SPARK

PROJECT REPORT

Submitted by

Akshath Vadapalli

Akshita Jakhar

Karan Samyal

Prerna Chakraborty

Shashwat Prakash

Shrey Sharma

Vedanta Patil

CSD345

SOFTWARE DESIGN LAB

TABLE OF CONTENTS

DECLARATION

We certify that the work contained in this report is original and has been done by us for the course CSD345 under the supervision of Dr. Snehasis Mukherjee**.**

a. The work has not been submitted to any other sources.

b. We have followed the guidelines provided by the course description.

BONAFIDE CERTIFICATE

This is to certify that the project report titled “Spark” is the bonafide work of Group 16 (members –Akshath, Akshita, Shrey, Karan, Shashwat, Prerna, Vedanta) under the guidance of Dr. Snehasis Mukherjee, submitted as a final project for CSD345, and is their original work.

ACKNOWLEDGEMENT

The success and result of this project required a lot of guidance and endorsement from many people and we are fortunate to get all of these throughout our entire internship project.

We were able to accomplish this project only with such assistance and supervision; therefore, we will never forget to thank them.

We respect and thank **Dr. Snehasis Mukherjee** who guided us throughout the semester on this specific project for the course **CSD345** and gave us all the support that motivated us to complete the project correctly.

Yours Sincerely,

Akshath Vadapalli
Akshita Jakhar
Karan Samyal
Prerna Chakraborty
Shashwat Prakash
Shrey Sharma
Vedanta Patil

ABSTRACT

In order to help the police allocate resources quickly and effectively, and to raise public awareness, we set out to tackle the following problem : - predicting riots and social unrest.

SPARK is an application that uses a predictive model which sets out to analyze and predict social unrest. With the use of an NLP model that uses word2vec to provide word embeddings for vectorization, we will extract data from Twitter (username, location, content, and timeline) and filter it.

We obtained the precise geo-coordinates of tweets using the SNScrape module. Then, we do text categorization and sentiment analysis on the tweets. The maps on our website, made with Folium consist of 3 plots which are describing various aspects like the heat map and clustering of hotspots.

The scraped tweets can be seen on an excel file that is hosted on our website.
With the aid of the three main sentiments that were used in our NLP Model - namely hate, offensiveness, and bad tweets, our software is able to predict riots throughout all of India's states. Furthermore, using any term that the user enters, our programme can operate anywhere in the world, in any nation or state.

INTRODUCTION

SPARK is a website that uses a predictive model which sets out to analyze and predict social unrest in an area.

The goal of this project is to help the Law Enforcement Agencies to identify possible areas of distress from Twitter using tweets that are being made in real time. This project helps in taking a large set of tweets from Twitter using SNScrape, whose results are then taken from the response and saved in an excel file.

How does SPARK work?

The information we require to do our analysis is provided to us by Twitter. We used a data-scraping tool called SNScrape. SNScrape is a scraper for social networking services (SNS). It retrieves the discovered objects, such as pertinent posts, by scraping things like user profiles, hashtags, or searches. We are able to gather data from a certain radius using OpenCage API. OpenCage Geocoding API offers open data-based geocoding that is both reverse (from latitude/longitude to text) and forward (from text to latitude/longitude) on a global scale.

While text classification employs machine learning to analyze unstructured data and extract essential information, sentiment analysis analyzes if an expression is positive, negative, or neutral. Using the data that we have gathered, we use an NLP model to filter out the tweets that have negative sentiment. We used Folium to visualize the data in a number of interactive leaflet maps that are easy to understand.

The aim is to help users evaluate and become aware of areas that are currently a hotspot of negative activity, and SPARK's analysis is expected to make things simple to read and transparent. An excel file that is hosted on our website also contains the tweets that are being used in our analysis. Our programme can function anywhere in the world, in any country or state, utilizing any phrase that the user enters.

How is SPARK useful in predicting social unrest?

Social unrest is a condition where law enforcement struggles to retain control following a widespread act of civil disobedience (such as a protest, riot, strike, or unlawful assembly). Tensions between a crowd and the government might increase for a variety of reasons. Civil unrest is sparked by a variety of factors, including political grievances (including opposition to dictatorial governments), economic disagreements, and historical persecution. In such situations especially, it proves imperative to have a model that helps in analyzing the mood of the society in real time in order for Law Enforcement Agencies to make impactful decisions at the time of civil unrest.

Our application also allows users to find out exactly where there is social distress or a possibility

of it, at that exact time or within a certain time period. The user can enter a certain keyword that they would like to focus their analysis on. The application then collates all the data and uses several graphs and maps to visualize the data in a manner that facilitates reading the data.

Our application has far reaching applications beyond just riot prediction. We can extend our project to topics like pandemics and other such events that have a wide ranging effect on society. The world is always changing and with it so is the public's mood. There's no saying when a change can set society off into civil unrest. Using our application, adding a dynamic analysis feature on top of the existing features could prove to be very helpful in predicting such mood swings in society. Being constantly attentive to the public voice might be useful in managing them when something happens to tick them off.

SPARK uses data which is already available to us and examines it in the hopes to help our society stay a safe place to live and have free speech in. Here is our noble effort hoping to achieve the same.

DESCRIPTION OF TOOL

TOOLS

- **Flask:** Flask uses get and post methods to send form data.
- **TweetNLP:** TweetNLP is a python library tweetnlp provides a collection of useful tools to analyze/understand tweets such as sentiment analysis, emoji prediction, and named entity recognition, powered by state-of-the-art language models specialized on Twitter.
- **Snscrape:** Snscrape uses twitter advanced search to filter through searches.
- **Pandas:** Pandas is used to create an excel file with all respective tweets.
- **Folium:** Folium is used to create static maps.
- **Altair:** Plotting and a mathematical visualization statistical tool.
- **Matplotlib:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- **Opencage:** Opencage is used to get areas under consideration.

LANGUAGES

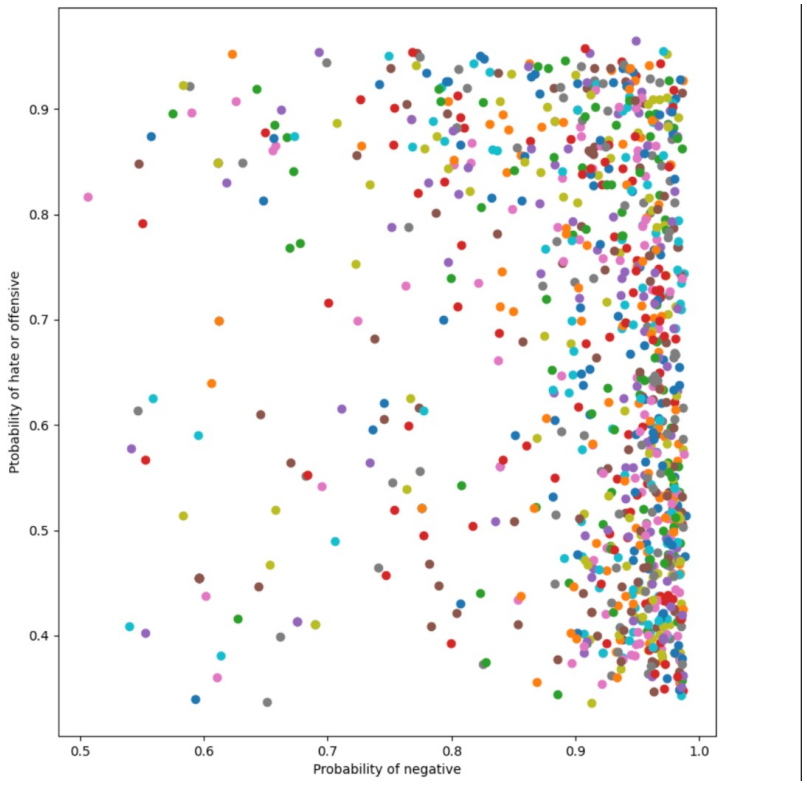
- Python
- HTML
- CSS
- JavaScript

PROJECT RUN-THROUGH

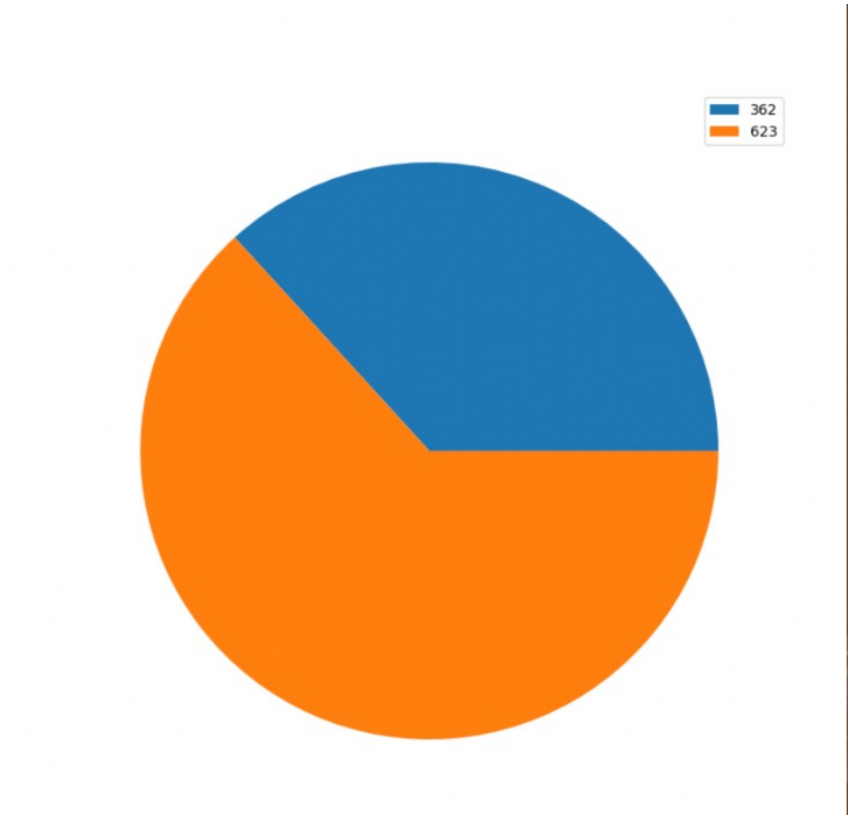
1. The image below depicts the home page.

2. On pressing the ‘Start’ button, the user was taken to the page shown below where they entered the keyword, region, and the timeframe that they wanted to search. The application then ran for the entered data. It provided us with over a 1000 tweets under 9 mins. The 1000 hits are only for negative tweets that our model had picked up.

3. The scatter plot shown below had given negative tweets related to the keyword kill in a 1000 km radius around Delhi (as per what the user had run). There were very few false positive tweets on hit, as was evident from the scatter plot diagram.



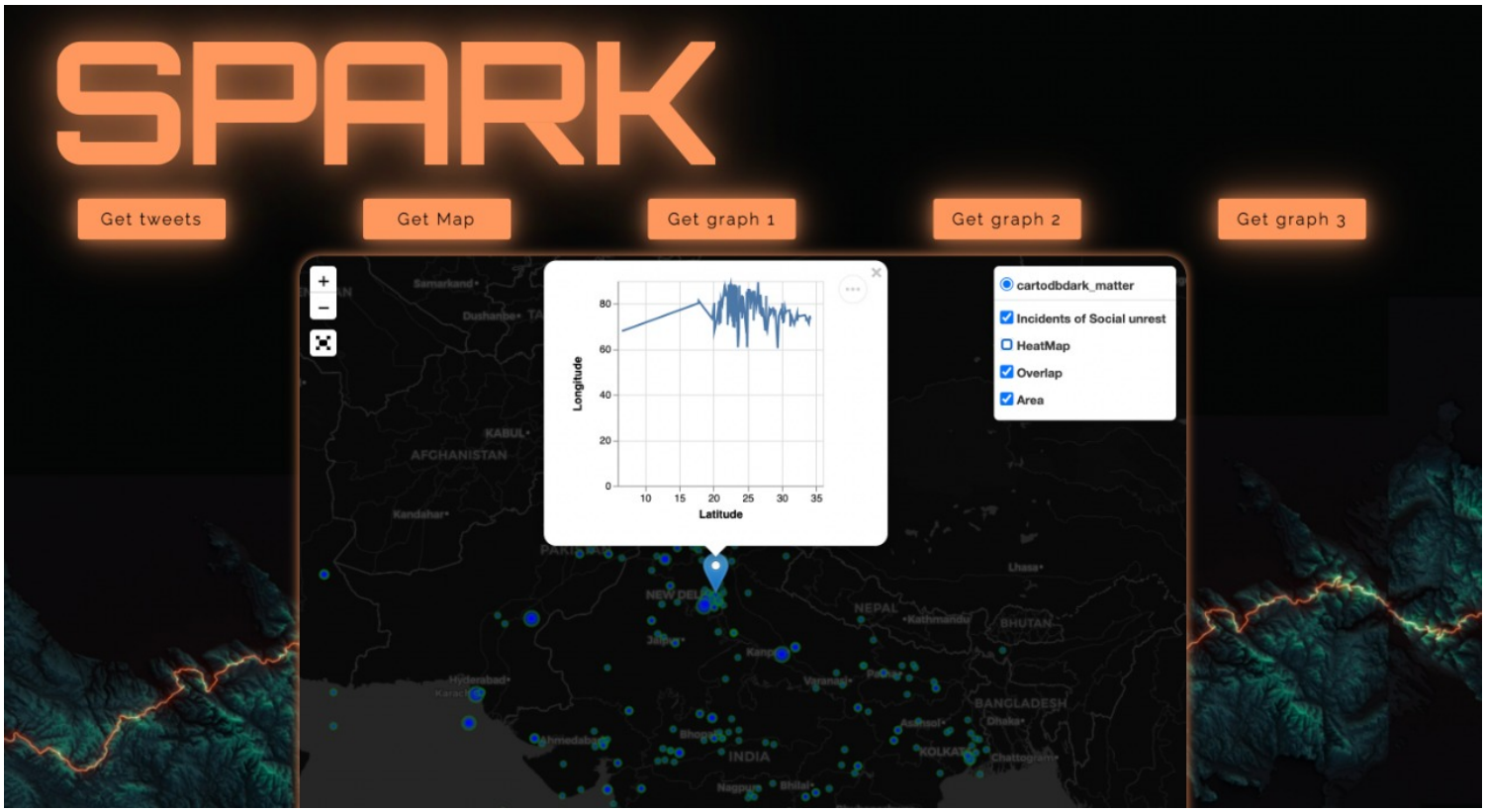
4. The pie chart shown below gave us negative tweets (in orange) and positive tweets (in blue) related to the keyword ‘kill’ in a 1000 km radius around Delhi.



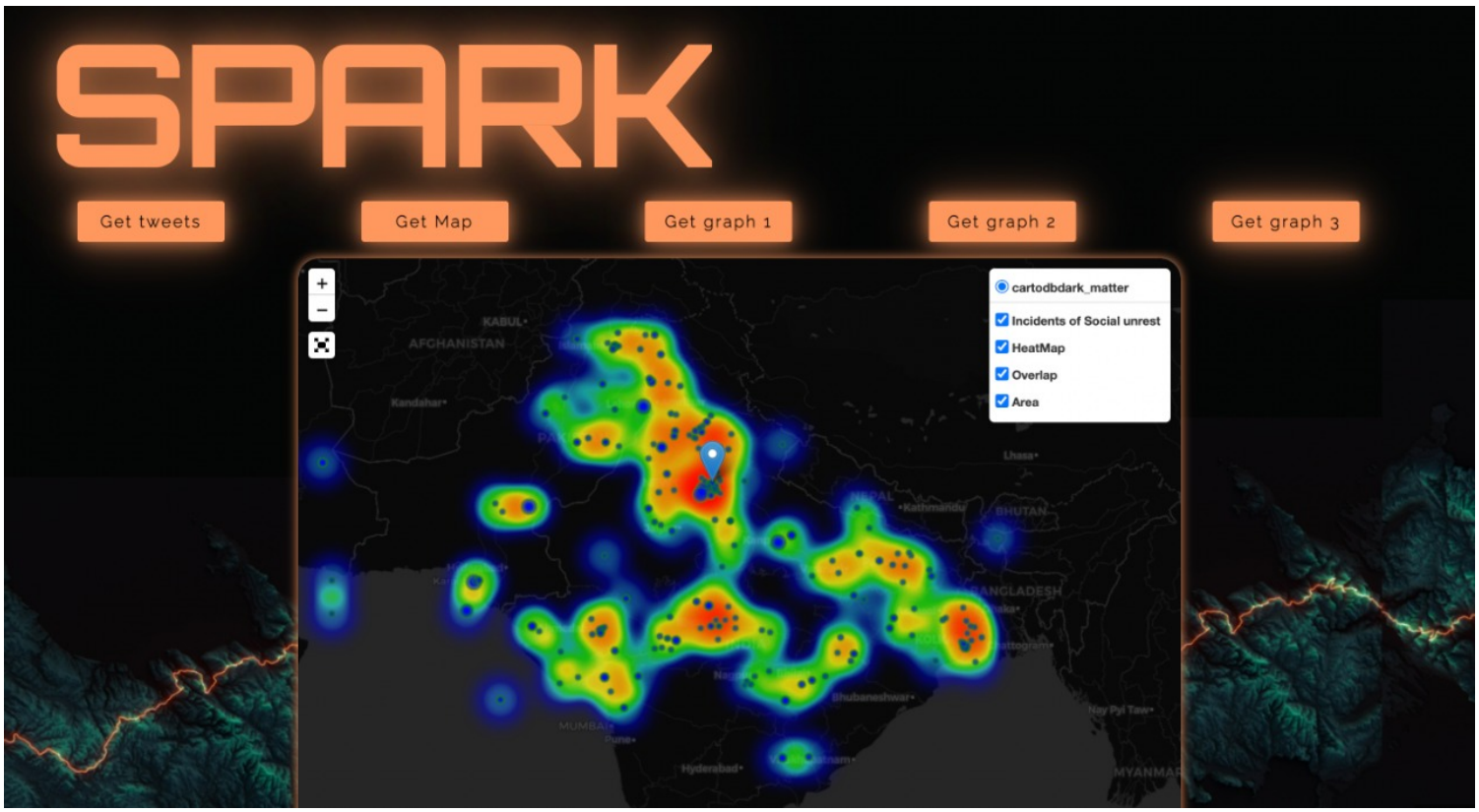
5. The image below depicted the tweets that had been picked up by our application extracted from the excel where the data had been recorded.

	you.\n\n#SaintRampalJi https://t.co/dEZWZdEdYx		
791	Thanksgiving Supreme God has forbidden to eat meat.\nYou kill God's beings for your own interests. When your account will be called for in the court of God, then no one can save you.\n\n#SaintRampalJi https://t.co/dEZWZdEdYx	28.269315	74.935942
792	Supreme God has forbidden to eat meat.\nYou kill God's beings for your own interests. When your account will be called for in the court of God, then no one can save you.\nThanksgiving\n#SaintRampalJi \nhttps://t.co/bGOdepdHYp https://t.co/sbL8YlSvHG	21.970758	83.331271
793	Meat eaters end up in hell. All living beings belong to God. Those who kill them for taste suffer badly in hell.\nThanksgiving\n\n#SaintRampalJi\nhttps://t.co/lDGrlOTv1y https://t.co/5uYkrG9ZTO	24.175067	73.520205
794	Meat eaters end up in hell. All living beings belong to God. Those who kill them for taste suffer badly in hell.\nThanksgiving\n\n#SaintRampalJi\nhttps://t.co/lDGrlOTv1y https://t.co/5uYkrG9ZTO	24.175067	73.520205
795	All living beings are the children of Supreme God.,,\n That Supreme God can never be pleased if one harms or kills another creature for fulfilment of one's own desire or taste. \n- Saint Rampal Ji Maharaj ,,,,,,\n\nThanksgiving\n\n#SaintRampalJi\n\nhttps://t.co/pjj6O9qBR5 https://t.co/ZOhqRCLnYN	23.070274	77.503158
796	Supreme God has forbidden to eat meat.\nYou kill God's beings for your own interests. When your account will be called for in the court of God, then no one can save you.Thanksgiving\n\n#SaintRampalJi https://t.co/4T1869fobP	25.277530	86.555588
	Supreme God has forbidden to eat meat.\nYou kill God's beings for your own interests		

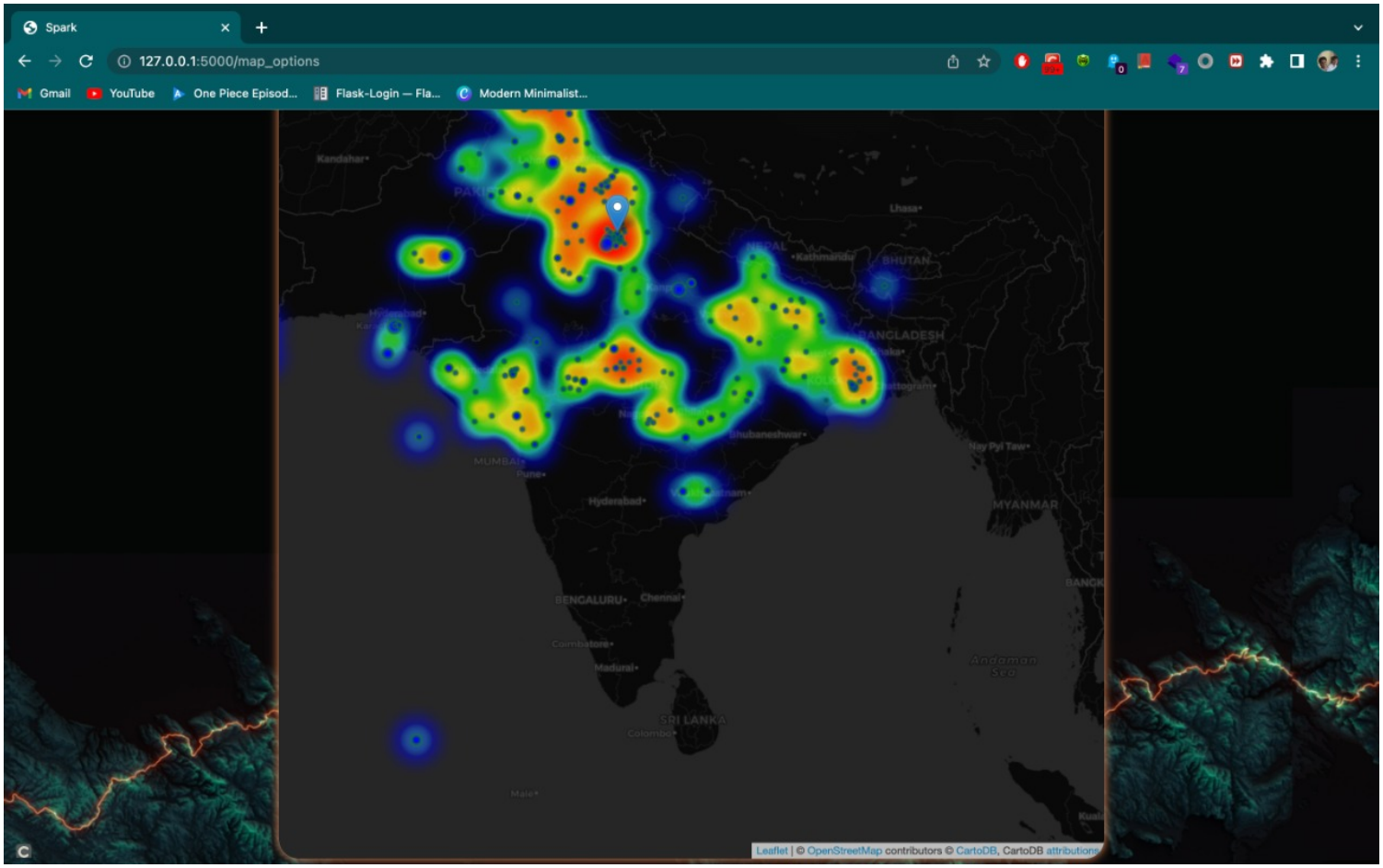
6. The snippet added below is the graph that showed us the negative tweets related to the keyword ‘kill’ in a 1000 km radius around Delhi.



7. Shown below is the visualization of the data on a ‘heat map’. Using the map interpretation, we got an idea about what regions were most likely to react negatively to a specific topic. This could help officials to allocate resources efficiently.



8. The map shown below provided us with areas with negative responses. This can also be used to see what areas need more awareness about certain topics. The map gave us an accurate representation of the data and was provided to the user in a way that was easy to understand.



CODING PART

To run this program, we run the file “heatmap_try.py” which generates a link, which we can open in our browser on the local server. This file calls the function “map_maker” from the file “location_scape.py” which has the main contents of the program. These are the code snippets from the file “location_scape.py.”

1) Importing all the necessary libraries and modules of python.

```
import math
import snscrape.modules.twitter as sntwitter
import pandas as pd
from time import sleep
from datetime import datetime
import os
import tweetnlp
import folium
from folium import plugins
from folium.plugins import HeatMap
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
plt.figure(figsize=(9,10))
import altair as alt
```

2) Defining the function “map_maker” which contains the main body of the program. Initializing a list “tweets_list2” for storing the extracted tweets. Also, defining 3 different models with respect to the three different categories namely “hate”, “offensive” and “multilingual sentiment analysis.”

```
def map_maker(keyword,from_date,to_date,region,radius):
    tweets_list2 = []
    tweets_list3 = []
    sentiment = []
    scatter = []
    model1 = tweetnlp.load('hate')
    model2 = tweetnlp.load('offensive')
    model3 = tweetnlp.load('sentiment_multilingual')
    good_ct, bad_ct = 0,0
```

3) Getting the latitude and longitude of the selected region with the help of OpenCage Geocoding and storing them in variables.

```
geocoder = OpenCageGeocode(key)
query = region
results = geocoder.geocode(query)
lat_for_map = results[0]['geometry']['lat']
lng_for_map = results[0]['geometry']['lng']
print (lat_for_map, lng_for_map)
loc = str(lat_for_map) +", "+ str(lng_for_map) + ", " + radius +"km"
print(loc)
print(from_date)
print(to_date)
lats_longs = []
```

4) Using TwitterSearchScraper to scrape data and incrementing the “count_hate” variable whenever the scrapper encounters a hateful or an offensive tweet. Also, appending the list “tweets_list2” and “lats_longs” with tweets and their locations respectively so that it can be loaded into a .csv file later.

```
count=0
count_hate = 0
for i,tweet in enumerate(sntwitter.TwitterSearchScraper('({}) geocode:"{}" since:{} until:{}'.format(keyword,loc,from_date,to_date)).
    if count > 1000:
        break
    try :

        hate_check = model1.hate(tweet.content)
        if hate_check['label'] != 'not-hate':
            count=count+1
            count_hate +=1
            tweets_list2.append([tweet.content, tweet.coordinates.latitude, tweet.coordinates.longitude])
            lats_longs.append([tweet.coordinates.latitude, tweet.coordinates.longitude])
            print(count)
            print(tweet)
        elif model2.offensive(tweet.content)['label'] == 'offensive':
            count=count+1
            count_hate +=1
            tweets_list2.append([tweet.content, tweet.coordinates.latitude, tweet.coordinates.longitude])
            lats_longs.append([tweet.coordinates.latitude, tweet.coordinates.longitude])
            print(count)
            print(tweet)
        temp = model3.sentiment(tweet.content)
```

```
        if temp['label'] == 'negative':
            tweets_list2.append([tweet.content, tweet.coordinates.latitude, tweet.coordinates.longitude])
            lats_longs.append([tweet.coordinates.latitude, tweet.coordinates.longitude])
            # sentiment.append(-temp['probability'])
            count+=1
            print(count)
            print(tweet)
            bad_ct +=1
        else:
            # sentiment.append(temp['probability'])
            good_ct +=1
            print("tweet not found negative")
        scatter.append([hate_check['probability'], temp['probability']])
    except AttributeError:
        print("discard tweet , had no attributes")
print("**************",count_hate)
```

5) Creating a dataframe from the tweets list “tweets_list2” above and then exporting the dataframe into csv file.

```
#tweets_df2 = pd.DataFrame(tweets_list2, columns=['Text', 'Latitide','Longitude'])
tweets_df3 = pd.DataFrame(tweets_list2, columns=['Text', 'Latitude','Longitude'])
#tweets_df2.to_csv("twitter data_hate_offensive.csv",index=False)
tweets_df3.to_csv("twitter data_bad.c
tweets_df4 = pd.DataFrame(sentiment, columns=['sentiment'])
tweets_df4.to_csv("twitter sentiment.csv")
```

columns: Axes | None

6) Using built in hash() function to convert the two dimensional location into one dimension by taking a dictionary where hash is assigned the root mean square value of the latitude and the longitude.

```
dt = {}
data = {}
for l in lats_longs:
    hash = math.sqrt(l[0]**2 + l[1]**2)
    data[hash] = l
    if hash in dt:
        dt[hash] +=1
    else:
        dt[hash] = 1
```

7) Setting the initial position of the folium map with the help of location co-ordiantes obtained using OpenCage Geocode.

```
unrest = folium.Map(location=[lat_for_map, lng_for_map], tiles='cartodbdark_matter', zoom_start = 5) #
```

8) Reading data from the previously generated csv file “twitter data_bad.csv” and storing it in a variable “source”. Using the alt function from the “Altair” library to plot the scatter plot using the longitudes and latitudes for a better understanding.

```
source = pd.read_csv("twitter data_bad.csv")
base = alt.Chart(source).mark_line().encode(
    alt.X("Latitude", axis=alt.Axis(title="Latitude")),
    alt.Y("Longitude", axis=alt.Axis(title="Longitude"))
)
vega = folium.features.VegaLite(base, width="%100",height="%100")
graph_popup = folium.Popup()
vega.add_to(graph_popup)

tooltip = "Click Me!!!"
folium.Marker(title = "Region:- " + str(region),location=[lat_for_map, lng_for_map], popup=graph_popup, tooltip=tooltip).add_to(unrest

plugins.Fullscreen(
    position='topleft',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True
).add_to(unrest)
```

9) Adding different features on the folium map to represent the data that we have extracted using heatmap, overlap, etc. Marking the location of tweets on the map using the co ordinates we have stored and overlapped them using the hash thus making the colour look more bold.

```
fg = folium.FeatureGroup(name='Incidents of Social unrest')
unrest.add_child(fg)

g1 = plugins.FeatureGroupSubGroup(fg, 'HeatMap')
unrest.add_child(g1)

g2 = plugins.FeatureGroupSubGroup(fg, 'Overlap')
unrest.add_child(g2)

g3 = plugins.FeatureGroupSubGroup(fg, 'Area')
unrest.add_child(g3)
print(lats_longs)
for i in range(len(lats_longs)):
    folium.CircleMarker(location=[lats_longs[i][0], lats_longs[i][1]],radius=3, fill=True,opacity=0.1,fill_opacity=0.25,fill_color="blue",color="green").add_to(g2)
for i in dt:
    print(data[i], dt[i])
    if dt[i]> 20:
        folium.CircleMarker(location=data[i],radius=2 + 5,fill=True,opacity=0.5,fill_opacity=0.6,fill_color="blue",color="green").add_to(g3)
    else:
        folium.CircleMarker(location=data[i],radius=2 + dt[i]/5,fill=True,opacity=0.5,fill_opacity=0.6,fill_color="blue",color="green").add_to(g3)

folium.LayerControl(collapsed=False).add_to(unrest)

HeatMap(lats_longs).add_to(g1)
unrest.save("templates/map1.html")
# unrest.save("templates/map2.html")
# unrest.save("templates/map3.html")

# map1.save("templates/map1.html")
# map2.save("templates/map2.html")
# map3.save("templates/map3.html")
print("!!!!!")
```

10) Plotting different scatter plots and storing them in images (png) and all the the graphs in html files.

```
y = np.array([good_ct, bad_ct])
plt.clf()
plt.pie(y)
plt.legend(y)
plt.savefig("/Users/shreysharma/Desktop/coding/Flask_practice/virtualenv_csd345/static/images/graph1.png")
plt.clf()
# plt.rcParams["figure.figsize"] = [7.00, 3.50]
# plt.rcParams["figure.autolayout"] = True

for tweet in scatter:
    x = tweet[0]
    y = tweet[1]
    plt.plot(x, y, marker = "o")
    plt.xlabel("Probability of negative")
    plt.ylabel("Ptobability of hate or offensive")

plt.savefig("/Users/shreysharma/Desktop/coding/Flask_practice/virtualenv_csd345/static/images/graph2.png")
plt.clf()
time = np.arange(len(lats_longs))
one = []
sec = []

for i in range(len(lats_longs)):
    one.append(lats_longs[i][0])
for i in range(len(lats_longs)):
    sec.append(lats_longs[i][1])
income = np.array(one)
expenses = np.array(sec)
fig, ax = plt.subplots(figsize=(9, 10))

Plot lines
ax.plot(time, income, color="green")
ax.plot(time, expenses, color="red")
```

```
# Fill area when income > expenses with green
ax.fill_between(
    time, income, expenses, where=(income > expenses),
    interpolate=True, color="green", alpha=0.25,
    label="Positive"
)

# Fill area when income <= expenses with red
ax.fill_between(
    time, income, expenses, where=(income <= expenses),
    interpolate=True, color="red", alpha=0.25,
    label="Negative"
)
plt.savefig("/Users/shreysharma/Desktop/coding/Flask_practice/virtualenv_csd345/static/images/graph3.png")
plt.clf()
```

22.

CONCLUSION

The purpose of the website SPARK, which employs a prediction model, is to assess and forecast local social unrest. The goal of this project is to assist law enforcement organizations in using real-time tweets from Twitter to locate potential locations of distress.

The application of this project could lead to a lot of improvement in the field of riot detection. Along with active detection, this data could also be used to analyze the mood of certain areas and can be used for much more than the use case that we had selected for this project. It can be used for recognising which areas need more medical assistance during a pandemic and other such exigencies.

SPARK can make a significant impact on someone's life. These range from **fighting crime with law enforcement agencies, preventing riots - and even saving lives**.

REFERENCES

1. https://opencagedata.com/api
2. https://www.analyticsvidhya.com/blog/2020/06/guide-geospatial-analysis-folium-python/
3. https://www.w3schools.com/html/default.asp
4. https://www.w3schools.com/w3css/default.asp
5. https://www.w3schools.com/python/default.asp
6. https://www.tutorialspoint.com/flask/index.htm
7. https://www.econstor.eu/bitstream/10419/125042/1/dp9522.pdf 'Religious Riots and Electoral Politics in India'. 'IZA Discussion Papers, No. 9522'