	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/ganeshjeya8/status/13561
8679	2021-05-31 11:54:16+00:00	1399333403695087620	Madhya Pradesh | Junior doctors go on strike a...	New Delhi, Delhi	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/WeForNews/status/1399333
2149	2021-02-27 08:11:28+00:00	1365575261496905735	27.2.2021\nA young medical doctor Myat Thuzar ...	Myanmar	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/NguNway/status/136557526
10934	2021-02-28 07:16:52+00:00	1365923908709453827	Medical universities strike in Yankin this mor...	United States	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/loveforhrj323/status/136
6110	2019-12-11 10:17:41+00:00	1204706812836102144	12 patients die and 25 doctors severely injure...	Pakistan/ Mohmand	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/M_Adnan_khan2/status/120
...	...	...	...	...	...	
3772	2020-10-20 10:58:53+00:00	1318506973667729408	@aproko_doctor Now that the govt of Lagos have...	Lagos	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/Aizekworld/status/131850
12731	2020-07-03 10:51:49+00:00	1279004916586106881	A doctors' strike in Sierra Leone has left Cov...	Zimbabwe	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/263Chat/status/127900491
2191	2021-02-27 07:58:55+00:00	1365572104276631555	Myanmar Military coup is terrorising the peopl...	Myanmar	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/kaykhaing87/status/136557
12934	2020-06-11 08:46:43+00:00	1271000899255595011	Telangana Junior Doctors Association says they...	New Delhi	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/timesofindia/status/1271
1522	2021-03-27 04:09:09+00:00	1375661142643728386	" White Coat Protest " \n\nMedical students, D...	Myanmar	[Photo(previewUrl='https://pbs.twimg.com/media...	https://twitter.com/freesia_tt/status/137566

500 rows × 11 columns

Figure 13 Tweet Dataset after randomly sampled(n=500)

5.2 Methods used to analyse and extract the data

The figure 14 shows the pre-task applied on the data before analysing the data as mentioned in previous chapters.

Figure 14 Steps for Twitter Data analysis (Amandeep & Rajinder, 2019)

5.2.1 Named Entity Recognition using spacy

The first step toward Information Retrieval is named entity recognition. It's a method for extracting relevant and usable data from an unstructured raw text content. NER locates and categories identified entities included in unstructured text into standard categories such as person names, locations, organisations, time expressions, amounts, monetary values, percentages, codes, and so on. SpaCy includes a statistical entity recognition system that assigns labels to tokens in continuous spans. SpaCy is a Python module for advanced language

processing that is free source. It's used to create information extraction systems or to prepare text for deep learning. Tokenization, Parts-of-Speech (PoS) Tagging, Text Classification, and Named Entity Recognition are some of the functionalities offered by SpaCy (Akshay, 2021). Below is the command to install the spacy library.

```
Pip install SpaCypython
Python -m SpaCy download ja_core_news_md
```

TYPE	DESCRIPTION
PERSON	People, including fictional
NORP	Nationalities or religious or political groups
FACILITY	Buildings, airports, highways, bridges, etc
ORG	Companies, agencies, institutions, etc
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
PRODUCT	Objects, vehicles, foods, etc (Not services)
EVENT	Named hurricanes, battles, wars, sports events, etc
WORK_OF_ART	Titles of books, songs, etc
LAW	Named documents made into laws
LANGUAGE	Any named language
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit
QUANTITY	Measurements, as of weight or distance
ORDINAL	"first", "second", etc
CARDINAL	Numerals that do not fall under another type

Figure 15 Summary of spaCy's entity types

The figure 15 shows the summary of the spaCy's entity types. NER can extract all these entities from text as shown in above figure.

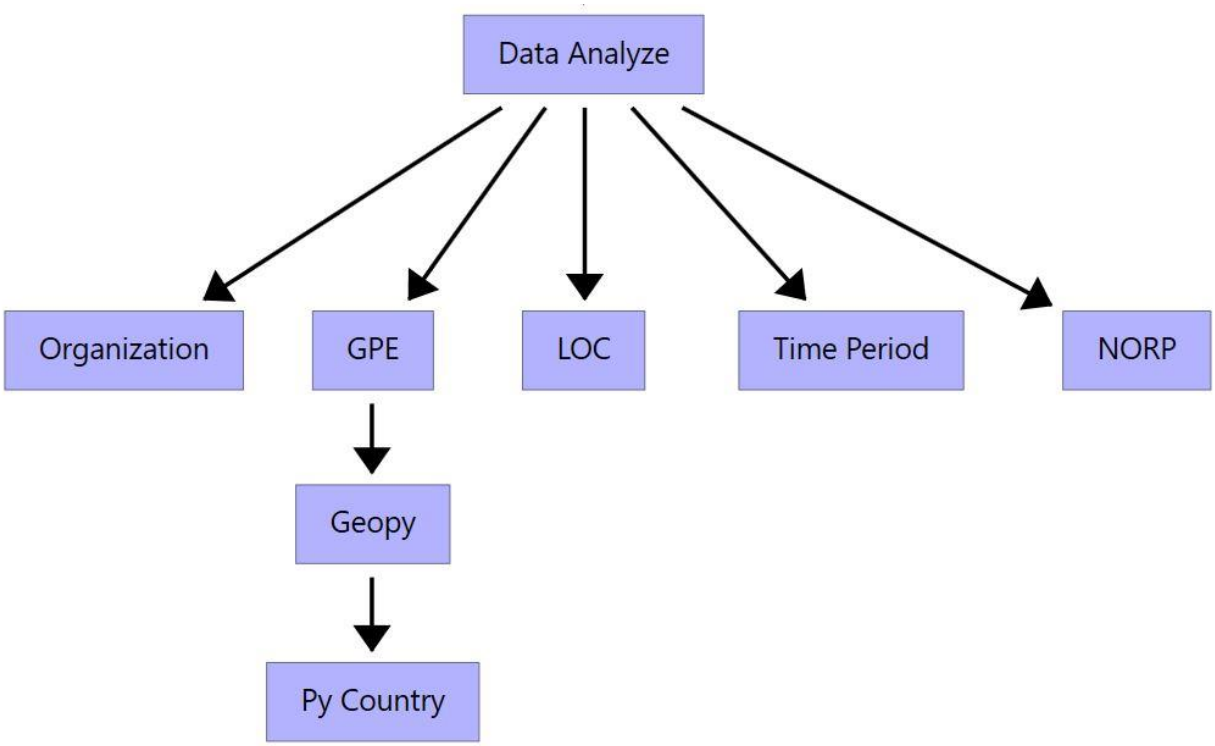


Figure 16 Process of data analysis

```
df.columns.unique
```

```
<bound method Index.unique of Index(['Tweet Id', 'Text', 'location', 'Media', 'TweetUrl', 'Hashtags',
       'Retweets Counts', 'Like Counts', 'clean_text', 'clean_text1',
       'Organization', 'Protest Type', 'NORP', 'Protest-Size', 'GPE', 'LOC',
       'NEW_DATE', 'name', 'geo_country', 'Country', 'Country_Code',
       'sentiment', 'topics'],
      dtype='object')>
```

Figure 17 New columns after extracting the entity

For this project, Location, person name, any NORP involved (Nationalities or religious or political groups), Date, Organization, GPE (Countries, Cities, States) and LOC (Non-GPE location, mountains ranges, bodies of water) entities have been extracted as shown in figure 16. While posting tweets in twitter users mention any kind of location which may be not well-known so using LOC entity it can extract any small places or local area. Moreover, Figure 17 depicts the rows of the dataset after analysing the data. In which some columns are extra to analyse the data.

As there are hundreds or thousands or more than that tweet related to same topic some are useful and some might be not so, it is difficult to find out tweets which contain useful information about the topic or protest. To filter and normalize the data and removed the unnecessary tweets from the dataset location entity has been made as target feature. If the location entity extracts the location from the tweet text, then this tweet is classified as useful. If there is no location found in the tweet, then the row has been dropped from the data as the

location is the key feature for this project. See the below figures for extracted results from the NER model and refer APPENDIX C for code implementation.

NER Extract Organisation name from cleaned tweet text:

```
print(tweets_df[["clean_text",'Organization']].tail(5))
[13]  ✓ 0.6s
...                                                          clean_text  \
Datetime
2021-02-13 03:18:03+00:00  they are kidnapping doctors  nurses  police  s...
2019-11-08 06:30:22+00:00   karnataka   doctors in  bengaluru sit on stri...
2020-06-12 18:53:10+00:00   covid19  doctorwholockdown florida had the hi...
2020-07-26 05:48:59+00:00  nearly 300 junior doctors and interns have bee...
2020-10-20 10:58:53+00:00   doctor now that the govt of lagos have delaye...

                                Organization
Datetime
2021-02-13 03:18:03+00:00                NaN
2019-11-08 06:30:22+00:00                NaN
2020-06-12 18:53:10+00:00                NaN
2020-07-26 05:48:59+00:00  ['mgm hospital']
2020-10-20 10:58:53+00:00                NaN
```

Figure 18 NER Extract Organisation name from cleaned tweet text

NER Extract Protest type using EVENT entity name from cleaned tweet text:

```
print(tweets_df[["clean_text",'Protest Type']].head(5))
 ✓ 0.1s
                                                          clean_text  \
Datetime
2019-09-17 09:02:00+00:00  first broadcast in the uk  otd in 1966 episode...
2020-08-25 00:24:08+00:00  doctor's suicide  mysuru protests may slow dow...
2019-10-29 14:50:00+00:00  families across the uk are considering a hunge...
2019-06-14 06:23:27+00:00  resident doctors association in  aiims delhi a...
2021-04-03 14:45:34+00:00  via   nigerians in london are back at "abuja h...

                          Protest Type
Datetime
2019-09-17 09:02:00+00:00          NaN
2020-08-25 00:24:08+00:00          NaN
2019-10-29 14:50:00+00:00          NaN
2019-06-14 06:23:27+00:00          NaN
2021-04-03 14:45:34+00:00          NaN
```

Figure 19 NER Extract Protest type from cleaned tweet text

NER Extract NORP entity from cleaned tweet text:

```
print(tweets_df[["clean_text",'NORP']].head(5))
✓ 0.5s
```

```
                                                          clean_text  \
Datetime
2019-09-17 09:02:00+00:00  first broadcast in the uk  otd in 1966 episode...
2020-08-25 00:24:08+00:00  doctor's suicide  mysuru protests may slow dow...
2019-10-29 14:50:00+00:00  families across the uk are considering a hunge...
2019-06-14 06:23:27+00:00  resident doctors association in  aiims delhi a...
2021-04-03 14:45:34+00:00  via   nigerians in london are back at "abuja h...

                                          NORP
Datetime
2019-09-17 09:02:00+00:00                  NaN
2020-08-25 00:24:08+00:00                  NaN
2019-10-29 14:50:00+00:00                  NaN
2019-06-14 06:23:27+00:00                  NaN
2021-04-03 14:45:34+00:00  ['nigerian', 'nigerians']
```

Figure 20 NER Extract NORP entity from cleaned tweet text

NER Extract Protest Size using Quantity entity from cleaned tweet text:

```
print(tweets_df[["clean_text",'Protest Type']].head(5))
✓ 0.1s
```

```
                                                          clean_text  \
Datetime
2019-09-17 09:02:00+00:00  first broadcast in the uk  otd in 1966 episode...
2020-08-25 00:24:08+00:00  doctor's suicide  mysuru protests may slow dow...
2019-10-29 14:50:00+00:00  families across the uk are considering a hunge...
2019-06-14 06:23:27+00:00  resident doctors association in  aiims delhi a...
2021-04-03 14:45:34+00:00  via   nigerians in london are back at "abuja h...

                           Protest Type
Datetime
2019-09-17 09:02:00+00:00          NaN
2020-08-25 00:24:08+00:00          NaN
2019-10-29 14:50:00+00:00          NaN
2019-06-14 06:23:27+00:00          NaN
2021-04-03 14:45:34+00:00          NaN
```

Figure 21 NER Extract Protest Size using Quantity entity from cleaned tweet text

NER Extract Health worker Involved using name entity from cleaned tweet text:

```
print(tweets_df[["clean_text",'name']].head(5))
✓ 0.3s
                                                          clean_text  \
Datetime
2019-09-17 09:02:00+00:00  first broadcast in the uk  otd in 1966 episode...
2020-08-25 00:24:08+00:00  doctor's suicide  mysuru protests may slow dow...
2019-10-29 14:50:00+00:00  families across the uk are considering a hunge...
2019-06-14 06:23:27+00:00  resident doctors association in  aiims delhi a...
2021-04-03 14:45:34+00:00  via   nigerians in london are back at "abuja h...

                            name
Datetime
2019-09-17 09:02:00+00:00  doctor
2020-08-25 00:24:08+00:00  doctor
2019-10-29 14:50:00+00:00  doctor
2019-06-14 06:23:27+00:00  doctor
2021-04-03 14:45:34+00:00  doctor
```

Figure 22 NER Extract Health worker Involved using name entity from cleaned tweet text

NER Extract Date entity from cleaned tweet text:

```
print(tweets_df[["clean_text",'NEW_DATE']].head(5))
✓ 0.3s
                                                          clean_text  \
Datetime
2019-09-17 09:02:00+00:00  first broadcast in the uk  otd in 1966 episode...
2020-08-25 00:24:08+00:00  doctor's suicide  mysuru protests may slow dow...
2019-10-29 14:50:00+00:00  families across the uk are considering a hunge...
2019-06-14 06:23:27+00:00  resident doctors association in  aiims delhi a...
2021-04-03 14:45:34+00:00  via   nigerians in london are back at "abuja h...

                           NEW_DATE
Datetime
2019-09-17 09:02:00+00:00  ['1966']
2020-08-25 00:24:08+00:00       NaN
2019-10-29 14:50:00+00:00       NaN
2019-06-14 06:23:27+00:00       NaN
2021-04-03 14:45:34+00:00       NaN
```

Figure 23 NER Extract Date entity from cleaned tweet text

NER Extract City, country, state using GPE entity from cleaned tweet text:

```
print(tweets_df[["clean_text",'GPE']].head(5))
✓ 0.4s
```

```
                                                                clean_text  \
Datetime
2019-09-17 09:02:00+00:00  first broadcast in the uk  otd in 1966 episode...
2020-08-25 00:24:08+00:00  doctor's suicide  mysuru protests may slow dow...
2019-10-29 14:50:00+00:00  families across the uk are considering a hunge...
2019-06-14 06:23:27+00:00  resident doctors association in  aiims delhi a...
2021-04-03 14:45:34+00:00  via   nigerians in london are back at "abuja h...

                                    GPE
Datetime
2019-09-17 09:02:00+00:00           ['uk']
2020-08-25 00:24:08+00:00    ['karnataka']
2019-10-29 14:50:00+00:00           ['uk']
2019-06-14 06:23:27+00:00        ['delhi']
2021-04-03 14:45:34+00:00  ['london', 'uk']
```

Figure 24 NER Extract City, country, state using GPE entity from cleaned tweet text

NER Extract Local address using LOC entity from cleaned tweet text:

```
print(tweets_df[["clean_text",'LOC']].head(5))
✓ 0.4s
```

```
                                                                clean_text  \
Datetime
2021-02-13 03:18:03+00:00  they are kidnapping doctors  nurses  police  s...
2019-11-08 06:30:22+00:00   karnataka   doctors in  bengaluru sit on stri...
2020-06-12 18:53:10+00:00   covid19  doctorwholockdown florida had the hi...
2020-07-26 05:48:59+00:00  nearly 300 junior doctors and interns have bee...
2020-10-20 10:58:53+00:00   doctor now that the govt of lagos have delaye...

                           LOC
Datetime
2021-02-13 03:18:03+00:00  NaN
2019-11-08 06:30:22+00:00  NaN
2020-06-12 18:53:10+00:00  NaN
2020-07-26 05:48:59+00:00  NaN
2020-10-20 10:58:53+00:00  NaN
```

Figure 25 NER Extract Local address using LOC entity from cleaned tweet text

5.2.2 Country name and code Extraction using geopy and pyCountry converter:

The location users are mentioning in tweet text might be local or not well-known or might be any kind of small address. So, to get the exact country name from the location geopy library has been applied to extract the full address of that location. Further, from this full address pycountry library has been applied to get the country name from that address and finally from that country name country code has been extracted using pycountry-converter as the data has been located using country code on map. See figure 26 for result. However, sometime both geopy and pyCountry does not generate the address and country name and generate null value so again the rows will drop if it contains null value.

```
print(df[["GPE",'geo_country','Country','Country_Code']].head(5))
```

```
                                    GPE  \
Datetime
2019-09-17 09:02:00+00:00           ['uk']
2020-08-25 00:24:08+00:00     ['karnataka']
2019-10-29 14:50:00+00:00           ['uk']
2019-06-14 06:23:27+00:00        ['delhi']
2021-04-03 14:45:34+00:00  ['london', 'uk']

                                                      geo_country  \
Datetime
2019-09-17 09:02:00+00:00                                United Kingdom
2020-08-25 00:24:08+00:00                              Karnataka, India
2019-10-29 14:50:00+00:00                                United Kingdom
2019-06-14 06:23:27+00:00  Delhi, Kotwali Tehsil, Central Delhi, Delhi, 1...
2021-04-03 14:45:34+00:00  London, Greater London, England, SW1A 2DX, Uni...

                             Country Country_Code
Datetime
2019-09-17 09:02:00+00:00  United Kingdom           GB
2020-08-25 00:24:08+00:00           India           IN
2019-10-29 14:50:00+00:00  United Kingdom           GB
2019-06-14 06:23:27+00:00           India           IN
2021-04-03 14:45:34+00:00  United Kingdom           GB
```

Figure 26 Country name and code conversion

5.2.2 Sentiment analysis using pre-trained model of Transformer

Since the beginning of Natural Language Processing, sentiment analysis has been a common activity (NLP). It is a subtask or application of text categorization that extracts and identifies sentiments or subjective information from various texts. As a result, sentiment analysis entails extracting human feelings, emotions, or moods from language - most commonly text. Many firms today utilise sentiment analysis to gain a better understanding of their customers and clients by studying sentiments across various target groups. It also has a wide range of applications in a variety of information sources, such as product reviews, online social media, survey feedback, and so on (Raymond, 2021). In this project, the Transformers library by

Huggingface have been used, which is the pretrained transformers pipeline to detect sentiment from the text so there is no need to train the model for sentiment classification. To use this library, Tensorflow in backend has been installed. the library's pipeline module, which provides a straightforward API for doing various NLP operations while hiding any code complexity behind an abstraction layer. To perform sentiment analysis on text stemming on the text using pre-processing Text hero library has been applied and removed stop words for better results. Figure 27 shows the result of the sentiment analysis applied to pre-processed sentences. We can see from the below result that all the sentences have negative sentiments as these are the tweets of protest. Refer to APPENDIX C for code implementation.

```
print(tweets_df[["clean_text",'sentiment']].head(5))
✓ 0.5s
```

Datetime	clean_text	sentiment
2019-09-17 09:02:00+00:00	first broadcast in the uk otd in 1966 episode...	NEGATIVE
2020-08-25 00:24:08+00:00	doctor's suicide mysuru protests may slow dow...	NEGATIVE
2019-10-29 14:50:00+00:00	families across the uk are considering a hunge...	NEGATIVE
2019-06-14 06:23:27+00:00	resident doctors association in aiims delhi a...	NEGATIVE
2021-04-03 14:45:34+00:00	via nigerians in london are back at "abuja h...	NEGATIVE

Figure 27 Sentiment analysis on pre-processed tweet text

The figure 28 shows the ratio of positive and negative tweets from in the final extracted data of the health protest. It can be notice from the pie-chart that there are 84.85% tweets are negative and 15.15 % tweets are positive.

```
tweets_df.groupby('sentiment').size().plot(kind='pie', autopct='%.2f')
<AxesSubplot:ylabel='None'>
```

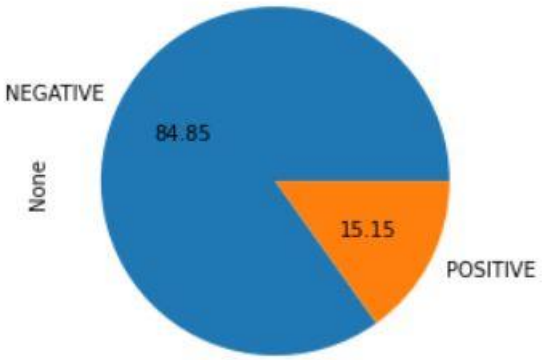


Figure 28 Sentiment analysis ratio- pie chart

5.3 Further Analysation of the data using Topic Modelling and Text Summarizer

5.3.1 Feature extraction for Topic Modelling

The extraction of features is more important in identifying significant patterns and obtaining information. Selecting these attributes aids in dimensionality reduction, which improves text mining performance. The features that were extracted and selected are represented as vectors. The vector representation transforms meaningful words into numbers representing context terms, with weights assigned to each word (Lingyun, et al., 2018). Frequency based Embedding (Guan-Bin & Hung-Yu, 2015), Prediction based Embedding (Guangxu, et al., 2016), and Non-Negative Matrix Factorization (Yong, et al., 2019) are some of the methods for extracting words. Count Vector, Co-occurrence vector, and document frequency are all part of the TF-IDF (Term-Frequency – Inverse Document Frequency) model (Harish & Revanasiddappa, 2017). The TF-IDF is a numerical statistic that is used to rate the importance of a word (term) in any material from a collection of documents based on the number of times the word appears, and it also checks how relevant the keyword is in the corpus. It also considers not only the frequency, but also the discriminative information for each phrase. Inverse document frequency measures how many documents a term appears in and divides it by the total number of documents in the corpus, whereas term frequency calculates how many documents a term appears in and divides it by the total number of documents in the corpus as shown below formula (1) and (2). In this project Counter vectorizer has been used to make a vector of the sentence and use it as input for the LDA.

Which will count the word frequency in the sentence and assigned the weight as per its frequency.

$$TF = \frac{num\ of\ occurrences\ of\ word\ in\ documents}{num\ of\ words\ in\ all\ documents} \quad (1)$$

$$IDF = \log \frac{num\ of\ documents}{num\ of\ documents\ with\ word\ occurs} \quad (2)$$

5.3.2 Latent Dirichlet Allocation (LDA)

LDA, developed by (David, et al., 2003) is a probabilistic model that is widely used in real-world applications to extract topics from document collections because it produces accurate results and can be trained online. In the LDA model, the corpus is arranged as a random mixture of latent topics, with the topic referring to a word distribution. In the Bayesian statistical paradigm, LDA is a generative unsupervised statistical algorithm for extracting theme information (topics) from a collection of documents. Figure 29 depicts a typical topic generation procedure. This means that we don't need to provide labels (that is, subject names that match each document) during training to train the model. This not only aids in the discovery of potentially intriguing topics, but it also saves the amount of time spent manually tagging texts. Each document, according to the LDA model, is made up of several subjects, each of which is a probability distribution over words. The ability to infer topics from a given collection is a significant benefit of employing the LDA model. The LDA model has the advantage of being able to infer themes from a given collection without the need for any prior knowledge. Figure 30 depicts a schematic design of the LDA topic model (Albalawi, et al., 2020). In this project, the LDA extracted 5 main topics from the pre-processed text to analyse further and then this topic can used to get the insights from the protest data such as which topic is most frequently used.

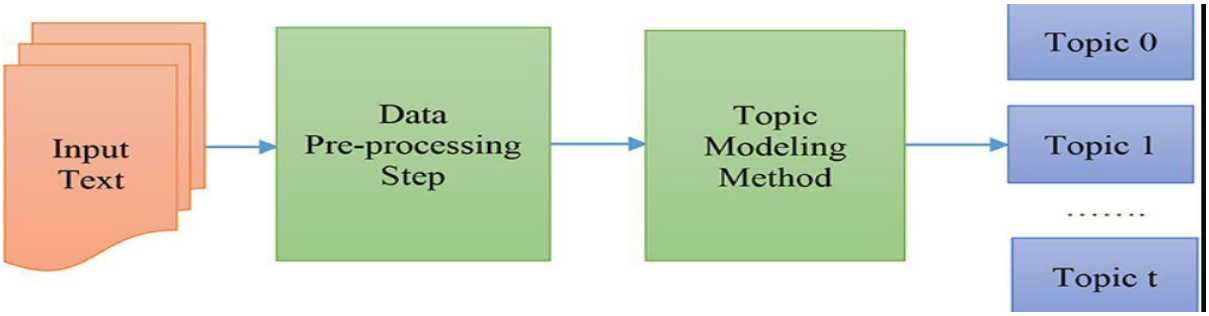


Figure 29 Topic modelling from the sentence (Rania, et al., 2020)

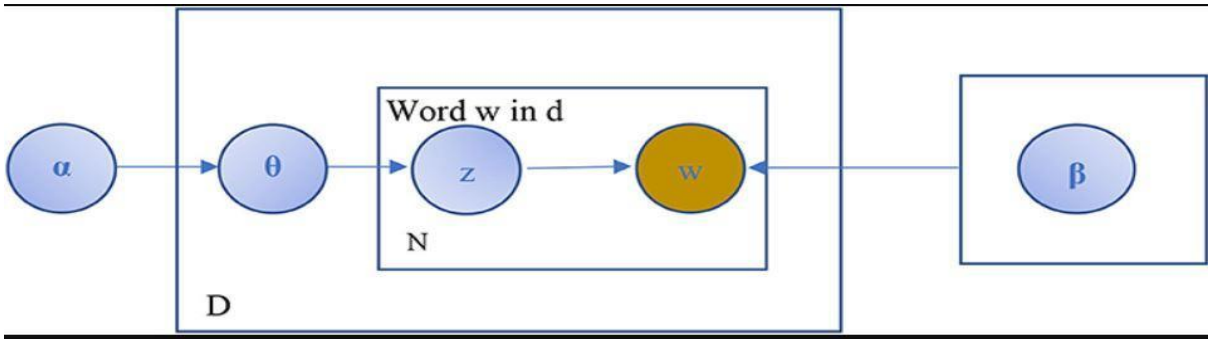


Figure 30 schematic design of the LDA topic model (Rania, et al., 2020)

From figure 31 It can be noticed that the LDA topic modelling has extracted unique and important topics from the sentence as a result and refer APPENDIX C for code implementation.

```
print(tweets_df['topics'].head(5))
✓ 0.7s
Datetime
2019-09-17 09:02:00+00:00    {0: 'smuggler doctor known 28th serial 02 pike...
2020-08-25 00:24:08+00:00    {0: 'suicid busi 19 time news updat download s...
2019-10-29 14:50:00+00:00    {0: 'consid cannabi strike hunger parent heart...
2019-06-14 06:23:27+00:00    {0: 'delhi doctor solidar aiim student protest...
2021-04-03 14:45:34+00:00    {0: 'nigerian strike check abuja uk wick hous ...
Name: topics, dtype: object
```

Figure 31 Result of Topic modelling on Pre-processed text

5.3.3 Summarizer

The objective of text summarising is to condense vast passages of text into a summary that retains vital information content and overall meaning. The Extractive Type and the Abstractive Type of Text Summarization are the two types of text summarization. Extractive

summarization derives information from the original text that is identical to it. In other words, rather than delivering a unique summary based on the entire material, it will compare each sentence in the document to every other sentence, based on how effectively each line explains. Abstractive, on the other hand, aims to create a one-of-a-kind summary by extracting the most important points from the original text (ANANDA, 2021). For this project, I have used a text summarizer by Huggingface transformer. The Hugging Face Transformer employs the Abstractive Summarization method, in which the model generates a new text that is shorter than the original by generating new phrases in a new structure, just like people do. This toolkit, which is built on top of PyTorch and TensorFlow, allows us to create Transformer models and use them for several purposes. Below is the library used to implement the summarizer.

```
from transformers import pipeline
summarizer = pipeline("summarization")
```

The figure 32 shows the summary result of the tweet text by its unique location.in which I have summarised all the row of the dataset by unique location such as tweet text, location, date, hashtags, media URL, tweet URL, sentiments and sum of retweet counts, and favourites counts. However, I have not achieved the expected result by performing a text summarizer. Refer to APPENDIX C for code implementation.

```
print(tweets_df['summary'].head(5))
✓ 0.4s
0    doctorwhofamilies across the uk are considerin...
0    doctor's suicide mysuru protests may slow down...
0    billboard services call 0740789778 easy to pla...
0    ibadan gve the minister of labour and employme...
0    l2lk ttimes top story 75 doctors walkout in pr...
Name: summary, dtype: object
```

Figure 32 Summary of tweet text by unique location

5.4 programming language used

-**Python Programming**: Python is a high-level, general-purpose programming language that is interpreted. The use of considerable indentation in its design philosophy emphasises code

readability. Its language elements and object-oriented approach are aimed at assisting programmers in writing clear, logical code for both small and large-scale projects.
It is one of the best languages for implementing machine learning and Natural language processing. It provides rich packages and libraries that are used in machine learning and NLP (WIKIPEDIA, 2020).

-HTML: HTML, or HyperText Markup Language, is the standard markup language for texts that are intended to be viewed on a web browser. Technologies such as Cascading Style Sheets (CSS) and programming languages like JavaScript can help (WIKIPEDIA, n.d.).

-CSS: CSS (Cascading Style Sheets) is a style sheet language for describing the appearance of a document authored in a markup language like HTML. Along with HTML and JavaScript, CSS is a key component of the World Wide Web (WIKIPEDIA, n.d.).

5.5 Technology Used

- **AJAX**: Ajax is a set of web development approaches that generate asynchronous web applications by combining multiple client-side web technologies. Ajax allows web applications to transmit and retrieve data from a server asynchronously without interfering with the existing page's appearance and behaviour (WIKIPEDIA, n.d.).

- **Python's Local Database-SQLite Database:** Many database systems, including MySQL, Postregsql, Oracle, Microsoft SQL Server, and Maria DB, have Python bindings. SQLite is one of these database management systems (DBMS). A relational database management system called SQLite. SQLite is the world's most extensively used SQL database engine. SQLite's source code is available in the public domain.

-Jupyter Notebook: Jupyter is a computational notebook, a free, open-source, interactive web tool that allows academics to mix software code, computational output, explanatory text, and multimedia resources in a single document. Jupyter has risen in popularity in recent years, even though computational notebooks have been available for decades. The notebook's rapid adoption has been aided by an enthusiastic community of user–developers and a redesigned architecture that allows it to speak dozens of programming languages — a fact reflected in its name, which was inspired by the programming languages Julia (Ju), Python (Py), and R, according to co-founder Fernando Pérez (Jeffrey, 2018).

-Visual Code: Microsoft's Visual Studio Code is a source-code editor for Windows, Linux, and macOS. Debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git are among the features. (WIKIPEDIA, n.d.)

-Pycharm: PyCharm is an integrated development environment for computer programming, with a focus on the Python programming language. JetBrains, a Czech firm, developed it. (WIKIPEDIA, n.d.)

-Google Collab: Google Colab, often known as Colaboratory, is a Google Research freemium product that is built on Jupyter. Colab is a wonderful tool for both novice and advanced users; it comes pre-installed with practically all necessary libraries, so we don't have to install them one by one. The notebook files in Colab are kept on our Google Drive and may be accessed from anywhere. It also allows us to share our notebook with a colleague without having to download it, which many people find to be the most useful function. In addition, it offers free GPU and TPU for our work, making it excellent for Deep Learning and Machine Learning projects (Buggyprogrammer, 2021).

-Django framework: Django is a free and open-source web framework built on Python and following the model–template–views architectural paradigm. It is maintained by the Django Software Foundation, a non-profit organisation based in the United States (WIKIPEDIA, n.d.).

5.6. Packages, Libraries used

-**Pandas**: pandas is a data manipulation and analysis software package for the Python programming language. It includes data structures and methods for manipulating numerical tables and time series (WIKIPEDIA, 2008).

-**matplotlib**: Matplotlib is a fantastic Python visualisation package for 2D array charts. Matplotlib is a multi-platform data visualisation package based on NumPy arrays and intended to operate with the SciPy stack. It was first introduced in 2002 by John Hunter. One of the most significant advantages of visualisation is that it provides us with visual access to massive volumes of data in simply understandable graphics. Matplotlib has a variety of plots such as line, bar, scatter, histogram, and so on (Meghna, 2018).

- **Numpy**: NumPy is a Python module that allows you to interact with arrays. It also provides functions for working with matrices, fourier transforms, and linear algebra.

Travis Oliphant invented NumPy in 2005. It is an open-source project that you are free to use. Numerical Python is referred to as NumPy (W3School, n.d.).

- **Snscrape**: snscrape is a scraper for social media platforms (SNS). It scrapes information such as user profiles, hashtags, and searches and returns the results, such as relevant postings (GIthub, n.d.).

- **datetime:** used to have functionality of the datetime.

- **alive_bar:** used to show the progress bar of the current task.

- **texthero**: Texthero is a Python tool for rapidly working with text data. It provides a reliable pipeline to clean and represent text data from zero to hero, giving NLP professionals a tool to swiftly grasp any text-based information (Texthero, n.d.).

- **Huggingface**: Hugging Face aspires to be the GitHub of the machine learning world. Hugging Face is one of the most well-known NLP start-ups. Its library is used in production by big tech companies including Apple, Monzo, and Bing (SHRADDHA, 2021).

-**Transformer Library by Huggingface**: It implements a variety of transformer models using easy-to-use and extendable APIs. They also give popular models that have been pre-trained on regularly used datasets, reducing the amount of time it takes to get started with transformers (Utkarsh, 2020).

-**sklearn:** In Python, Scikit-learn (Sklearn) is the most usable and robust machine learning library. It uses a Python consistency interface to give a set of efficient tools for machine learning and statistical modelling, such as classification, regression, clustering, and dimensionality reduction. NumPy, SciPy, and Matplotlib are the foundations of this package, which is mostly written in Python (Tutorialspoint, n.d.).

-**Pytorch:** PyTorch is a tensor library intended for usage with GPUs and CPUs in Deep Learning applications. It is a Python-based open-source machine learning package developed mostly by the Facebook AI Research team. It's one of the most popular machine learning libraries, alongside TensorFlow and Keras (Narasimha, 2021).

-**Spacy:** spaCy is a Python library for advanced Natural Language Processing (NLP). It is free and open source. spaCy is a production-ready tool that allows us to create apps that analyse and "understand" massive amounts of text. It may be used to create data extraction and natural language understanding systems, as well as to pre-process text for deep learning (spaCy, n.d.).

-**pyCountry**: It is used to get the country name from the city, state, or any address.

-**geopy**: It identifies and tracks the location of linked electronic devices using various location technologies such as GPS and IP addresses. Python's GeoPy package will be used. When you type in a location name, it returns all pertinent information, including the postal code, city,

state, and nation, as well as latitudes and longitudes. It returns the location name when we provide the coordinates. Geopy is not a Python library that comes pre-installed. It must be explicitly installed (Srijata, n.d.).

6. Result &Testing

The below figure 33 shows the result of the file existence which stores the last collected date of the data.

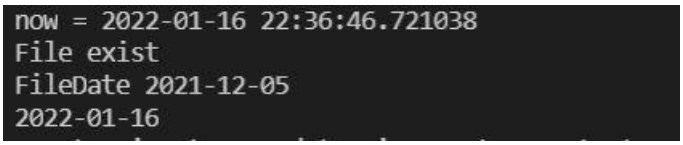


Figure 33 Check for date stored in text file

The below figure 34 shows the new data collected.

Figure 34 New data collected from the stored date in file

The below figure 35 shows the data pre-processing and removed the unnecessary data from the collected data and generate new csv. In which missing values from the location rows has been dropped.

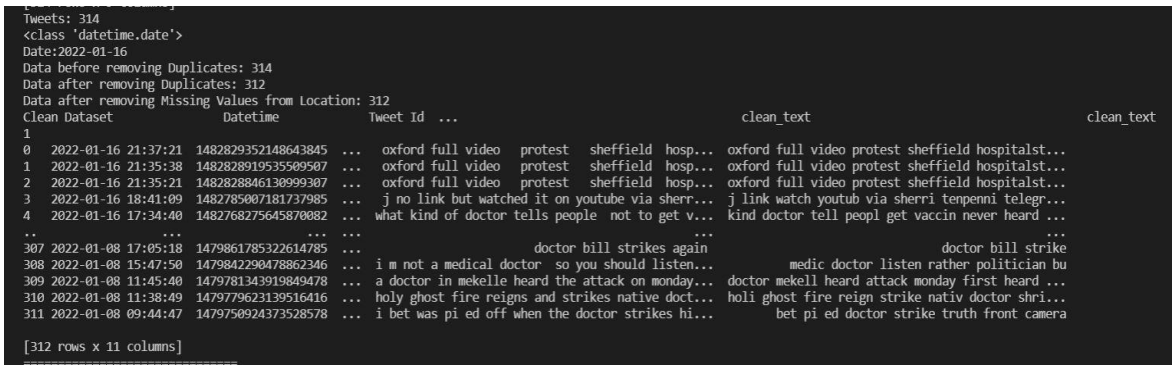


Figure 35 Data Pre-processing on new collected data

The figure 36 shows the clean text after Pre-processing of the new collected data.

```
print(tweets_df['clean_text'].tail(10))
✓ 0.4s
Datetime
2022-01-13 15:12:22                    doctor  says  india   ghonia ...
2022-01-13 02:12:46                      doctor  alokmittal  india  ...
2022-01-13 02:12:22                       doctor  alokmittal  india ...
2022-01-12 20:51:34     oxford for the same reason that the people wh...
2022-01-12 17:55:16                                india former ntag...
2022-01-12 09:25:02    india lost one more doctor  a bright student  ...
2022-01-15 21:07:13    so  a free dumb doctor in england points out t...
2022-01-11 10:44:22    thousands of doctors medicos test covid positi...
2022-01-10 18:24:55    who do guys on strikes  that is more than a ju...
2022-01-10 00:20:37    white goy strikes again  romania  poland  russ...
Name: clean_text, dtype: object
```

Figure 36 Clean text after pre-processing of new data

The figure 37 shows data analysis on new collected data after pre-processing, in which NER extracts all the required fields from the text. Also, geopy and pyCountry extract the address and the country name from the location mentioned in tweet text. After, extracting location from the GPE entity if there are any missing values it will be dropped and process further. From GPE, geopy and pyCountry will extract the address and country name is there are any missing value found it will also drop the missing values and generate the final csv based on available data.

```
[312 rows x 11 columns]
==============================
Getting Organization.. ..
==============================
Getting Type of Protest.. ..
==============================
Getting nationalities, religious, or political groups involved.. ..
==============================
Getting Protest Size.. ..
==============================
Getting GPE_Location from Tweet Text.. ..
==============================
Getting Location from Tweet Text.. ..
==============================
Getting Date from Tweet Text.. ..
==============================
Getting person name.. ..
Data after removing Missing Values from GPE: 77
==============================
Finding Location Using Geopy Libray.. ..
==============================
Finding Country pycountry.. ..
Data after removing Missing Values from Country: 13
Data after removing Missing Values from Country_Code: 13
```

Figure 37 Data analysis on new data

The figure 38 shows the result of the extracted address, country name from the address and country code from the country name using geopy, pyCountry and pyCountry-converter.

```
print(tweets_df[["GPE",'geo_country','Country','Country_Code']].head(5))
✓ 0.8s
```

Datetime	GPE	geo_country	Country	Country_Code
2022-01-16 14:46:04	['vancouver']	Vancouver, District of North Vancouver, Metro ...	Canada	CA
2022-01-15 06:45:08	['india']	India	India	IN
2022-01-15 02:27:02	['india']	India	India	IN
2022-01-13 15:12:22	['india']	India	India	IN
2022-01-13 02:12:46	['india']	India	India	IN

Figure 38 Location Extraction

The figure 39 shows the result of the text summarizer which extract the unique country to summarize all data.

```
DONE!!!!!
==============================
Extract Summary.. ..
X is['Canada' 'India' 'United States' 'United Kingdom' 'Poland']
```

Figure 39 Shows summary of the data by grouping the unique location

The figure 40 shows the sentiment analysis of the new collected tweet text. It can be notice from the below pie-chart that there are more negative tweets than positive as the pattern of the protest have been negative always.

```
tweets_df.groupby('sentiment').size().plot(kind='pie', autopct='%.2f')
```

```
<AxesSubplot:ylabel='None'>
```

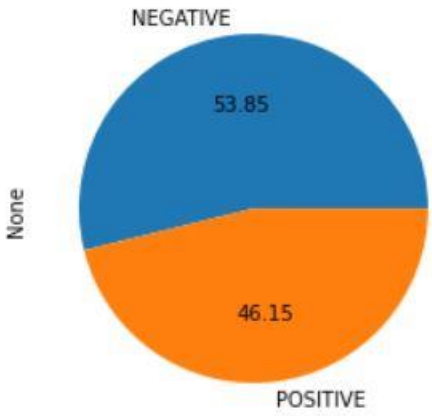


Figure 40 Sentiment analysis of the new data collected

The below figure 41, 42, 43 and 44 shows the word cloud of most frequently used text, location and hashtags in while posting the tweets about the health workers protest.

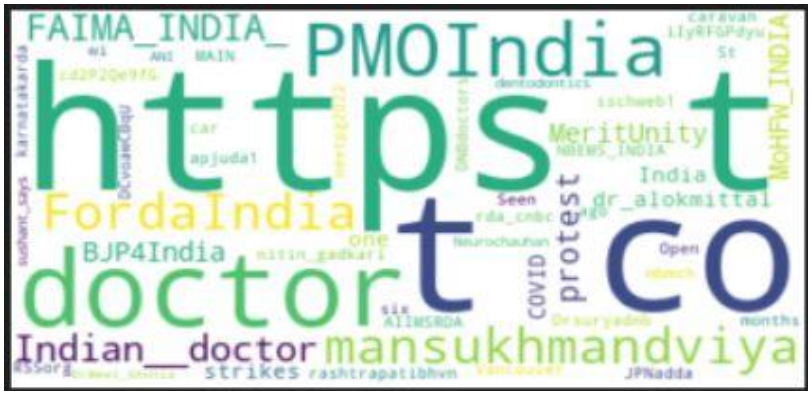


Figure 41 Word Cloud of most frequently used raw tweets

Figure 42 Word Cloud of most frequently used Pre-processed tweets

Figure 43 Word Cloud of most frequently used hashtags

Figure 44 Word Cloud of most frequently used location

7. Software Development

To visualise this Health Protest data, an interactive Web Application has been developed using the Django framework containing Graphical views of the data in the table and the SVG Interactive Map javascript. This website has been created using Python programming as a back-end and Sqlite3 to manage the database; code for the same is shown in Appendix D. However, the front-end has been implemented using CSS, AJAX JavaScript, and HTML code for the same is shown in Appendix C.

svgMap:

The svgMap.js library allows us to create an interactive, SVG-based global map on a website, complete with an Info Window that can be used to display any data when the user hovers over a certain country or region (stephan, 2021).

Following are the screenshots of the developed Application:

The figure 45 shows the design of the whole webpage view of an application. While scrolling down it will show all visualisation of data using an interactive map and data table showing all the information about the protest.

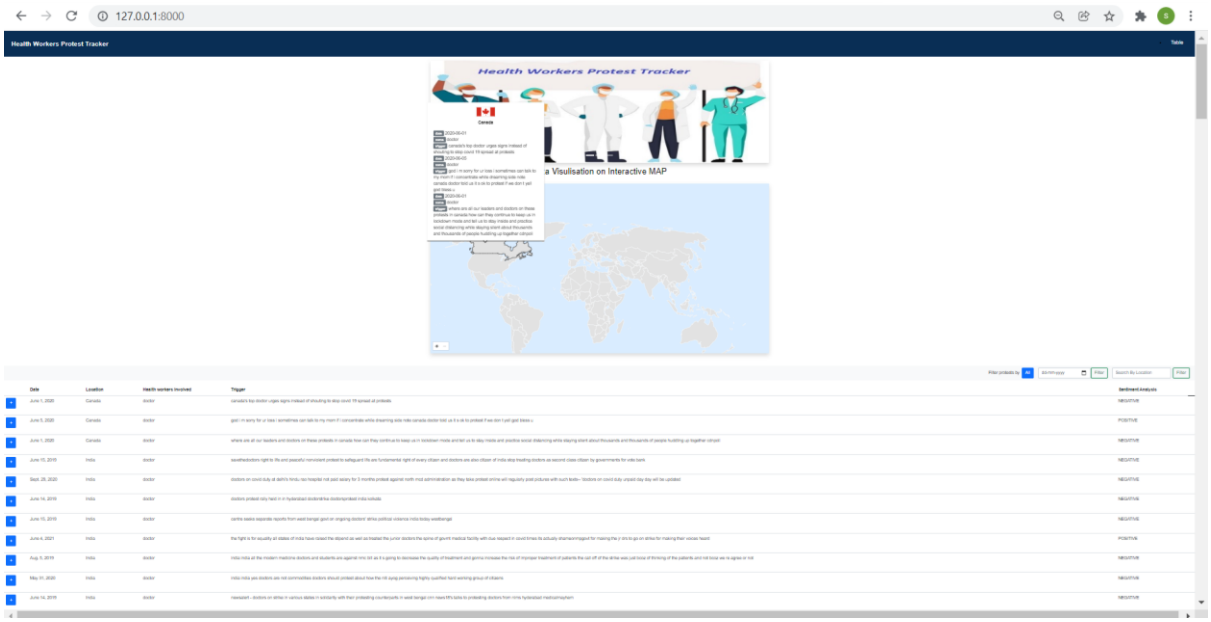


Figure 45 Whole Webpage with all integrated functionalities

Figure 46 Home page 1

The below figures. 47,48,49 shows the information on countries hover. While hovering on the map it will show the data if that country's protest data is available in our database. However, if the country's protest data is not available then it will show there is no protest data available. We can see from the below figures that the United Kingdom, India and the United States have protest data available, so it is showing on that country's hovering. The data is about the protest date, name of Health worker involved and the trigger for the protest.

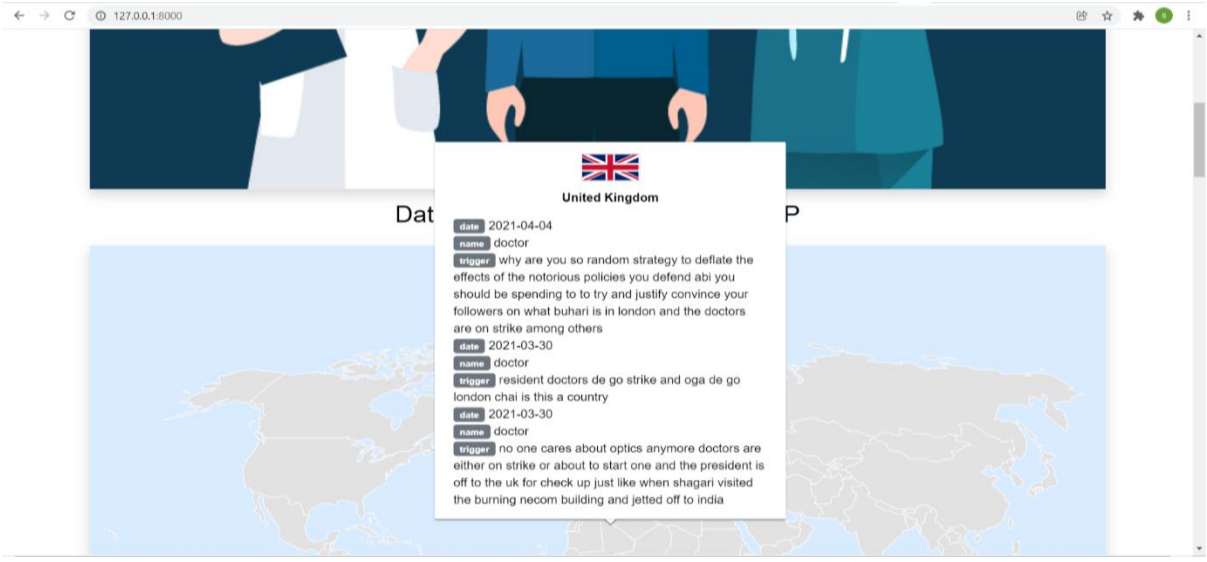


Figure 47 Data Visualisation on Interactive MAP -United Kingdom

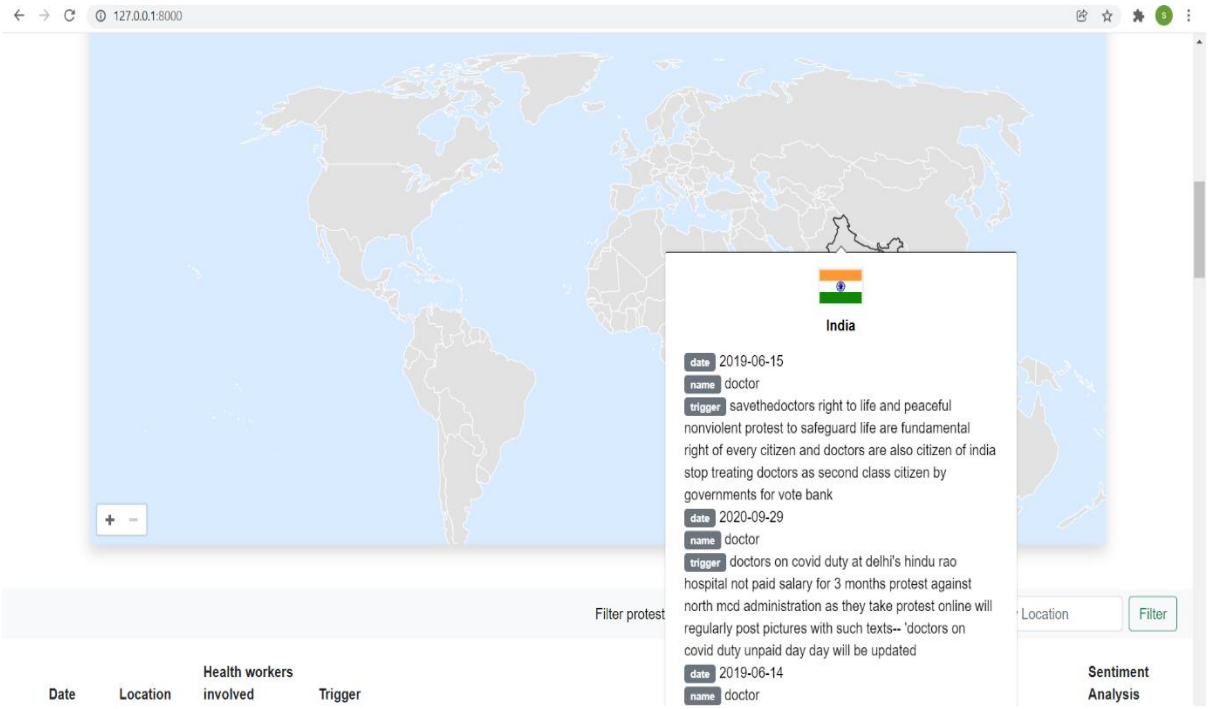


Figure 48 Data Visualisation on Interactive MAP- India

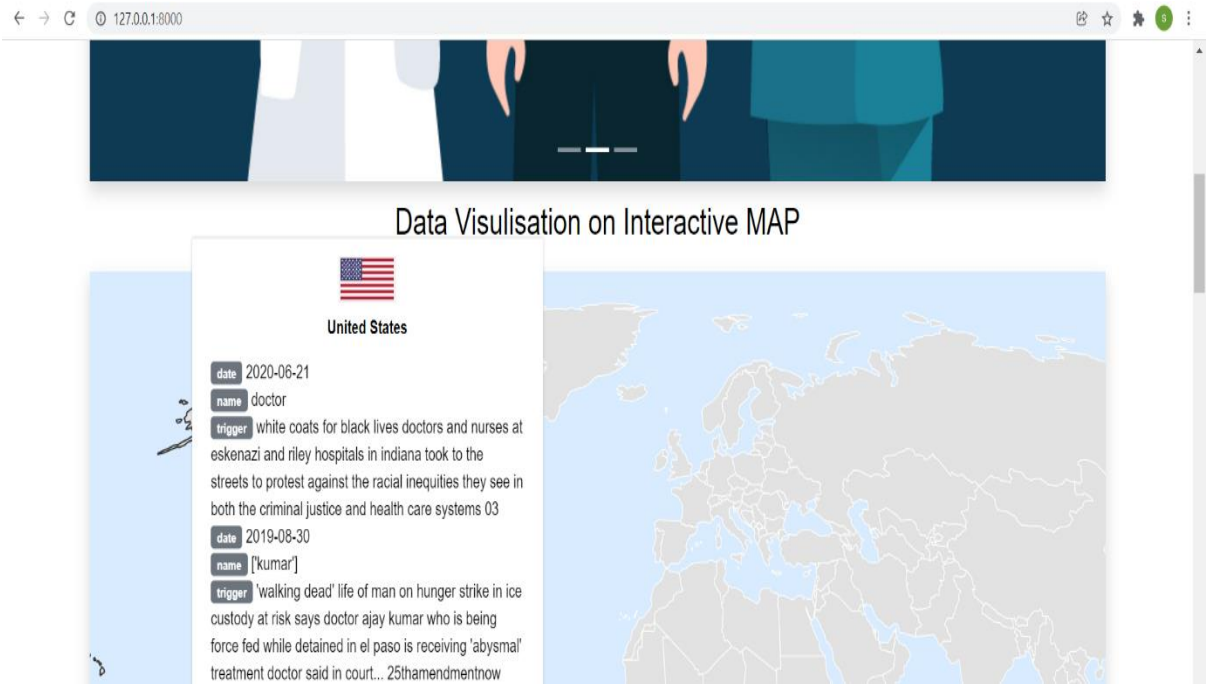


Figure 49 Data Visualisation on Interactive MAP- United States

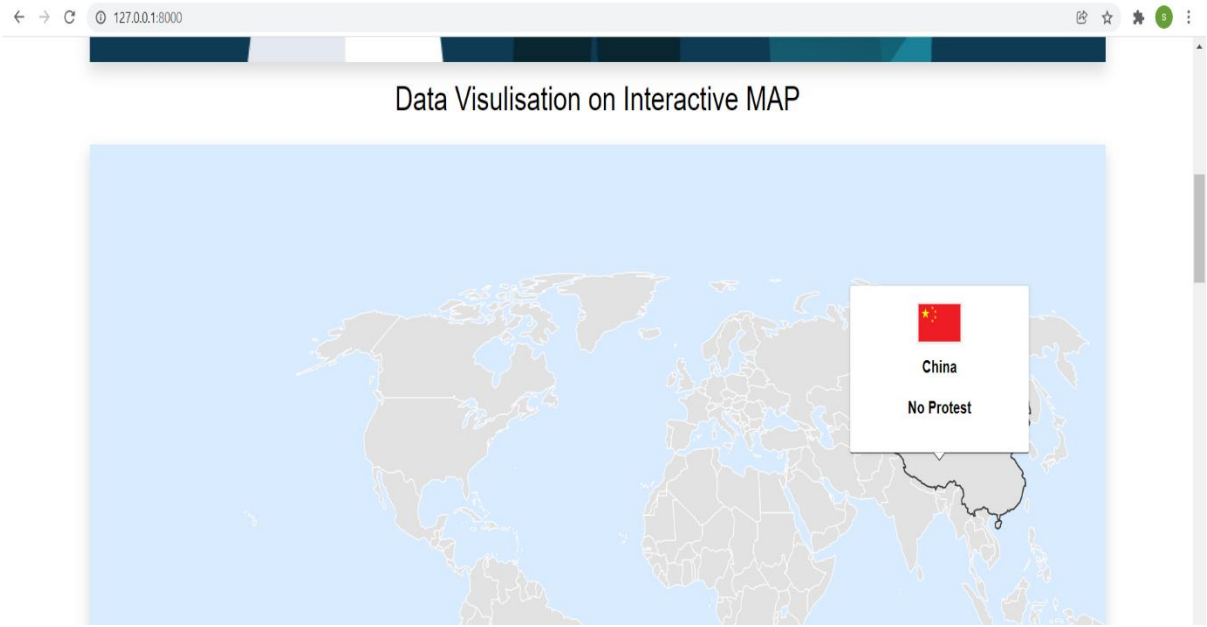


Figure 50 Data Visualisation on Interactive MAP- showing No data available for this country

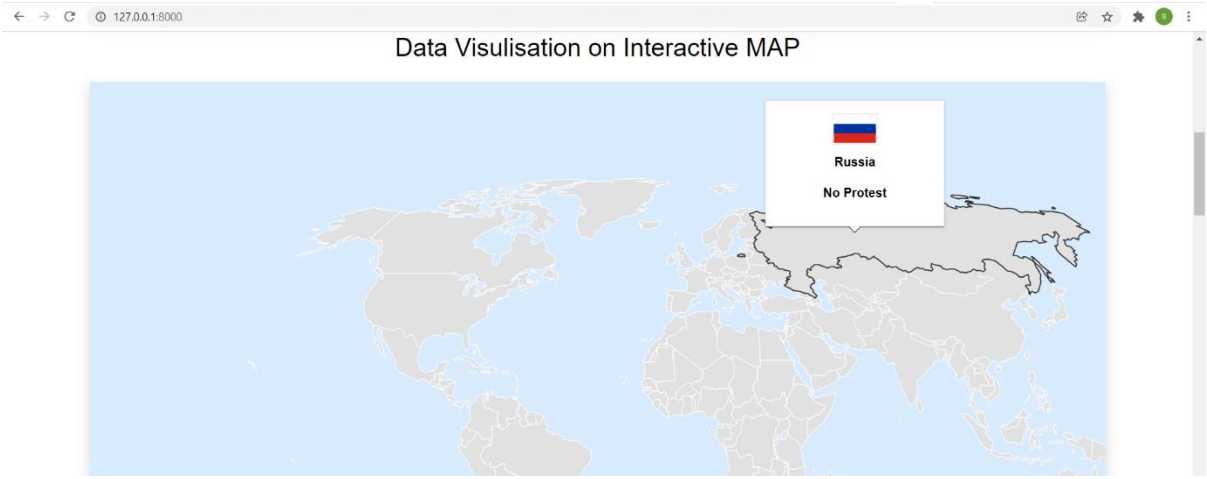


Figure 51 Data Visualisation on Interactive MAP- showing No data available for this country

In this web application there are some features of filtering results such as filter by Date, filter by Country and all data. In this we can filter our data by performing search filters. And we can download filter data in the csv format on click of the Download CSV button to research more on that data shown as in the figures 52 and 54. However, figure 53 shows the expandable view for the row data. This data shows more information about the protest such as Tweet URL, Media URL, Hashtags used in the tweets, total retweets, and favourites (Likes) on that tweet. However, the Media URL is not functioning as it should be. There is some limitation on the

Media URL, but we can see on tweet URL link if there is any Media has been attached in the tweet.

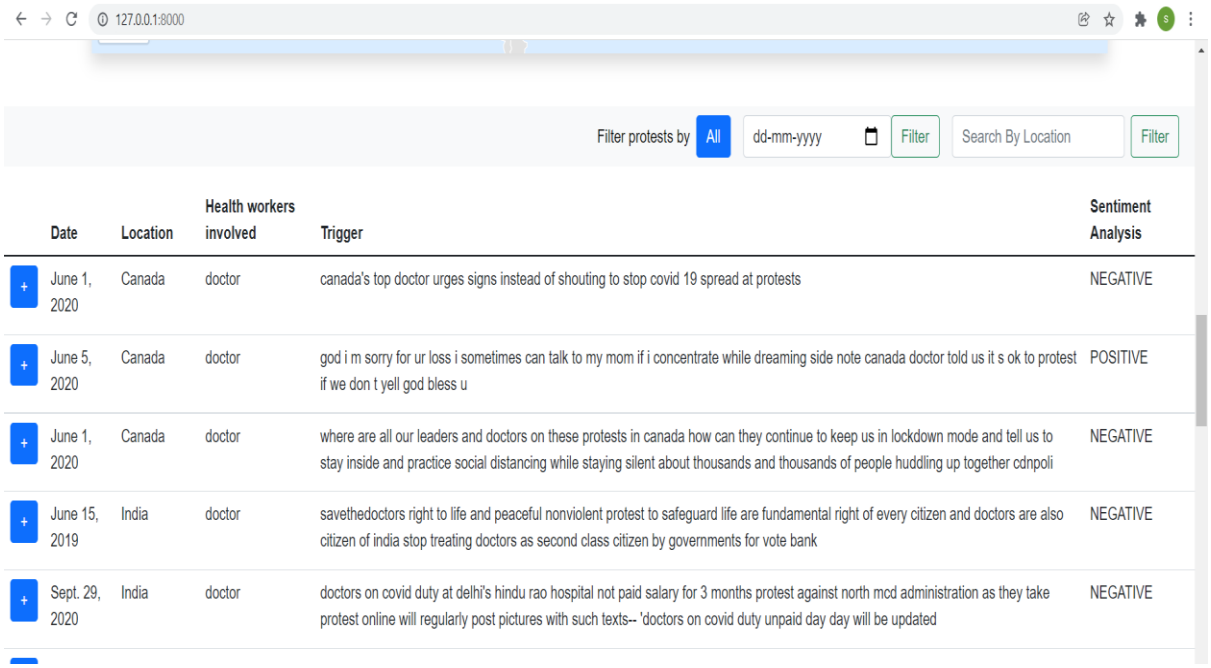


Figure 52 Table Showing all information of country protest

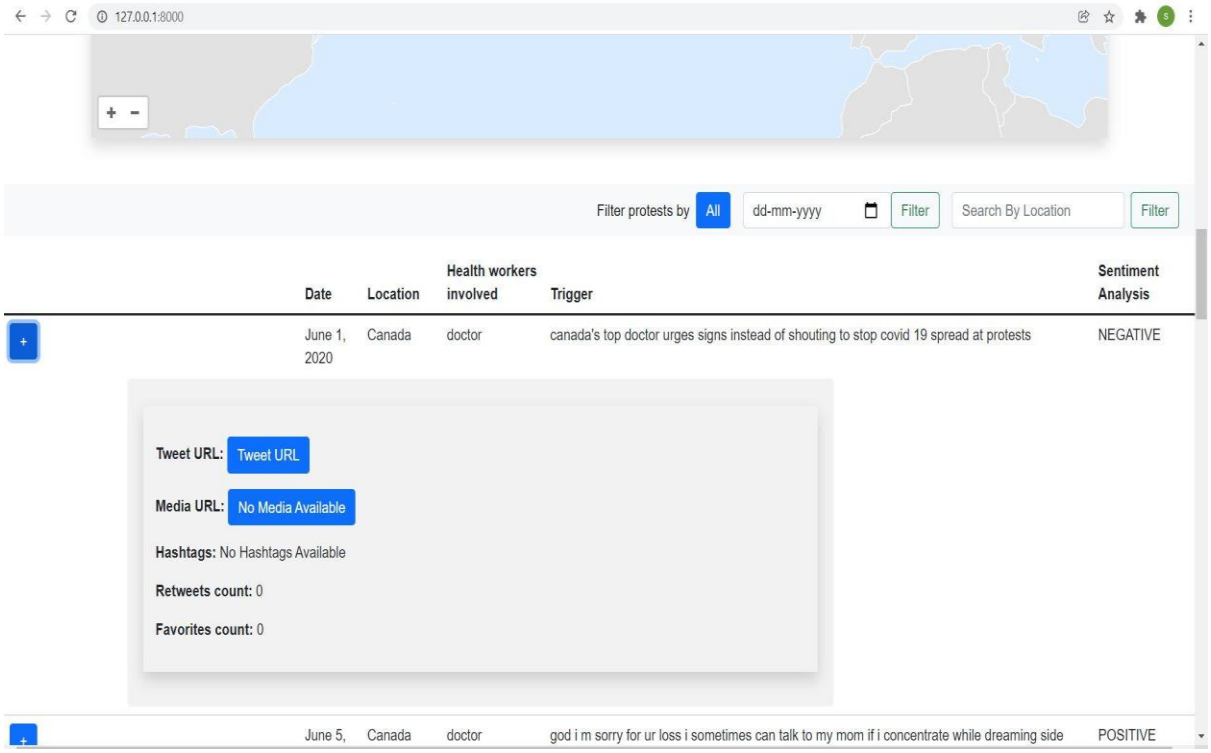


Figure 53 Table in expandable view Showing all other field of country protest

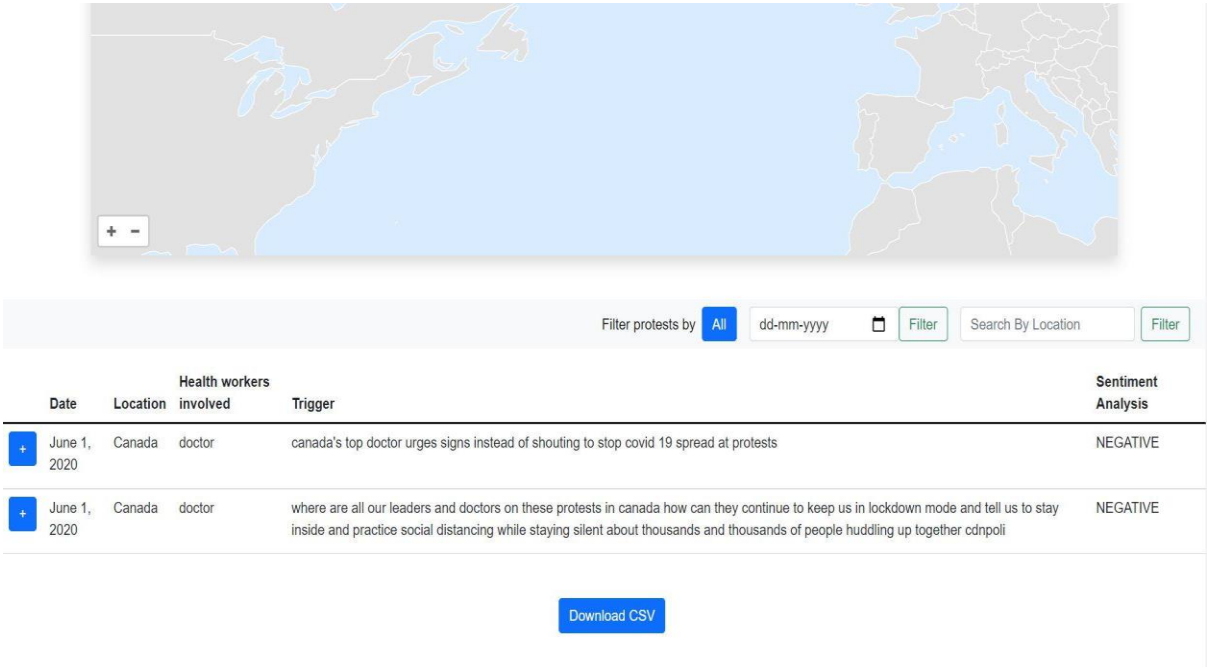


Figure 54 Table showing information about the protest filter by Date

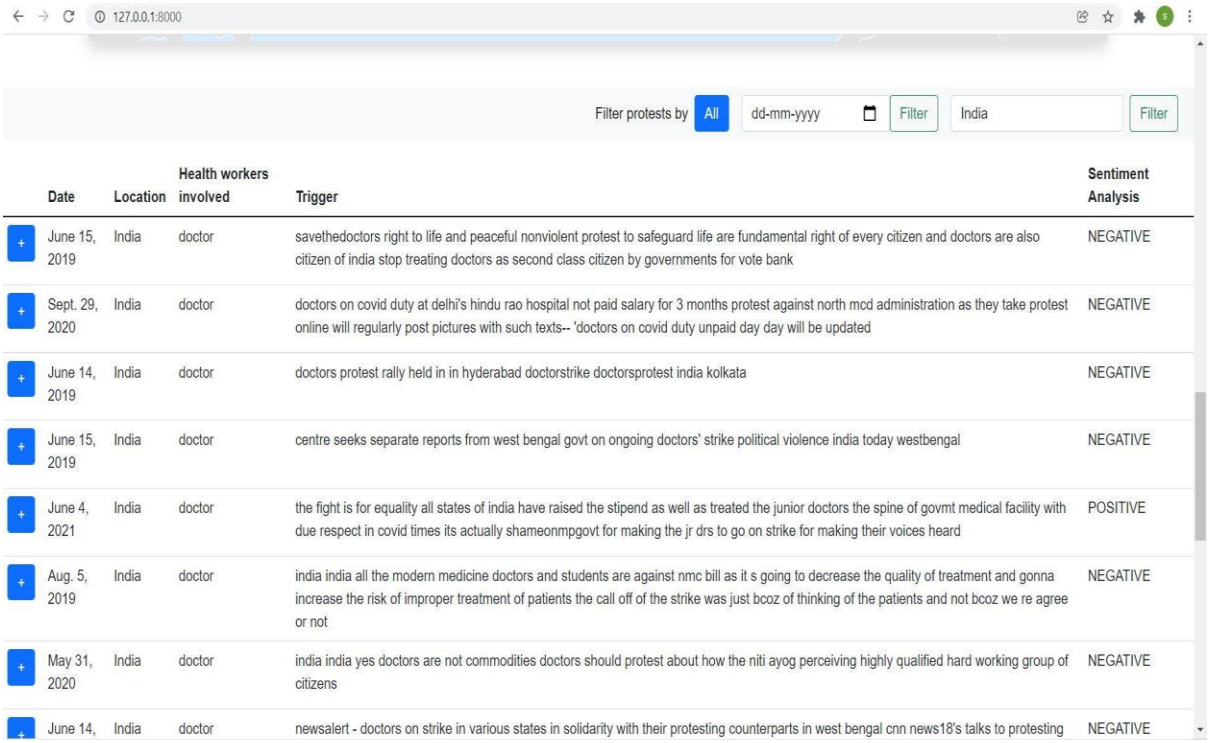


Figure 55 Table showing information about the protest filter by Country

8. Conclusion

8.1 Summary of the investigation study

Other protest tracking systems currently set their databases manually, so this is a unique approach to collecting data on health worker strikes for further investigation using social media, which is one of the most powerful platforms nowadays, particularly Twitter, which is one of the most popular microblogging platforms.

- In this proposed project, information about Health workers protests has been collected from twitter to better comprehend results of the protest on a global scale. On Twitter, data is usually unstructured, noisy, cluttered/disorganized, and written in slang. This research has gained the necessary insights and gather data from a global viewpoint to determine how widespread and this behaviour is. For this, 2.2 Lakhs historical tweets data has been collected from the Date 1st Jan,2019 to 5th Dec, using list of Keywords related to Health worker Protests. However, for the new data Twitter's API tweepy V2 has been used as Snscrape sometimes create issues for latest tweets due to that library is under development so it has been used only once to extract historical data however it was still working fine.
- After that Pre-processing was applied to clean the dataset to remove irrelevant, duplicate data and missing values. However, to analyse the data randomly 500 rows from the collected data has been sampled.
- Then to extract the information from that sample data, the Information Extraction technique of NLP has been used, which is Named Entity Recognition to automatically extract structured information from unstructured and/or semi-structured machine-readable texts. NER identifies and extracts all Entities (person names, Location, Event, etc) from the text. As Twitter has very noisy data it is difficult to find the relevant tweets about one topic so to find out the informative tweets, so using Location extracted by NER has been made as target feature. Using NER if the HPE or LOC entities extracts the location from the tweet text, then this tweet is classified as useful. If there is no location found in the tweet, then that row has been dropped from the data as the location is the key feature for this project.
- Moreover, the location, users are mentioning in tweet text might be local or not well-known or might be any kind of small address. So, to get the exact country name from the location geopy library has been applied to extract the full address of that location.

Further, from this full address pycountry library has been applied to get the country name from that address and finally from that country name country code has been extracted using pycountry-converter as the data has been located using country code on map.

- Then this all-analyzed data CSV will be generated, and it will be stored in the SQlite Database. However, all above steps from data collection to Data storage are automated. Every day the script will run with the help of Windows Task Scheduler and process the data and store it into the database. And then this data will be visualized on the Interactive map through the Web application
- Then to analyse the data more Countvectorizer method has been applied to extract the feature names from the tweet text using Tfidf vectorizer method of sklearn. The Count Vectorizer returns the number of occurrences per word index. After having features names, unsupervised Topic modelling method LDA has been used to get the most frequent topics from the vectors and summaries the same topic using summarizer model of NLP by its unique location to get all tweets related to same topics and location.

8.2 Challenges of the proposed system

This proposed Healthcare Worker Protest Tracker has been designed to track the Health Worker Protest, but it has some limitations such as with regards to Extraction of information about the Protest and Data visualization on Map.

During the development stages of this product, some challenges were faced such as followings:

- All social media does not allow to scrape all the information. Such as recently Facebook has kept usage limitation using its API to scrape the data. Due to this reason, I have focused on twitter only.
- Complexity in collecting historical data from twitter using snscrape library which is sometimes not recognised by twitter because it is still under development stage. Alternatively, in this project both snscrape and tweepy have been used to collect the data. The twitter's own API which is tweepy has many restrictions to collect the data such as we can extract past 7 days tweets only though twitter's academic research API has Full-archived API which allows us to extract up to 30 days or more than that. Also, to use this Twitter API's it requires the Developer account and API authentication Keys

such as consumer key, consumer secret, access token, access token secret. However, it's easy to get that but there are still use case restrictions.

- Complexity in identifying the relevant and informative tweets about healthcare protest from the hundred, thousand or more than that for the same topic or same protest.
- Complexity in extracting all the required fields such as duration, size of the protest by NER model (Named Entity Recognised) as NER can extract limited entities from the text.
- Complexity in extracting country name from the local addresses using pyCountry library so additionally I have also used geopy library which extract full address from any kind of address with the country name. But there is usage limitation in geopy such as one request per second and to process the large data it is paid.
- Complexity in using all functionalities of an interactive SVGMAP image library as the customization of the library version is paid.
- Complexity in using Text summarizer using Transformer by Huggingface Library

8.3 Recommendation for future work

- As the API used in this project has use case limitation but applying for twitter's academic research Full-archived API which allows us to extract up to 30 days or more than that would be a great option as sometimes Twitter does not recognise the snscrape library while using that to extract the data.

- There is limitation using functionalities of an Interactive Map of SVGmap-Image as to use all the functionalities of that interactive map Javascript plugin for that map is paid. So, to use that functionality for better result we can purchase the plugins in future.

- There is some limitation using libraries such as python's package geopy (used to identify and track the location). This service is based on donated servers and has a use limitation in the service. As a result, they ask that we limit our use and follow their usage policy. The text summarizer by Huggingface Transformer (run on top of PyTorch and TensorFlow) requires a lot of memory because PyTorch or TensorFlow libraries need to be installed to use this transformer. So, in future we can purchase the geopy library to process the large data. And, to run this big library such as PyTorch or

TensorFlow for summarise it needed to use the high-configuration system or use the cloud.

References

Akshay, S., 2021. Named Entity Recognition using SpaCy (NER).

Alan, R., Sam, C. & Mausam, a. O. E., 2011. *Named entity recognition in tweets: An experimental study. In Proceedings of the conference on empirical methods in natural language processing.* [Online].

Albalawi, R., Yeap, T. H. & Benyoucef, M., 2020. Using Topic Modeling Methods for Short-Text Data: A Comparative Analysis. *Frontiers in Artificial Intelligence,* Volume 3, p. 42.

Amandeep, K. & Rajinder, S., 2019. Implementing sentiment analysis in relevance with. *J. Emerg. Technol. Innov.*

Buggyprogrammer, 2021. *Jupyter Vs Colab Which Is Better And Which One Should You Use In 2021*. [Online]
Available at: https://buggyprogrammer.com/jupyter-vs-colab/

Chima, S. C., 2013. Global medicine: is it ethical or morally justifiable for doctors and other. *BMC Med Ethics,* Volume 14 Suppl 1(Suppl 1):S5.

Chima, S. C., 2020 . *Doctor and healthcare workers strike: are they ethical or morally justifiable: another view, Current Opinion in Anaesthesiology.* [Online]
Available at: https://journals.lww.com/co-anesthesiology/Abstract/2020/04000/Doctor_and_healthcare_workers_strike__are_they.13.aspx
[Accessed 23 MAY 2021].

Chima, S. C., 2020. Doctor and healthcare workers strike: are they ethical or morally. *Curr Opin Anaesthesiol,* Volume 33(2):203-210.

David, M. B., Andrew, Y. N. & Michael, I. J., 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research,* p. 3:993–1022.

Gerald, O. et al., 2019. Effect of strikes by health workers on mortality between 2010 and 2016 in Kilifi, Kenya: a population-based cohort analysis.. *Lancet Glob Health,* pp. 7(7):e961-e967.

GIthub, n.d. *snscrape.* [Online]
Available at: https://github.com/JustAnotherArchivist/snscrape

Giuliano, R. et al., 2019. Health Workers' Strikes in low-income countries. *Bull World Health Organ,* Volume 97, pp. 460-467H.

Guan-Bin, C. & Hung-Yu, K., 2015. Word Co-occurrence Augmented Topic Model in. Volume Vol. 20, pp. pp. 45-64.

Guangxu, X. et al., 2016. Topic Discovery for Short Texts Using Word Embeddings.
Harish, B. S. & Revanasiddappa, M. B., 2017. A Comprehensive Survey on various Feature Selection Methods to Categorize Text Documents. *International Journal of Computer Applications,* Volume 164(8), pp. 1-7.

harmila, P., 2020. *20+ POPULAR NLP TEXT PREPROCESSING TECHNIQUES IMPLEMENTATION IN PYTHON.* [Online]
Available at: https://dataaspirant.com/nlp-text-preprocessing-techniques-implementation-python/

Indian Institute of Technology Bombay, n.d. *Information extraction and retrieval.* [Online]
Available at: https://rnd.iitb.ac.in/research-glimpse/information-extraction-and-retrieval
Insecurity Insight, 2021. *Violence Against Health Care in Myanmar 11 February - 12 April 2021.*. [Online].

Jaffali S., J. S. K. N. H. A., 2020. Survey on social networks data analysis," in Innovations for Community Services.. *Communications in Computer and Information Science,* Volume vol 1139.

Jeffrey, M. P., 2018. Why Jupyter is data scientists' computational notebook of choice.
Lambert, S., 2021. *Social Network Usage & Growth Statistics: How Many People Use Social.* [Online]
Available at: https://financesonline.com/number-of-social-media-users/#:~:text=A%20more%20recent%20estimate%20showed%20that%20there%20were,gro

wth%20from%20the%200.97%20billion%20users%20in%202010.
[Accessed 07 June 2021].

Leon, D., Diana, M., Niraj, A. & Kalina, B., 2013. Microblog-genre noise and impact on semantic annotation accuracy. *ACM*, pp. 21-30.

Likhitha, S., Harish, B. S. & Keerthi Kumar, H. M., 2019. A Detailed Survey on Topic Modeling for Document and. *International Journal of Computer Applications (0975 – 8887,* p. Volume 178 – No. 39.

Lingyun, L., Yawei, S., Xu, H. & Cong, W., 2018. Research on Improve Topic Representation over Short Text. pp. pp. 848-853.

Meghna, K., 2018. *Python | Introduction to Matplotlib.* [Online]
Available at: https://www.geeksforgeeks.org/python-introduction-matplotlib/

Narasimha, J., 2021. *A Gentle Introduction to PyTorch Library for Deep Learning.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/04/a-gentle-introduction-to-pytorch-library/

Oyebode, O. et al., 2021. Health, Psychosocial, and Social Issues Emanating From the COVID-19 Pandemic Based on Social Media. *JMIR Med Inform*, 04 6.Volume 9.

PM., M. et al., 2016. Applying Multiple Data Collection Tools to Quantify Human Papillomavirus Vaccine Communication on Twitter. *J Med Internet Res,* Volume 18(12):e318.

pypi, 2021. *How to Compute Sentence Embedding Fast and Flexible.* [Online]
Available at: https://pypi.org/project/sent2vec/
[Accessed 30 12 2021].

PyPI, 2021. *snscrape.* [Online]
Available at: https://pypi.org/project/snscrape/
[Accessed 29 12 2021].

Raghav, A., 2021. *Must Known Techniques for text preprocessing in NLP.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/

Raymond, c., 2021. Sentiment Analysis with Pretrained Transformers Using Pytorch.
Ryan, E. & Sharon, M. W., 2021. Health Care Worker Strikes and the Covid Pandemic. *N Engl J Med,* p. 384:e93.

Sarlan, A., Nadam, C. & Basri, S., 2014. Twitter sentiment analysis. *Proceedings of the 6th International Conference on Information Technology and Multimedi,* pp. pp. 212-216.

Scott, T., 2020. Scraping Tweets by Location in Python using snscrape.
SHRADDHA, G., 2021. *Hugging Face And Its Tryst With Success.* [Online]
Available at: https://analyticsindiamag.com/hugging-face-and-its-tryst-with-success/

Soomro., Z. T., Ilyas., S. H. W. & Yaqub., U., 2020. Sentiment, count and cases: Analysis of twitter discussions during COVID-19 pandemic. *2020 7th International conference on behavioural and social computing (BESC),* pp. pp. 1-4.

spaCy, n.d. *spaCy 101: Everything you need to know.* [Online]
Available at: https://spacy.io/usage/spacy-101
Srijata, C., n.d. Get Geolocation in Python using GeoPy.
stephan, w., 2021. *Interactive SVG World Map Library – svgMap.js.* [Online]
Available at: https://www.cssscript.com/interactive-svg-world-map/
Subhashini, G. & Grishma, S., 2021. Topic Modeling in Natural Language. Vol. 10(06, June-2021).

Suresh, H., 2020. WordClouds: Basics of NLP.
Suvarna, G., 2021. *How to Deal with Missing Data using Python.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/10/how-to-deal-with-missing-data-using-python/

Texthero, n.d. *Text preprocessing, representation and visualization from zero to hero.*. [Online]
Available at: https://texthero.org/

Tutorialspoint, n.d. *Scikit Learn Tutorial.* [Online]
Available at: https://www.tutorialspoint.com/scikit_learn/index.htm

Twitter, n.d. *Tweepy.* [Online]
Available at: https://www.tweepy.org/
[Accessed 2021].

Utkarsh, D., 2020. Encoder Decoder models in HuggingFace from (almost) scratch.
van Erp, G., Rizzo, a. R. & Troncy, 2013.

W3School, n.d. *NumPy Introduction.* [Online]
Available at: https://www.w3schools.com/python/numpy/numpy_intro.asp

WIKIPEDIA, 2008. *pandas (software).* [Online]
Available at: https://en.wikipedia.org/wiki/Pandas_(software)

WIKIPEDIA, 2020. *Python (programming language).* [Online]
Available at: https://en.wikipedia.org/wiki/Python_(programming_language)

WIKIPEDIA, n.d. [Online]
Available at: https://en.wikipedia.org/wiki/Ajax_(programming)

WIKIPEDIA, n.d. *CSS.* [Online]
Available at: https://en.wikipedia.org/wiki/CSS

WIKIPEDIA, n.d. *Django (web framework).* [Online]
Available at: https://en.wikipedia.org/wiki/Django_(web_framework)

WIKIPEDIA, n.d. *HTML.* [Online]
Available at: https://en.wikipedia.org/wiki/HTML

WIKIPEDIA, n.d. *PyCharm.* [Online]
Available at: https://en.wikipedia.org/wiki/PyCharm

WIKIPEDIA, n.d. *Visual Studio Code.* [Online]
Available at: https://en.wikipedia.org/wiki/Visual_Studio_Code

Yong, C. et al., 2019. Experimental explorations on short text topic mining between LDA and NMF based Schemes. *Sciencedirect,* Volume 163, pp. 1-13.

Appendix A: Libraries require installation for this project

pip install snscrape

pip install tweepy

pip install alive-progress

pip install texthero

pip install geopy

pip install transformers

pip install --user -U nltk

pip install spacy

pip install pycountry-convert

pip install country_converter --upgrade

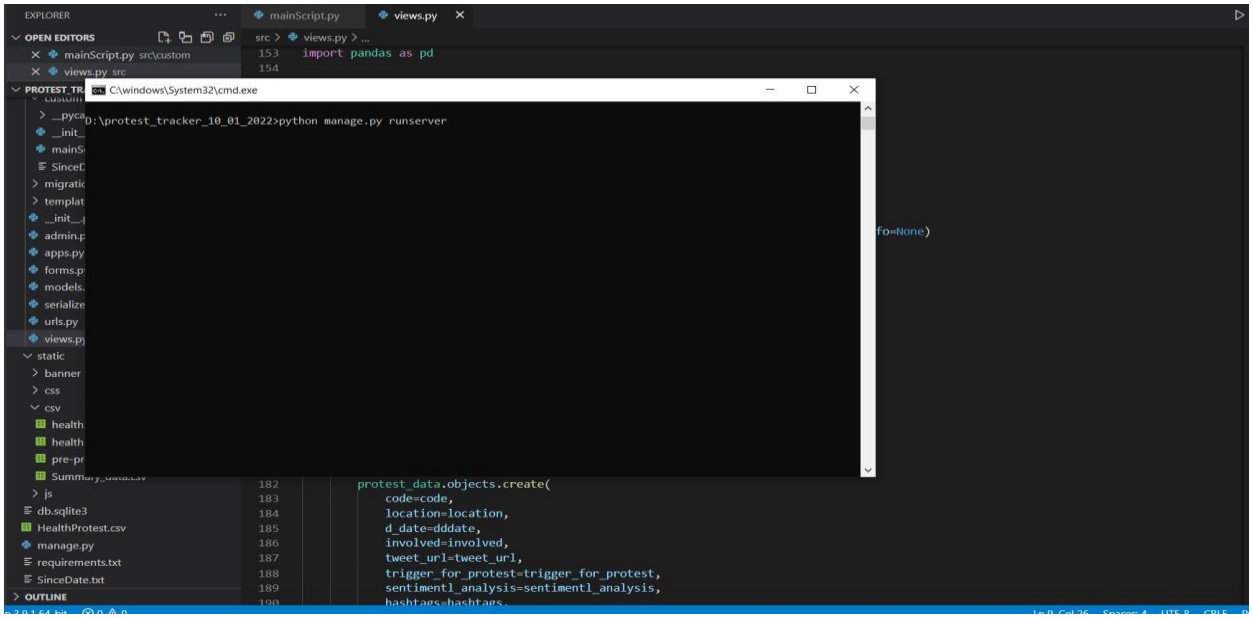
pip install tensorflow

pip install djangorestframework

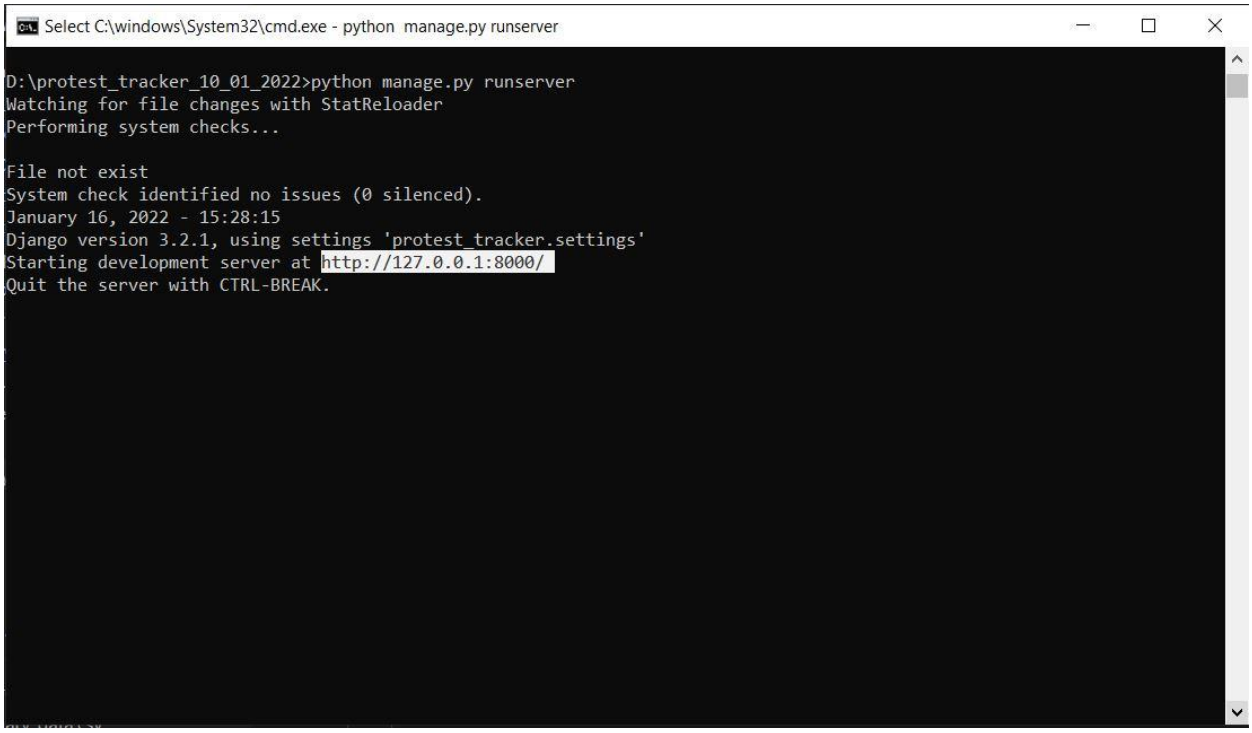
How to run the Project:

The below steps are to run Web application however the data collection script will run automatically everyday with the help of the Windows Task scheduler. Also, to run it manually simply run it as a script in the project.

1. Open Project and click Ctrl +Shift+ C on same window. It will open command prompt. Then type python manage.py runserver shown as below. It will run the project on localhost.



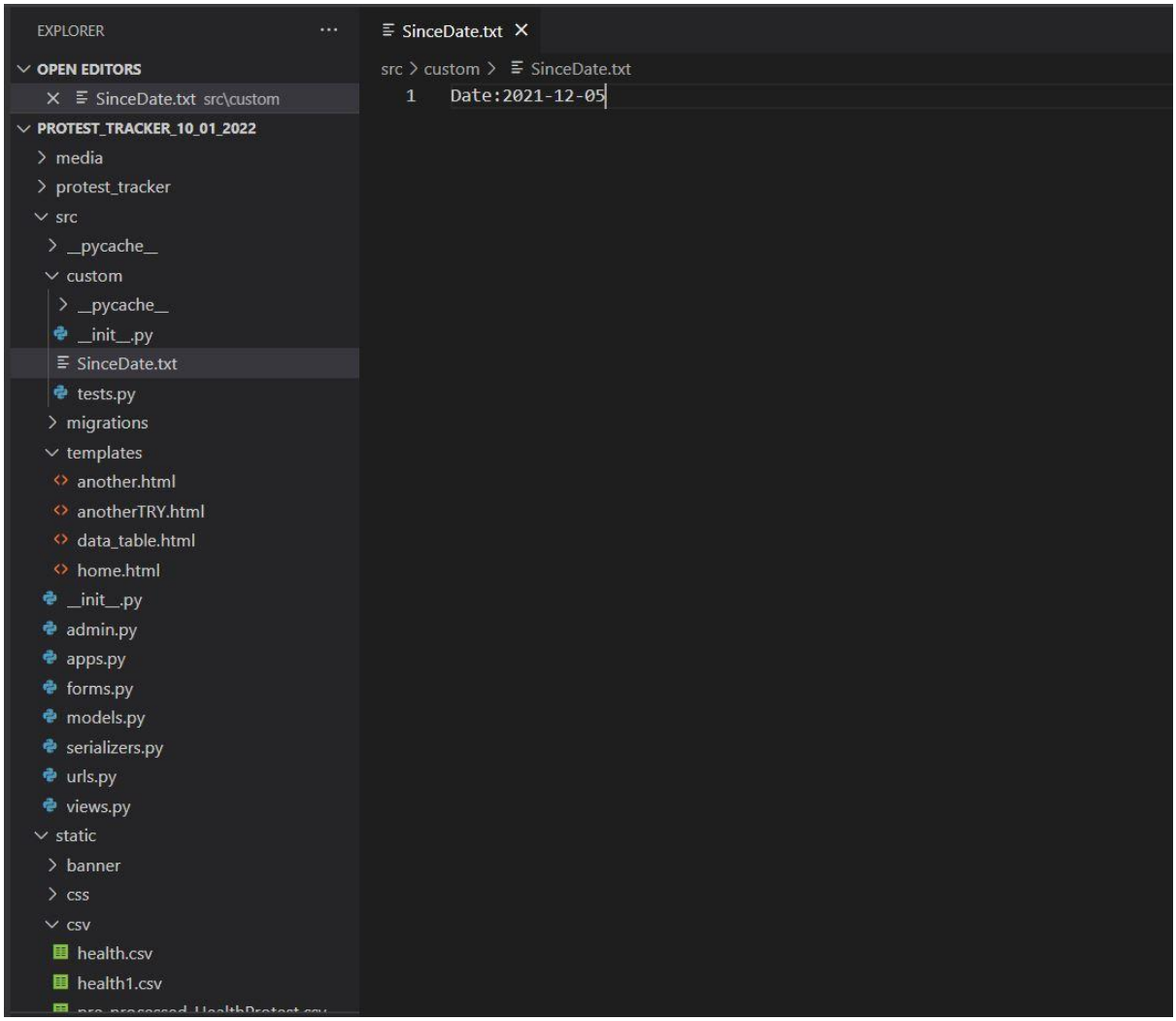
2. Press Enter and Copy that http link paste it into a browser. It will open a web Application as shown below.



Appendix B: Task Automation

Saved Date:

In this file date has been saved while collecting tweets data from tweet to collect data next time this date parameter will be passed to the tweepy API as a parameter (Since) automatically.

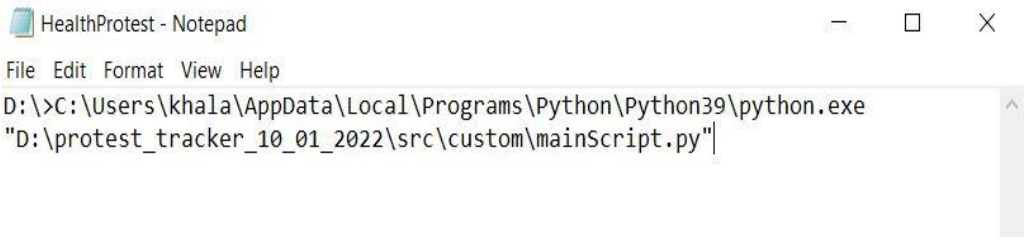


Steps to Automate the Task of Python script Using Window Task Scheduler:

1. Create Python Executable Files (bat file)

   In this Project I have created one python script **mainScript.py** which includes every task require to prepare dataset. Then from this python script BAT file has been created. BAT file is a DOS batch file that may be used with the Windows Command

Prompt to run commands (cmd.exe). It comprises a set of line commands that would normally be typed into the DOS command prompt. BAT files are extensively used in Windows to start programmes and conduct maintenance routines (Vincent, 2019).

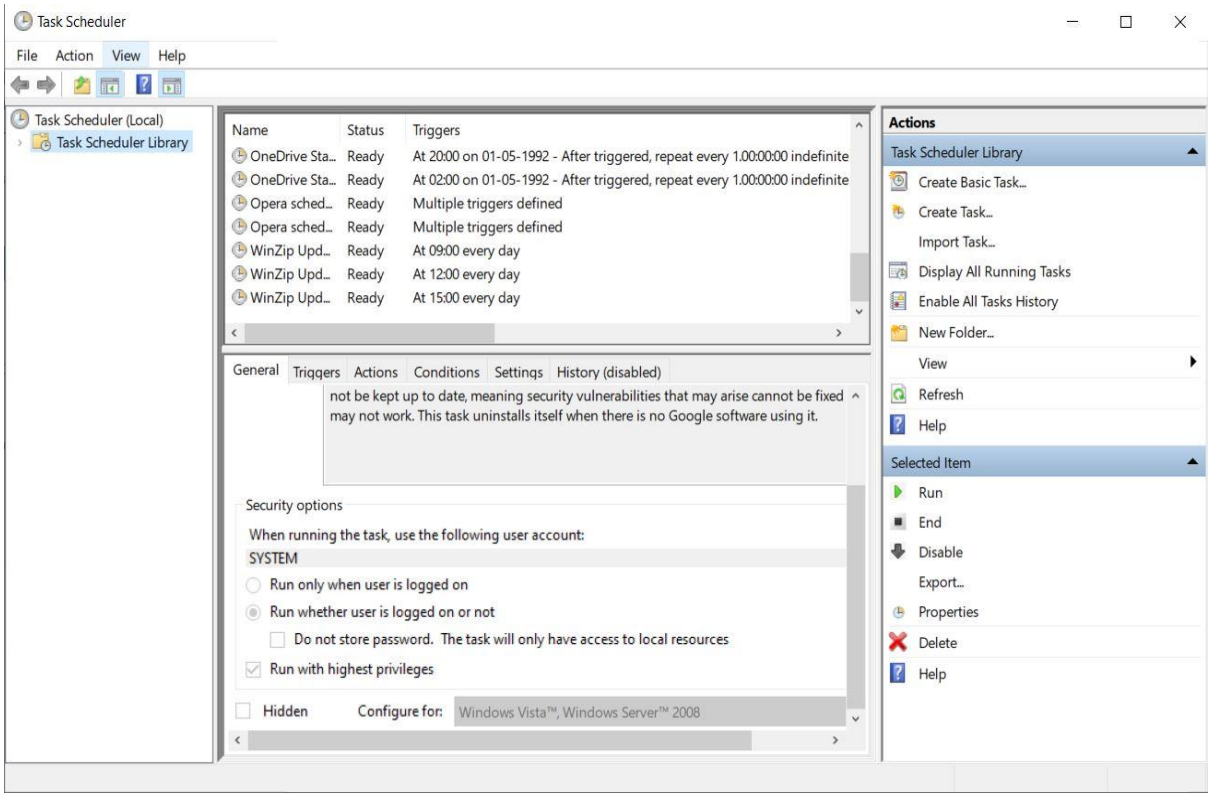


bat file (e.g: HealthProtest.bat) the executable script with the format of <Our Python.exe Location> <Our python Scripts Location>. Here, change the mainscript.py path as per the location of the folder in the local file system.

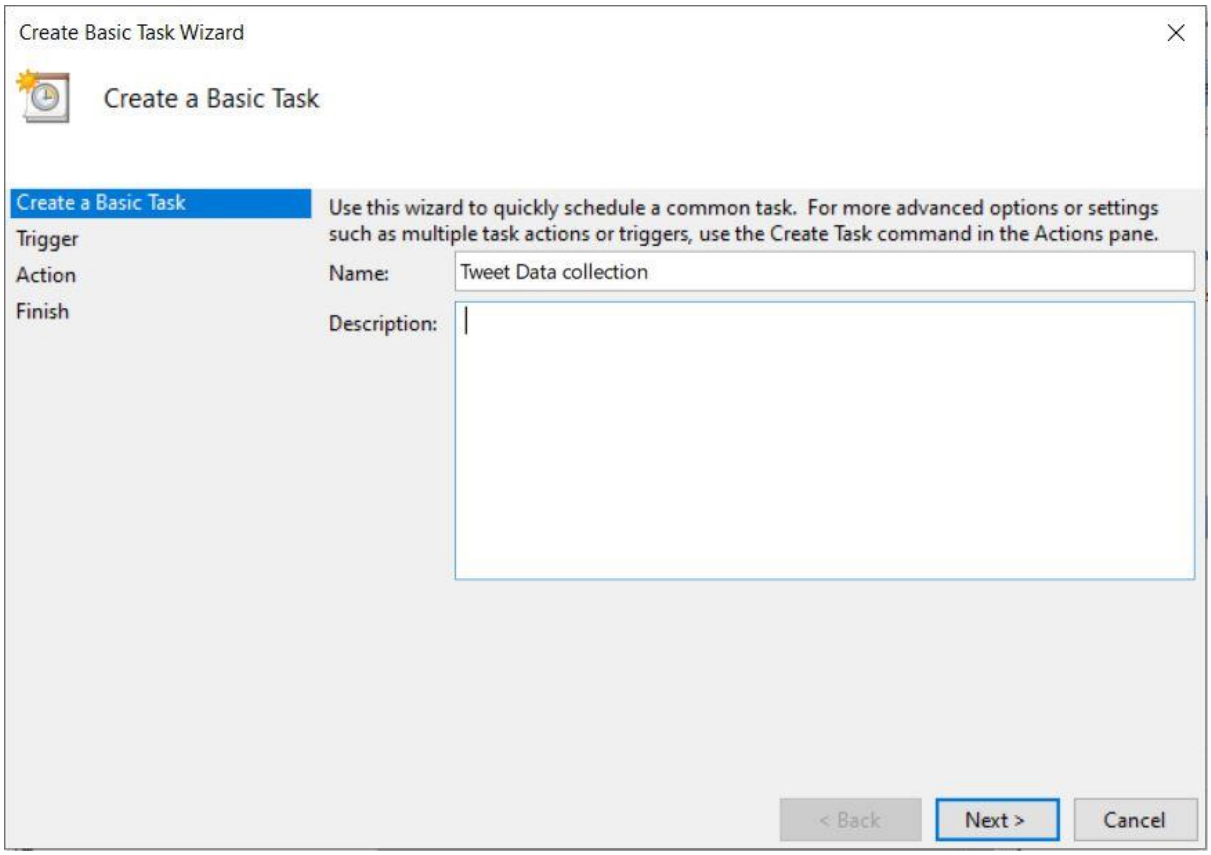
2. Configure Task in Windows Task Scheduler

   The below are the step to configure the created BAT file to Windows task Scheduler:

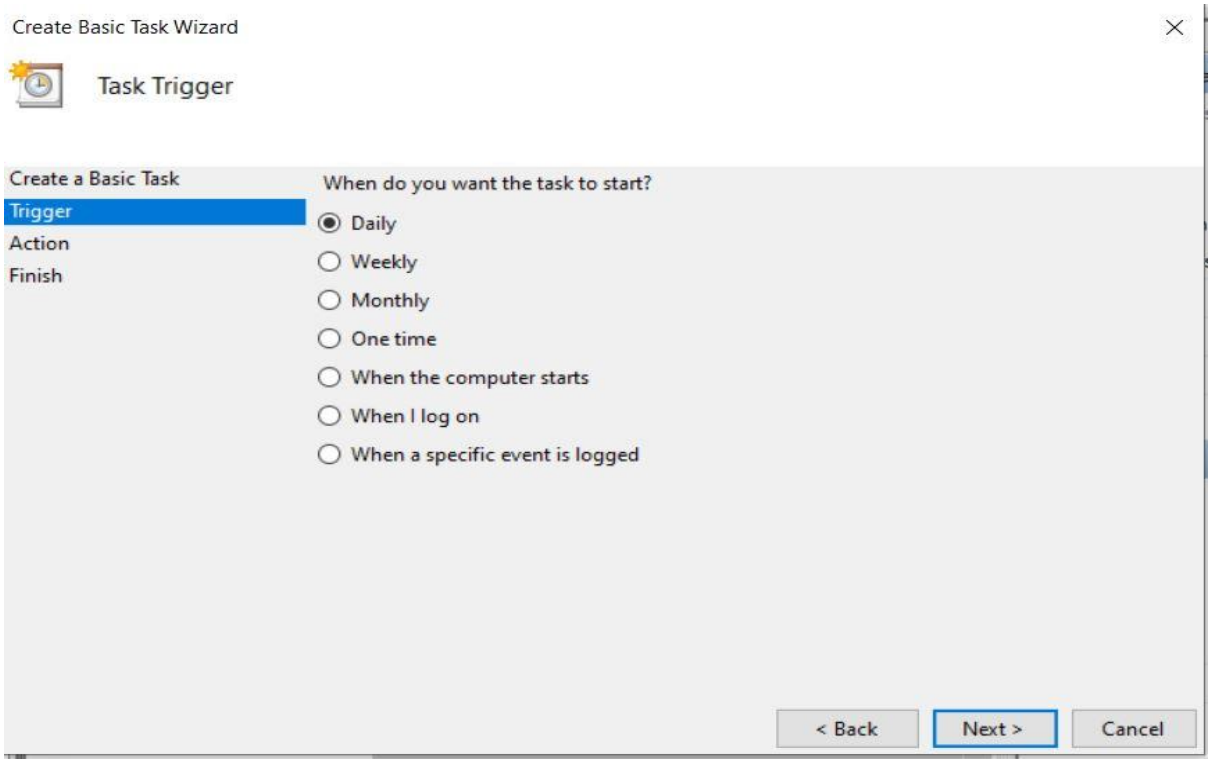
   1. Search for Task Scheduler in Start Windows and open it.



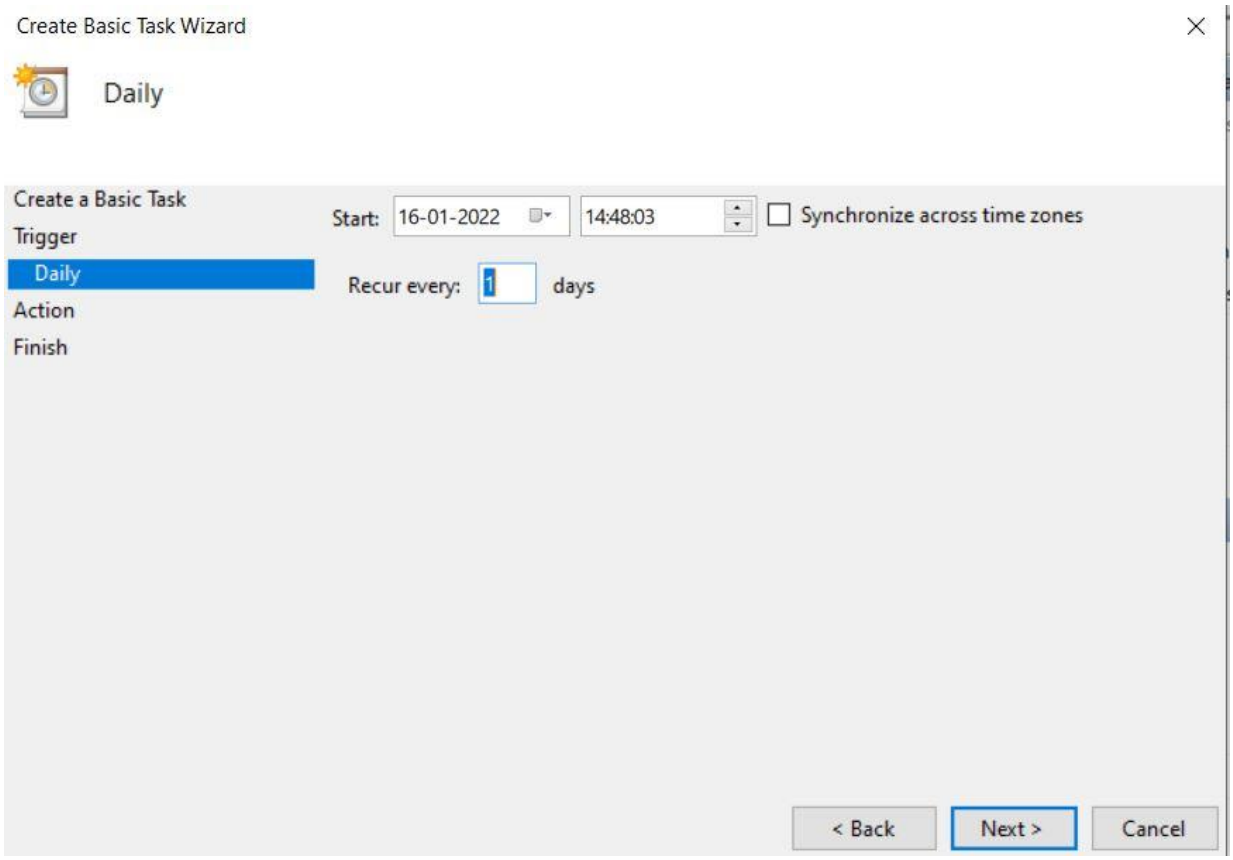
3. In the right box, click Create Basic Task.



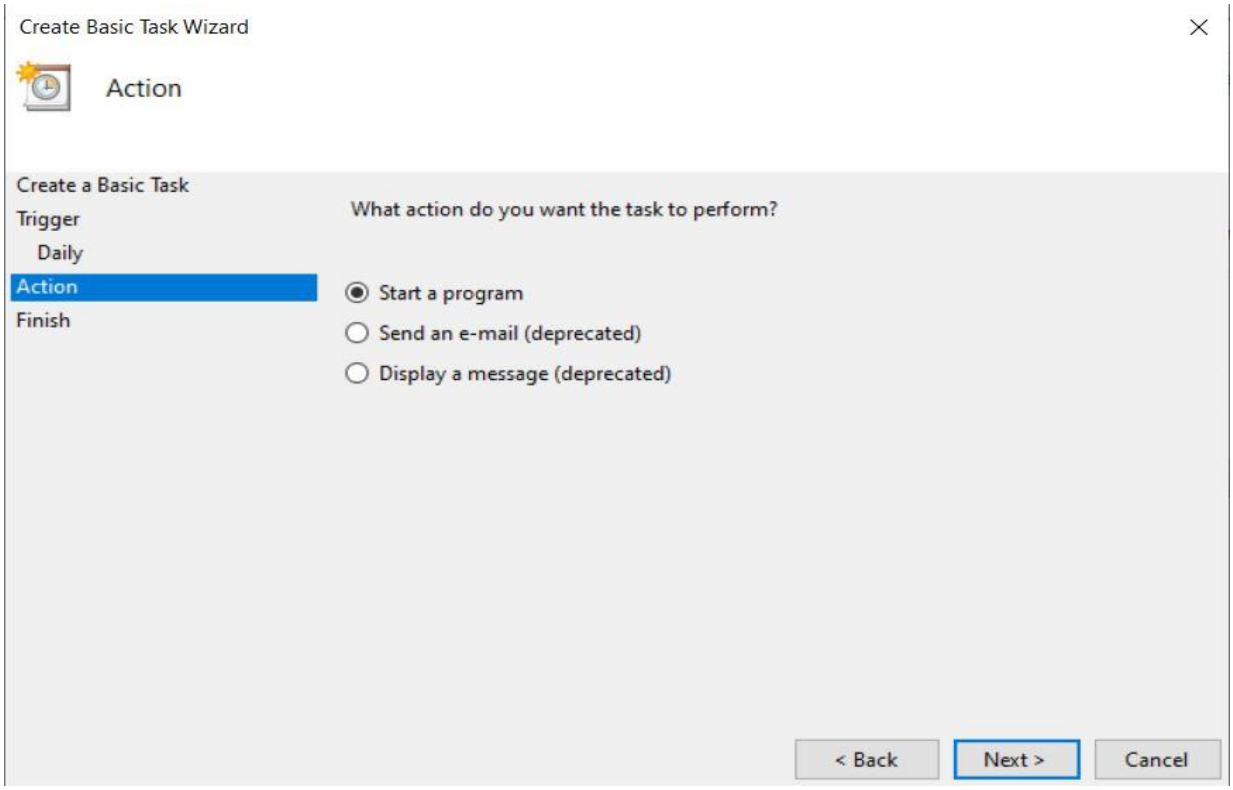
4. Pick a time for your trigger.



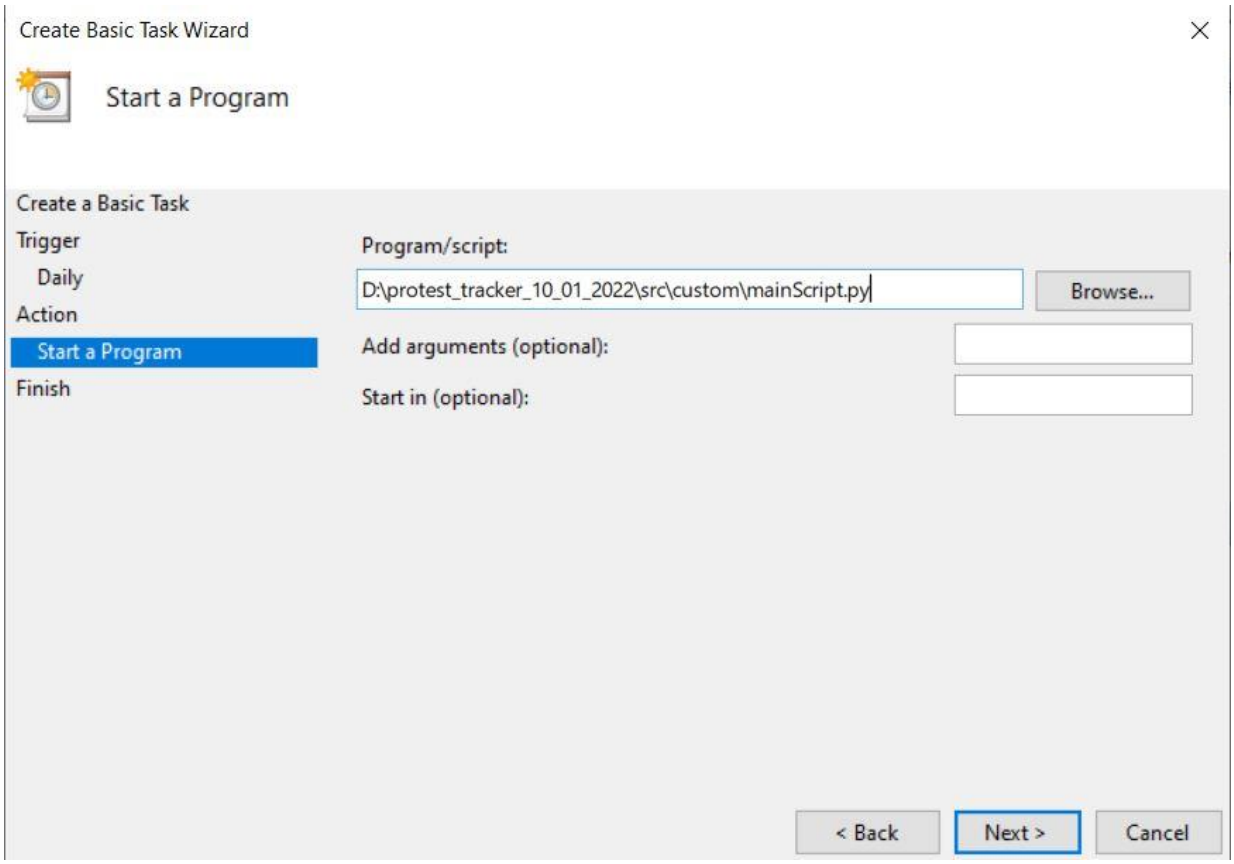
5. For our previous option, choose the exact time.



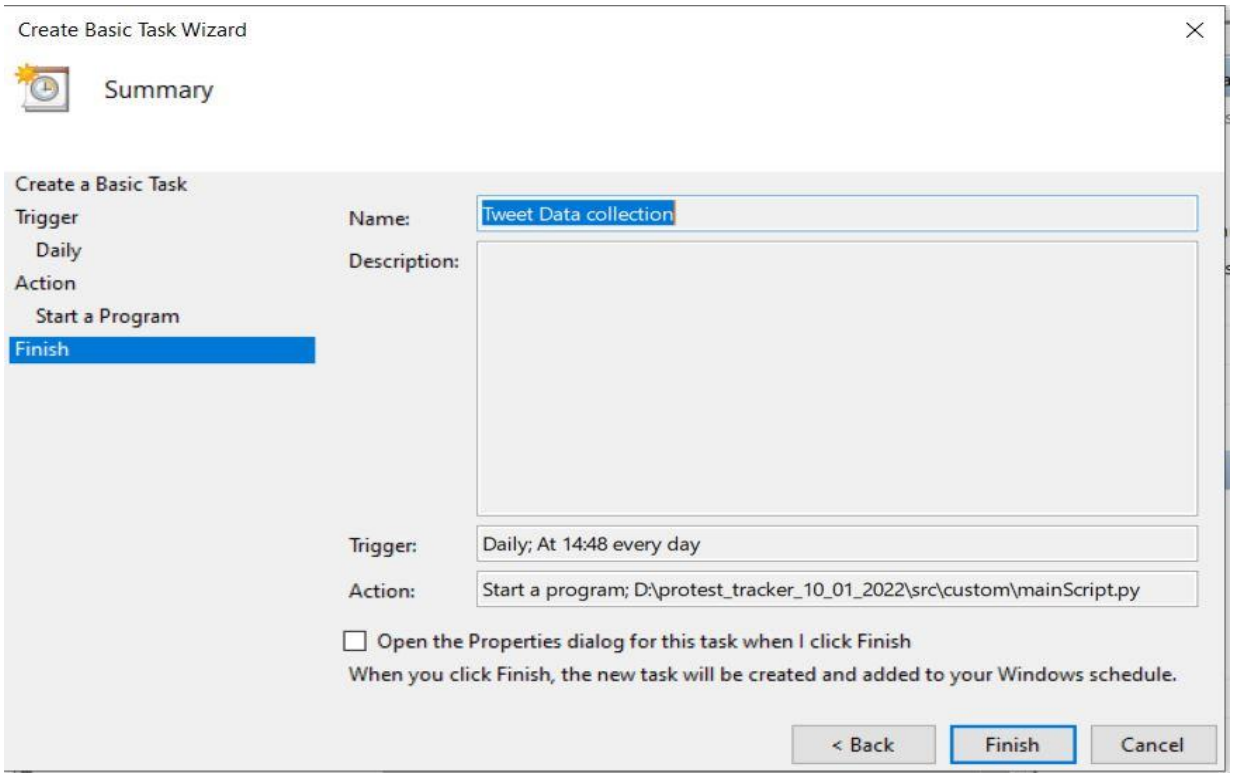
6. Begin a programme.



7. Wherever you stored your bat file earlier, paste your programme script.



8. Finish by clicking the Finish button.



Appendix C: Front-end Code Implementation

Front-end code of Web Application:

```
{% load static %}
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78i

    <!-- JQUERY CDNS -->
    <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

    <!-- JQUERY UI CDNS -->

    <!-- MAP HEAD -->
    <script src="https://cdn.jsdelivr.net/npm/svg-pan-zoom@3.6.1/dist/svg-pan-zoom.min.js"></script>
    <link href="{% static 'css/demo.css' %}" rel="stylesheet" />
    <link href="{% static 'css/svgMap.css' %}" rel="stylesheet" />
    <script src="{% static 'js/svgMap.js' %}"></script>
  <!-- END MAP HEAD -->

<style>
    .mainptrn{
        font-size: 25px;
    }
    .tbl{
        font-size: 20px;
    }
</style>
  <!-- NAVBAR START FORM HERE -->

  <nav class="navbar navbar-expand-lg p-3" style="background:#092e55">
    <div class="container-fluid">
        <a class="navbar-brand text-white fw-bold mainptrn" href="{% url 'home' %}">Health Workers Protest Tracker</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="n
            <span class="navbar-toggler-icon"></span>
          </button>
          <ul>
            <li>
                <a class="nav-link text-white mt-1 tbl" href="{% url 'get-search' %}">Table</a>
              </li>

          </ul>


    </div>
</nav>


<!-- NAVBAR END HERE -->


</head>
  <body>

        <!-- ALL MAP START FROM HERE  -->
        <div class="container">

            <div id="carouselExampleIndicators" class="carousel slide mt-3 mb-3 shadow" data-bs-ride="carousel">
                <div class="carousel-indicators">
                  <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="0" class="active" aria-current="tr
                  <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="1" aria-label="Slide 2"></button>
                  <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="2" aria-label="Slide 3"></button>
                </div>
```

```
            <div class="carousel-inner">
              <div class="carousel-item active">
                <img src="{% static 'banner/Protest.jpg'%}" class="d-block w-100" alt="...">
              </div>
              <div class="carousel-item">
                <img src="{% static 'banner/image4.jpg'%}" class="d-block w-100" alt="...">
              </div>
              <div class="carousel-item">
                <img src="{% static 'banner/image3.jpg'%}" class="d-block w-100" alt="...">
              </div>
            </div>
            <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide="prev">
              <span class="carousel-control-prev-icon" aria-hidden="true"></span>
              <span class="visually-hidden">Previous</span>
            </button>
            <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide="next">
              <span class="carousel-control-next-icon" aria-hidden="true"></span>
              <span class="visually-hidden">Next</span>
            </button>
          </div>

          <style>
                #sme {
                width: 400px;
                animation: rotatebox 2s ease forwards;
                animation-play-state: paused;
                }

                #sme:hover {
                animation-play-state: running;
                }

                @keyframes rotatebox {
                100% {
                    transform: rotate(180deg);
                }
                }
          </style>


                <h2>Data Visulisation on Interactive MAP</h2>

                <div id="svgMapEuroCurrency" class="shadow mb-5"></div>
                <script>

                function add() {
                parentDict = {}
                let token = "{{csrf_token}}";
                $.ajax({
                    url: "{% url 'get-data'  %}",
                    method: "POST",
                    headers: {
                        "X-CSRFToken": token
                    },
                    data: {

                        'country': "India",

                    },
                    success: (data) => {
                        if (Array.isArray(data)) {

                            const search_result = document.getElementById("finaldata");

                            data.forEach(company => {
                                    an_data = {
                                        "euro": 1,
                                        "eurozone": 1,
                                        "color": "#528FCC",
                                        'location': company.location,
                                        'trigger': company.trigger_for_protest,
                                        'date': company.d_date,
```

```
                                    }
                                    parentDict[company.code] = an_data

                                }) // end for loop


                            } // end array

                        }, // end success

                    }); // end ajax

                    values = parentDict
                    return values
                }


                    var svgMapEuroCurrency = new svgMap({
                        targetElementID: 'svgMapEuroCurrency',
                        data: {
                            data: {
                                location:{
                                    name:"location"
                                },

                                euro: {
                                    name:"euro"
                                }
                            },
                            applyData: 'euro',
                            values: add(),
                        },
                        colorMin: '#E2E2E2',
                        colorMax: '#297ACC',
                        colorNoData: '#E2E2E2',
                        thresholdMax: 1,
                        thresholdMin: 0,
                        initialZoom: 3,
                        initialPan: {
                            x: 420,
                            y: 50
                        },
                        mouseWheelZoomEnabled: true,
                        mouseWheelZoomWithKey: true,
                        onGetTooltip: function(tooltipDiv, countryID, countryValues) {
                            // Geting the list of countries
                            var countries = svgMapEuroCurrency.countries;

                            // Create tooltip content element
                            var tooltipContentElement = document.createElement('div');
                            tooltipContentElement.style.padding = '16px 24px';

                            // Fill content
                            var innerHTML =
                                '<div style="margin: 0 0 10px; text-align: center"><img src="https://cdn.jsdelivr.net/gh/hjnilsson/c
                                    '{0}',
                                    countryID.toLowerCase()
                                );

                            innerHTML +=
                                '<div style="min-width: 180px; font-weight: bold; margin: 0 0 15px; text-align: center">' +
                                countries[countryID] +
                                '</div>';

                            if (countryValues && countryValues.location == countries[countryID]){
                                $("#onabout").one("mouseover", function() {
                                    $("#onabout ul").addClass('permahover');
                                    });

                            values1=newadd()

                            //   TRIGGER
                            function newadd() {
```

```
function newadd() {
                parentDict = []
                let token = "{{csrf_token}}";
                $.ajax({
                    url: "{% url 'get-card-data'  %}",
                    method: "POST",
                    headers: {
                     "X-CSRFToken": token
                    },
                    data: {

                        'country': countryValues.location,

                 },
                 success: (data) => {
                       if (Array.isArray(data)) {

                           data.forEach(company => {

                               data={
                                   "date": company.d_date,
                                   "name":company.involved,
                                   "trigger": company.trigger_for_protest,



                               }


                               // ${key}: ${data[key]}
                               for (var key in data) {
                                   // check if the property/key is defined in the object itself
                                   console.log(key[0,5])
                               innerHTML +=`
                               <div id="sme">
                                       <span class="badge bg-secondary">${key}</span></h6>
                                       ${data[key]}


                               </div>  `
                               }

                               parentDict.push(data)


                               tooltipContentElement.innerHTML = innerHTML;
                               return tooltipContentElement;

                               })

                               // end for loop


                       } // end array

                   }, // end success

                }); // end ajax

                values1 = parentDict
                return values1
            }
// END TRIGGER




}
else{
    innerHTML +=
    '<div style="min-width: 180px; font-weight: bold; margin: 0 0 15px; text-align: center">' + "No Prot
    '</div>';
}
```

```
                        tooltipContentElement.innerHTML = innerHTML;
                        return tooltipContentElement;
                    }
                });
            </script>
        </div>

    </body>

    </html>


    <!-- ALL MAP END HERE -->


    <!-- table start from here -->

    <style>
        .needdiv{
            width: 250%;
            height: 100%;
            background-color: #f2f2f2;
            border-radius: 5px;
            padding: 20px;
            margin-top: 20px;
            position: relative;
            margin-left:150px;

        }
    </style>

<div>
    <nav class="navbar navbar-expand-lg navbar-light bg-light mb-3">
        <div class="container-fluid">
            <a class="navbar-brand" href="#"></a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-con
                <span class="navbar-toggler-icon"></span>
              </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav me-auto mb-2 mb-lg-0">

                </ul>
                Filter protests by
                <a class="btn btn-primary mx-2" href="{% url 'home'  %}" role="button">All</a>
                <form class="d-flex mx-2" method="post" action="{% url 'home'  %}">
                    {% csrf_token %}
                    {{form}}
                    <button class="btn btn-outline-success" type="submit" id="search_button" >Filter</button>
                </form>
                <form class="d-flex mx-2" method="post" action="{% url 'home'  %}">
                    {% csrf_token %}
                    <input class="form-control me-2" name="location" type="search" placeholder="Search By Location" id="search_locat
                    <button class="btn btn-outline-success" type="submit" id="search_button" >Filter</button>
                </form>

            </div>
        </div>
    </nav>


    <table class="table mb-5" id="mytable">
        <thead>
            <tr>
                <th scope="col"></th>
                <th scope="col">Date</th>
                <th scope="col">Location</th>
```

```
                <th scope="col">Trigger</th>


                <th scope="col">Sentiment Analysis</th>

            </tr>
        </thead>

        <tbody id="get_body">

            {% for i in all_data %}

            <tr id=#{{i.id}}>
                <td>
                    <p>
                        <!-- <a class="btn btn-primary" data-bs-toggle="collapse" href="#collapseExample" role="button" aria-expande
                            +L
                        </a> -->
                        <button class="btn btn-primary" type="button" data-bs-toggle="collapse" data-bs-target="#data{{i.id}}" aria-
                            +
                        </button>
                    </p>



                    <div class="collapse needdiv" id="data{{i.id}}">

                        <div class="shadow p-3 mt-2 mb-3">

                            <p>
                                <!-- <b>involved</b>
                                {{i.involved}} -->

                                <p>
                                    <b>Tweet URL:</b>

                                    <a class="btn btn-primary" href="{{i.tweet_url}}" role="button">Tweet URL</a>
                                    <a class="btn btn-primary" href="{{i.tweet_url}}" role="button">Tweet URL</a>
                                </p>

                                <p>
                                    {% if  i.media_url == "No Media Available"   %}
                                    <b>Media URL:</b>
                                    <a class="btn btn-primary"  role="button">No Media Available</a>
                                    {% else %}
                                    <b>Media URL:</b>
                                    <a class="btn btn-primary" href="{{i.media_url}}" role="button">Media URL</a>
                                    {% endif %}
                                </p>


                                <p>
                                    <b>Hashtags:</b>
                                    {{i.hashtags}}
                                </p>


                                <p>
                                    <b>Retweets count:</b>
                                    {{i.retweets_count}}

                                </p>

                                <p>
                                    <b>Favorites count:</b>
                                    {{i.likes_count}}
                                </p>

                            </p>
```

```
                        </div>

                    </td>


                <td>{{ i.d_date}} </td>
                <td>{{ i.location}} </td>

                <td>{{ i.involved}} </td>

                <!-- <td>
                    <a class="btn btn-primary" href="{{ i.tweet_url}}" role="button">tweet</a>
                </td> -->
                <td>{{ i.trigger_for_protest}} </td>


                <td>{{ i.sentimentl_analysis}} </td>

                <td>
                    {% empty %}
                    No Data Available
                </td>


            </tr>




            {% endfor %}



        </tbody>
    </table>
    <style>
        .mybutton{ margin-left: 376px;

        }
    </style>
    <div class="row justify-content-center">
        <div class="col-4 mb-5 mybutton">
            <a class="btn btn-primary" href="{% url 'export_users_csv' %}" role="button">Download CSV</a>
        </div>

    </div>



    <!-- <style>
       .hidden{
           display: none;
       }
    </style> -->

    </div>


    <!-- table end here -->


<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
<script src="https://code.jquery.com/ui/1.13.0/jquery-ui.js"></script>
<script>
        //// get location ######
                $(function()
                {
                    $("#search_location").autocomplete({
                        source: "{% url  'get-country' %}"
                                                    });
```

```
                    });
                //// end

</script>



            <!-- bootstrap js -->


    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-7+zCNj/IqJ95wo16oMtfsKbZ9c
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+Z
    -->
  </body>
</html>
```

Appendix D: Back-end code of Web Application

src > views.py > ...

```python
import json
from django.shortcuts import render
from .models import *
from django.http import  JsonResponse
import json
from .serializers import ConvertSerializer
from datetime import datetime, timedelta
from datetime import date
from .forms import MyForm
from django.http import HttpResponse
import csv

def home(request):
    try:
        location = request.POST.get('location')
        ddate = request.POST.get('d_date')

        request.session['data']=''
        request.session['data_d']=''

        all_data=protest_data.objects.all().order_by('location')
        if location:
            request.session['data'] = str(location)
            all_data=protest_data.objects.filter(location__icontains=location)


        if ddate:
            request.session['data_d'] = ddate
            all_data=protest_data.objects.filter(d_date=ddate)

        context={

        'all_data':all_data,
        'form':MyForm()

        }
        return render(request, 'anotherTRY.html', context)
    except Exception as e:
        print(e)
        return JsonResponse({'error': str(e)})
def make_thing(request):
    all_data=protest_data.objects.all()
    context={
        'all_data':all_data
    }
    return render(request, 'another.html', context)

def get_data(request):
    if request.method == 'POST':
        country_name=request.POST.get('country')
        all_data=protest_data.objects.all()
        convert=ConvertSerializer(all_data, many=True)
        return JsonResponse(convert.data, safe=False)



def get_country(request):
    if 'term' in request.GET:
        qs = protest_data.objects.filter(location__icontains=request.GET.get('term'))
        companys = list()
        for comp in qs:
            companys.append(comp.location)
        return JsonResponse(companys, safe=False)
    return render(request, 'another.html')




from pytz import timezone
def get_search(request):
    try:
        location = request.POST.get('location')
        ddate = request.POST.get('d_date')

```

```
        request.session['data']=''
        request.session['data_d']=''
        check_var=''

        all_data=protest_data.objects.all().order_by('location')
        if location:
            request.session['data'] = str(location)
            all_data=protest_data.objects.filter(location__icontains=location)


        if ddate:
            request.session['data_d'] = ddate
            all_data=protest_data.objects.filter(d_date=ddate)


        if not all_data:
            check_var='NO_DATA'



        context={

        'all_data':all_data,
        'form':MyForm(),
        'check_var':check_var

        }


        return render(request, 'data_table.html',context)

    except Exception as e:
        print(e)
        return JsonResponse({'error': str(e)})


def get_card_data(request):
    try:
        location = request.POST.get('country')
        all_data=protest_data.objects.filter(location__icontains=location)
        convert=ConvertSerializer(all_data, many=True)
        return JsonResponse(convert.data, safe=False)
    except Exception as e:
        print(e)
        return JsonResponse({'error': str(e)})

def another_try(request):
    return render(request, 'anotherTRY.html')




def export_users_csv(request):
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="users.csv"'

    writer = csv.writer(response)
    writer.writerow(['code','location', 'd_date', 'type_of_action', 'involved', 'tweet_url', 'trigger_for_protest', 'size_of_protest', '

    all_data =protest_data.objects.all()
    print("type",type(all_data))

    alldata=request.session.get('data')
    d_data=request.session.get('data_d')
    if alldata:
        users=protest_data.objects.filter(location__icontains=alldata).values_list('code','location', 'd_date', 'type_of_action', 'invol
    elif d_data:
        users=protest_data.objects.filter(d_date=d_data).values_list('code','location', 'd_date', 'type_of_action', 'involved', 'tweet_u
    else:
        users = protest_data.objects.all().values_list('code','location', 'd_date', 'type_of_action', 'involved', 'tweet_url', 'trigger_
```

```
145
146     for user in users:
147         writer.writerow(user)
148
149     return response
150
151
152
153 import pandas as pd
154
155
156
157
158 df = pd.read_csv('static/csv/HealthProtest_Data.csv')
159
160 for f in range(len(df)):
161
162     ddate=df.loc[f]['Datetime']
163
164     # since = df['Datetime'].iloc[0]
165     # since=datetime.strptime(str(ddate), '%Y-%m-%d %H:%M:%S+00:00').replace(tzinfo=None)
166     since=datetime.strptime(str(ddate), '%Y-%m-%d %H:%M:%S')
167     print(since)
168     dddate=since.strftime('%Y-%m-%d')
169     print(dddate)
170
170
171     code=df.loc[f]['Country_Code']
172     location=df.loc[f]['Country']
173     involved=df.loc[f]['name']
174     tweet_url=df.loc[f]['TweetUrl']
175     trigger_for_protest=df.loc[f]['clean_text']
176     sentimentl_analysis=df.loc[f]['sentiment']
177     hashtags=df.loc[f]['Hashtags']
178     media_url=df.loc[f]['Media']
179     retweets_count=df.loc[f]['Retweets Counts']
180     likes_count=df.loc[f]['Like Counts']
181     print(dddate,location)
182     protest_data.objects.create(
183         code=code,
184         location=location,
185         d_date=dddate,
186         involved=involved,
187         tweet_url=tweet_url,
188         trigger_for_protest=trigger_for_protest,
189         sentimentl_analysis=sentimentl_analysis,
190         hashtags=hashtags,
191         retweets_count=retweets_count,
192         likes_count=likes_count,
193         media_url=media_url,
194     )
195
196
```

Appendix E: NER code implementation

E.1. NER Extract Organisation:

```
print('==============================')
print('Getting Organization.. .. ')

def get_organization(text):
  if type(text)==str and len(text)>10:
    sents = en(text)
    value = [str(ee) for ee in sents.ents if ee.label_ == 'ORG']
    value=list(set(value))
    if len(value)>0:
      return value
    else :
      return 'null'
  else:
    return 'null'
############  Function call get_organization  ##############
tweets_df["Organization"] = tweets_df["clean_text"].apply(lambda text: get_organization(text))
```

E.2. NER Extract Protest type:

```
print('==============================')
print('Getting Type of Protest.. .. ')

def get_Event(text):
  if type(text)==str and len(text)>10:
    sents = en(text)
    value = [str(ee) for ee in sents.ents if ee.label_ == 'EVENT']
    value=list(set(value))
    if len(value)>0:
      return value
    else :
      return 'null'
  else:
    return 'null'
############  Function call get_organization  ##############
tweets_df["Protest Type"] = tweets_df["clean_text"].apply(lambda text: get_Event(text))
```

E.3. NER Extract NORP:

```
print('==============================')
print('Getting nationalities, religious, or political groups involved.. .. ')

def get_AnyGroupInvolved(text):
  if type(text)==str and len(text)>10:
    sents = en(text)
    value = [str(ee) for ee in sents.ents if ee.label_ == 'NORP']
    value=list(set(value))
    if len(value)>0:
      return value
    else :
      return 'null'
  else:
    return 'null'
############  Function call get_AnyGroupInvolved  ##############
tweets_df["NORP"] = tweets_df["clean_text"].apply(lambda text: get_AnyGroupInvolved(text))
```

E.4. NER Extract Protest Size:

```
print('==============================')
print('Getting Protest Size.. .. ')
def get_ProtestSize(text):
  if type(text)==str and len(text)>10:
    sents = en(text)
    value = [str(ee) for ee in sents.ents if ee.label_ == 'QUANTITY']
    value=list(set(value))
    if len(value)>0:
      return value
    else :
      return 'null'
  else:
    return 'null'
############  Function call get_ProtestSize  ##############
tweets_df["Protest-Size"] = tweets_df["clean_text"].apply(lambda text: get_ProtestSize(text))
```

E.5. NER Extract Health worker Involved:

```
print('==============================')
print('Getting person name.. .. ')

def get_Person_nameSpacy(text):
  if type(text)==str and len(text)>10:
    sents = en(text)
    value = [str(ee) for ee in sents.ents if ee.label_ == 'PERSON']
    value=list(set(value))
    if len(value)>0:
      return value
    elif 'doctor' in text:
      return 'doctor'
    elif 'nurse' in text:
      return 'nurse'
    elif 'Healthworker' in text:
      return 'Healthworker'
  else:
    return 'null'

tweets_df["name"] = tweets_df["clean_text"].apply(lambda text: get_Person_nameSpacy(text))
```

E.6. NER Extract Date:

```
print('==============================')
print('Getting Date from Tweet Text.. .. ')
def get_newDATEInText(text):
  if type(text)==str:
    sents = en(text)
    value = [str(ee) for ee in sents.ents if ee.label_ == 'DATE']
    value=list(set(value))
    if len(value)>0:
      return value
    else :
      return 'null'
  else:
    return 'null'
############  Function call get_organization  ##############
tweets_df["NEW_DATE"] = tweets_df["clean_text"].apply(lambda text: get_newDATEInText(text))
```

E.7. NER Extract GPE:

```
print('==============================')
print('Getting GPE_Location from Tweet Text.. .. ')
def get_LocationInText(text):
  if type(text)==str:
    sents = en(text)
    value = [str(ee) for ee in sents.ents if ee.label_ == 'GPE' or ee.label_ == 'LOC']
    value=list(set(value))
    if len(value)>0:
      return value
    else :
      return np.NaN
  else:
    return np.NaN
############  Function call get_organization  ##############
tweets_df["GPE"] = tweets_df["clean_text"].apply(lambda text: get_LocationInText(text))
```

E.8. NER Extract LOC:

```
print('==============================')
print('Getting Location from Tweet Text.. .. ')
def get_newLocationInText(text):
  if type(text)==str:
    sents = en(text)
    value = [str(ee) for ee in sents.ents if ee.label_ == 'LOC']
    value=list(set(value))
    if len(value)>0:
      return value
    else :
      return np.NaN
  else:
    return np.NaN
############  Function call get_organization  ##############
tweets_df["LOC"] = tweets_df["clean_text"].apply(lambda text: get_newLocationInText(text))
```

E.9. Code for sentiment analysis:

```
# print('==============================')
# print('Sentiment-analysis started.. .. ')

model = pipeline("sentiment-analysis")
def perdict(text):
  text=model(text)
  x=text[0]
  return x.get('label')

tweets_df["sentiment"] = tweets_df["clean_text1"].apply(lambda text: perdict(text))
```

E.10. Code for Topic Modelling:

```
def extract_topic(text,n_top_words):
  q=(text,)
  value={}
  lda = LatentDirichletAllocation(n_components=1, max_iter=5,
                                  learning_method='online',
                                  learning_offset=50.,
                                  random_state=0)
  tf_vectorizer = CountVectorizer(
                                  max_features=50,
                                  stop_words='english')
  tf = tf_vectorizer.fit_transform(q)
  tf_feature_names = tf_vectorizer.get_feature_names_out()
  lda.fit(tf)

  for topic_idx, topic in enumerate(lda.components_):
        count=topic_idx
        x=" ".join([tf_feature_names[i]
                    for i in topic.argsort()[:-n_top_words - 1:-1]])
        value.update({count:x})

  return value
```

```
  # print('==============================')
  # print('Extracting Topics.. .. ')

  def get_topics(text):
    if type(text)==str and len(text)>10:

      value =extract_topic(text,10)
      if len(value)>0:
        return value
      else :
        return 'null'
    else:
      return 'null'

  tweets_df["topics"] = tweets_df["clean_text1"].apply(lambda text: get_topics(text))
```

E.11. Code for Text Summarizer:

```
print('==============================')
print('Extract Summary.. .. ')
def get_summry(place):

  data=tweets_df.query("Country==@place")
  data.columns = data.columns.str.replace(' ', '')

  text=''
  for i in data.clean_text.values:
    text=text+i

  summary=summarizer(text, max_length=300, min_length=50, do_sample=False,)

  summary=summary[0]
  summary=summary.get('summary_text')

  date=list(data.Datetime.values)
  text=summary
  tweet_ids=list(data.TweetId.values)
  location=list(data.location.values)
  media=list(data.Media.values)
  url=list(data.TweetUrl.values)
  hastags=list(data.Hashtags.values)
  retweet_count=sum(list(data.RetweetsCounts.values))
  likeCount=sum(list(data.LikeCounts.values))
  Organization=list(data.Organization.values)
  name=list(data.name.values)
  sentiment=list(data.sentiment.values)
  topics=list(data.topics.values)
  geo_country=list(data.geo_country.values)
  Country=list(data.Country.values)

  Country= set(Country)

  d={'Datetime':[date],'TweetId':[tweet_ids],'location':[location],'Media':[media],
     'url':[url],'hastag':[hastags],'retweet_count':retweet_count,'likeCount':likeCount,
     'Organization':[Organization],'name':[name],'sentiment':[sentiment],
     'topics':[topics],'summary':[text],'geo_country':[geo_country],'Country':[Country]}

  dataframe=pd.DataFrame(d)
  return dataframe
```

```
x=tweets_df.Country.unique()

frame=[]
for i in x:
  c=get_summry(i)
  frame.append(c)
final_df=pd.concat(frame)
final_df.to_csv('final_data.csv')
```