GRS CS 660
Graduate Introduction to Database Management Systems

Project Extra - Assignment 3
Mongo Tweets Final Report

Twee-a-tweet
Using Mongodb(NoSQL), PyMongo and Pycharm

By:

Aastha Anand
(aastha24@bu.edu)

TABLE OF CONTENTS

Slno.	Topic	Page no.
1	Part 1	3-4
2	Part 2: A	5
3	Part 2: B	6-7
4	Part 2: C	8-9
5	Part 2: D	10-12
6	Extra credit	13-14

Part 1

The **pymongo_tweepy.py** contains the code for mining the tweets with the keywords #deeplearning, #computervision, #datascience and #bigdata.
The **partA.py** contains all the code for each of the questions 1,2 and 3.

1) The “data” word is checked in the text of the tweet using regular expressions and incorporating the **IGNORECASE** (ensures case insensitive) of the **regex** (re) library in python. Pretty print is used to display it.
2) For query 2 the “data” word is checked again with the geo_enabled taken to be **not equal to false** and a count is used to count and display.
3) TextBlob and **sentiment polarity** is used to find the tweet: text sentiment to be positive, neutral or negative. (Checked it with text: “I don’t know” gives **polarity as 0**, “Good work” gives **polarity>0** and “horrible bad day” gives **polarity<0** and hence the sentiment is obtained with analysis and exploiting TextBlob

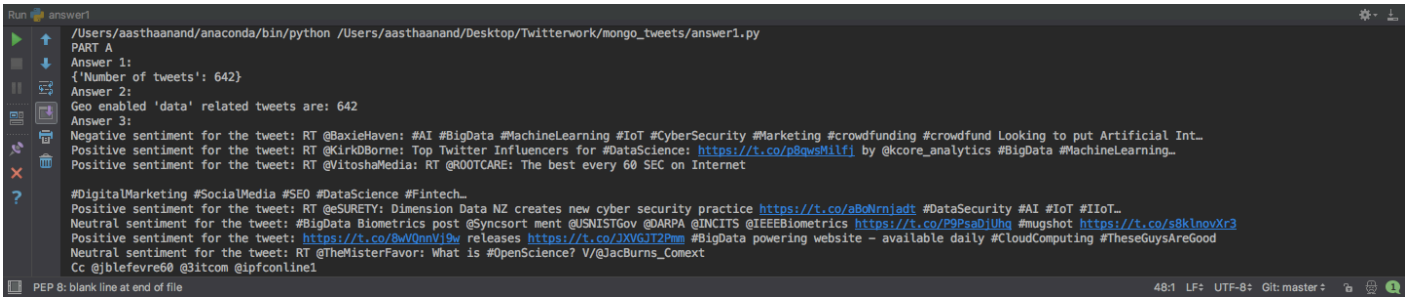
```
print("PART A")
# Number of tweets that have "data" in the tweets text: (CASE INSENSITIVE SEARCH)
regx = re.compile(".*data.*", re.IGNORECASE)
a1 = db.twitter_search.aggregate([{'$match':{'text':{'$regex': regx}}}, {'$count':"Number of tweets"}])
print("Answer 1: ")
for i in a1:
    pprint.pprint(i)


# From all the "data" related objects finding the ones which are "geo enabled"
print("Answer 2: ")
a2 = db.twitter_search.find({"$and":[{"user.geo_enabled":{"$ne": 'false'}},{'text':{'$regex': regx}}]}).count()
print("Geo enabled 'data' related tweets are: {}".format(a2))


# Detecting tweet sentiment
print("Answer 3: ")
a3 = db.twitter_search.find({'text':{'$regex':regx}}, {"text":1,"_id":0})
#for i in a3:
    #pprint.pprint(i)

for each_tweet in a3:
    tweet = each_tweet['text']
    text = TextBlob(tweet)
    if (text.sentiment.polarity > 0):
        print("Positive sentiment for the tweet: " + tweet)
    elif (text.sentiment.polarity == 0):
        print("Neutral sentiment for the tweet: " + tweet)
    else:
        print("Negative sentiment for the tweet: " + tweet)
```

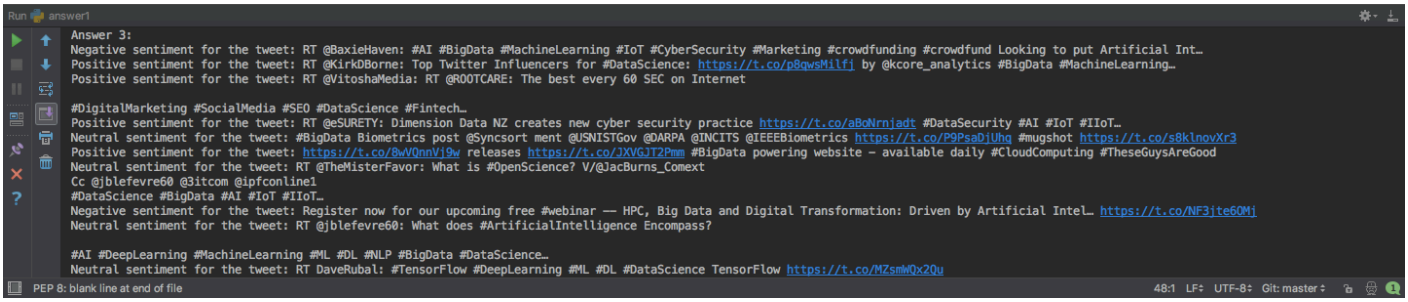
In the first question the "**Number of tweets**" are the number of tweets that have data somewhere in the tweet's text.

Geo enabled "data" related tweets are the data related objects which are geo_enabled

TextBlob python library is used to detect sentiment of tweet "text". The sentiment of each of the 1000 tweets are given in answer 3.

A screenshot of answers as obtained from my database tweets are:





Part 2

A) A new script is created that mines tweets from Twitter using Tweepy API and it is based on location. The **stream.filter** is changed as follows:

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
# Set up the listener. The 'wait_on_rate_limit=True' is needed to help with Twitter API rate limiting.
listener = StreamListener(api=tweepy.API(wait_on_rate_limit=True))
streamer = tweepy.Stream(auth=auth, listener=listener)
#USA bounding box
streamer.filter(locations=[-175.1, 22.4, -59.8, 72.3])
```

B) Did some googling to find extraction of emojis and created a function that uses from **import UNICODE_EMOJI.** For this part of the questions the integral part is the pymongo query, emoji finder function and the rest is simple dictionary python coding.

```
# PART 2B:
print("PART 2B: ")
print(" ")
query = db.usa_tweets_collection.find({"$or":[{"place.place_type":"city"},{"place.place_type":"neighborhood"}]},{"text":1,"place.full_name":1})
#for i in data:
#    pprint.pprint(i)

# Define dictionaries according to the question
p1 = {}
p2 = {}
p3 = {}
count = 0

# Obtained from StackOverflow the emoji - finder
def emoji_finder(str):
    emoji_list = []
    for check in str:
        if check in UNICODE_EMOJI:
            emoji_list.append(check)
    #print(emoji_list)
    return (emoji_list)
```

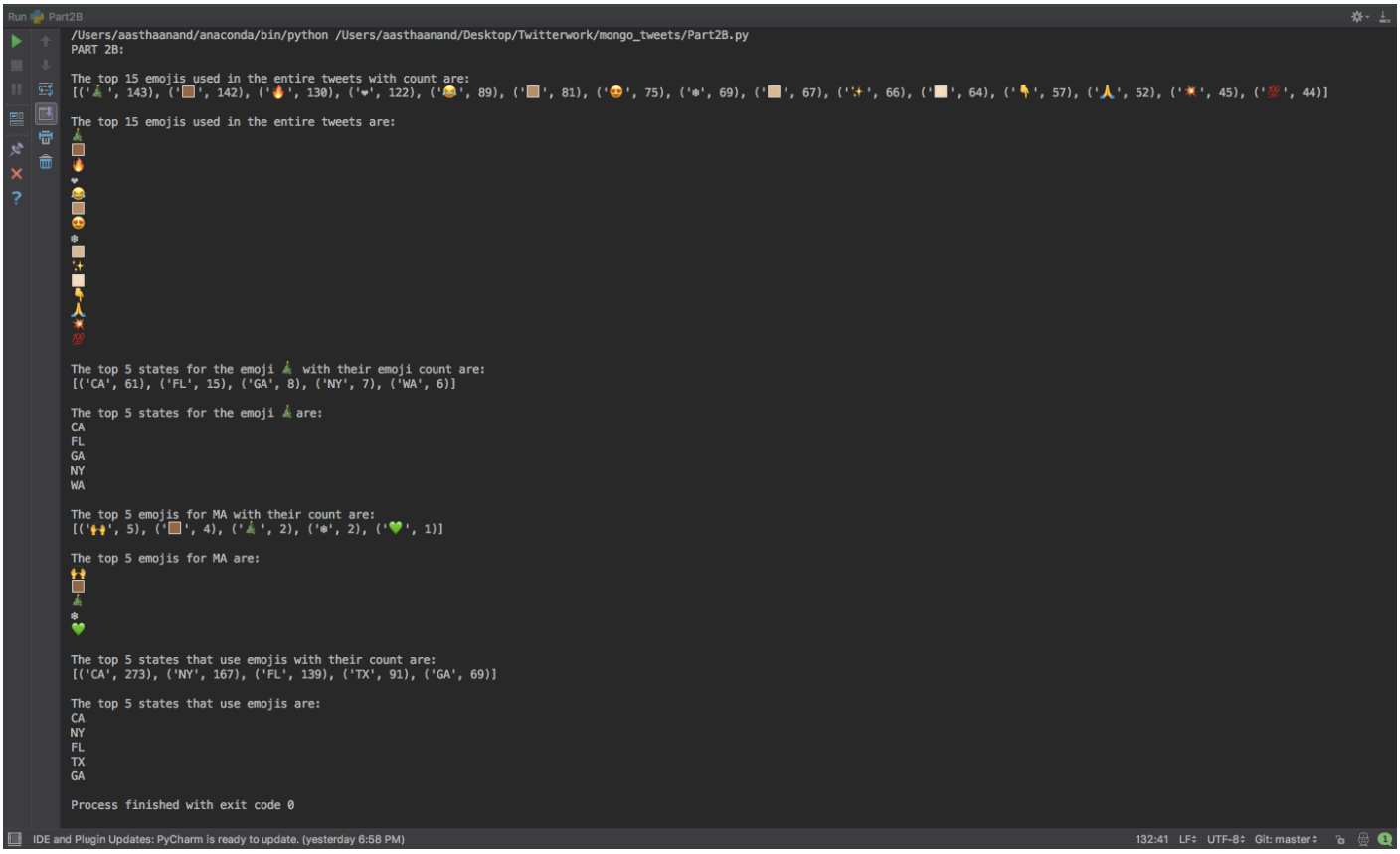
The major part of the programming includes the extraction of states which are mentioned in the last two characters which need to be extracted, rest is simple to understand and uses dictionary and its properties in python for the flow to the final output.

```
for each_tweet in query:
    # Obtaining the text from the tweet
    text = each_tweet['text']
    emo = emoji_finder(text)

    # Considering the emojis are present
    if emo:
        state = each_tweet['place']['full_name'][-2:]
        # Creating the dic with keys as emojis and values as their count
        for element in emo:
            if element not in p1.keys():
                p1[element] = 1
            else:
                p1[element] = p1[element] + 1

            # including the state information
            if element in p2.keys():
                if state not in p2.get(element, {}):
                    p2[element][state] = 1
                else:
                    p2[element][state] = p2[element][state] + 1
            else:
                p2[element] = {}
                p2[element][state] = 1

            # state with total emoji counts
            if state in p3.keys():
                if element in p3.get(state, {}):
                    p3[state][element] = p3[state][element] + 1
                else:
                    p3[state][element] = 1
            else:
                p3[state] = {}
                p3[state][element] = 1
```

Each of the dictionary outputs with the specific outputs of the questions asked are given below:

I considered only the text field for extracting all the emojis (not extended tweet since mentioned on piazza).

The answers are below:-



Part 2C

- The part2C.py contains the code for the questions for this part of the project.
- The first question requires the top 5 states to for which we require to extract the place_type and text as given in the query below:

```
# PART 2: C1
print("Part C1:")
query = db.usa_tweets_collection.find({"$or":[{"place.place_type":"city"},{"place.place_type":"neighborhood"}]},{"place.full_name":1,"_id":0})
#for i in query:
    #pprint.pprint(i)

states = []
for each_tweet in query:
    state = each_tweet['place']['full_name'][-2:]
    states.append(state)
#print(states)

counter_states = dict(Counter(states))
#print(counter_states)

sorted_cs = sorted(counter_states.items(), key=operator.itemgetter(1), reverse=True)
print(sorted_cs)
print(" ")

print("The top 5 states which have tweets are:")
no=0
for tup in sorted_cs:
    if no < 5:
        print(tup[0])
        no = no+1
print("")
print("")
```

- The rest part of the code involves sorting to **descending order** in order to get the highest count.
- The second question is similar to the first but here we take only the **place_type as "city"** and only **consider CA** (California abbreviation).

```
# PART 2: C2
print("Part C2:")
query2 = db.usa_tweets_collection.find( {"place.place_type":"city"}, {"place.name":1, "place.full_name":1, "_id":0})
#for i in query2:
    #pprint.pprint(i)

cities = []
for each_tweet in query2:
    state = each_tweet['place']['full_name'][-2:]

    if state == "CA":
        city = each_tweet['place']['name']
        cities.append(city)
#print(cities)
counter_cities = dict(Counter(cities))
#print(counter_states)

sorted_c = sorted(counter_cities.items(), key=operator.itemgetter(1), reverse=True)
print(sorted_c)
print("")

print("The top 5 cities in California which have tweets are:")
no_c = 0
for tup in sorted_c:
    if no_c < 5:
        print(tup[0])
        no_c = no_c+1
```

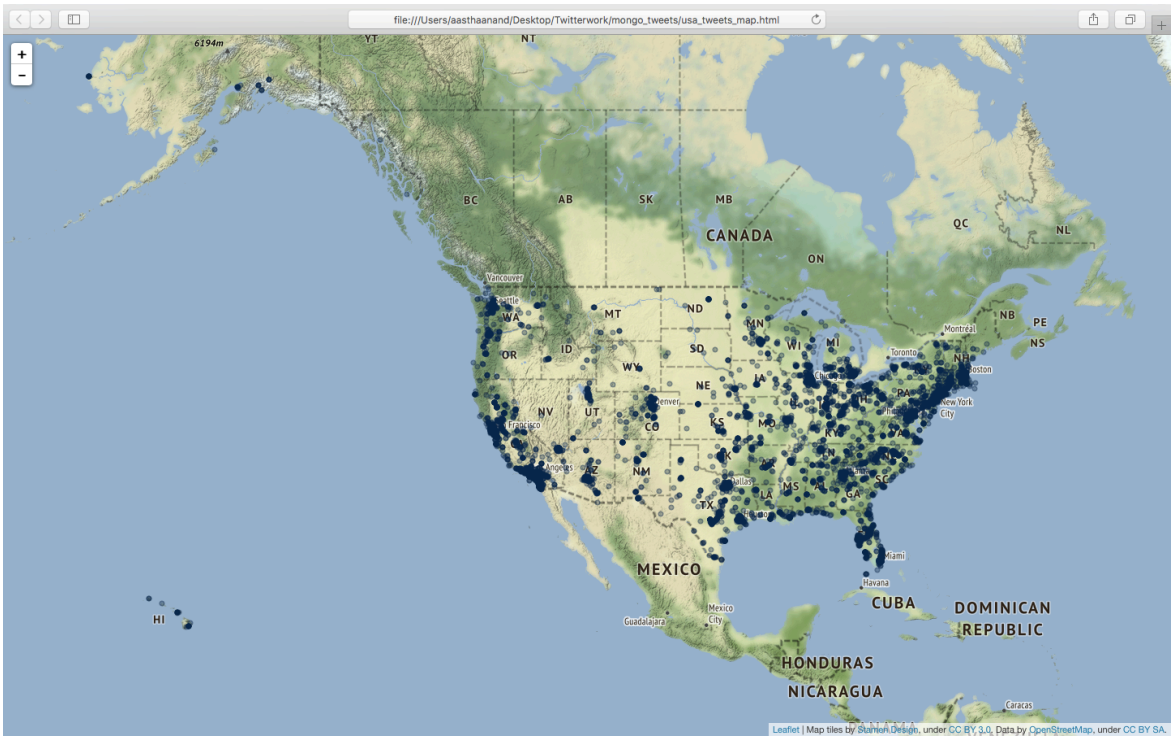
Each of the answers are given below:

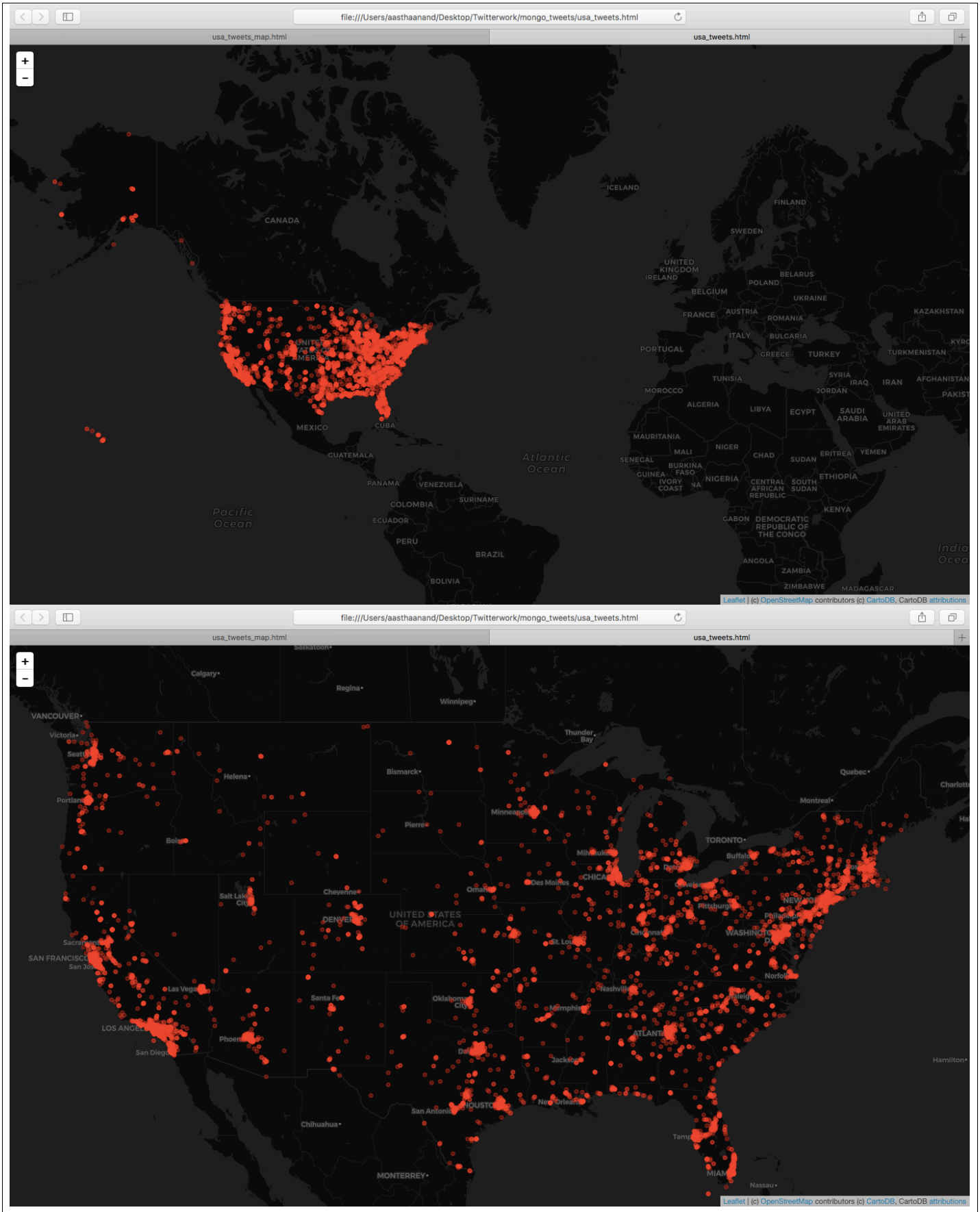
```
Run Part2C
/Users/aasthaanand/anaconda/bin/python /Users/aasthaanand/Desktop/Twitterwork/mongo_tweets/Part2C.py
Part C1:
[('CA', 2266), ('NY', 915), ('TX', 691), ('FL', 574), ('IL', 291), ('PA', 277), ('GA', 248), ('WA', 241), ('OH', 232), ('NV', 221), ('OR', 213), ('MA', 204), ('NJ', 190), ('VA', 184), ('MI', 172),

The top 5 states which have tweets are:
CA
NY
TX
FL
IL


Part C2:
[('Los Angeles', 655), ('San Francisco', 184), ('San Diego', 117), ('Anaheim', 70), ('San Jose', 50), ('West Hollywood', 46), ('Sacramento', 39), ('Oakland', 37), ('Long Beach', 33), ('Grass Valle

The top 5 cities in California which have tweets are:
Los Angeles
San Francisco
San Diego
Anaheim
San Jose

Process finished with exit code 0

IDE and Plugin Updates: PyCharm is ready to update. (yesterday 6:58 PM)
47:15 LF: UTF-8: Git: master
```

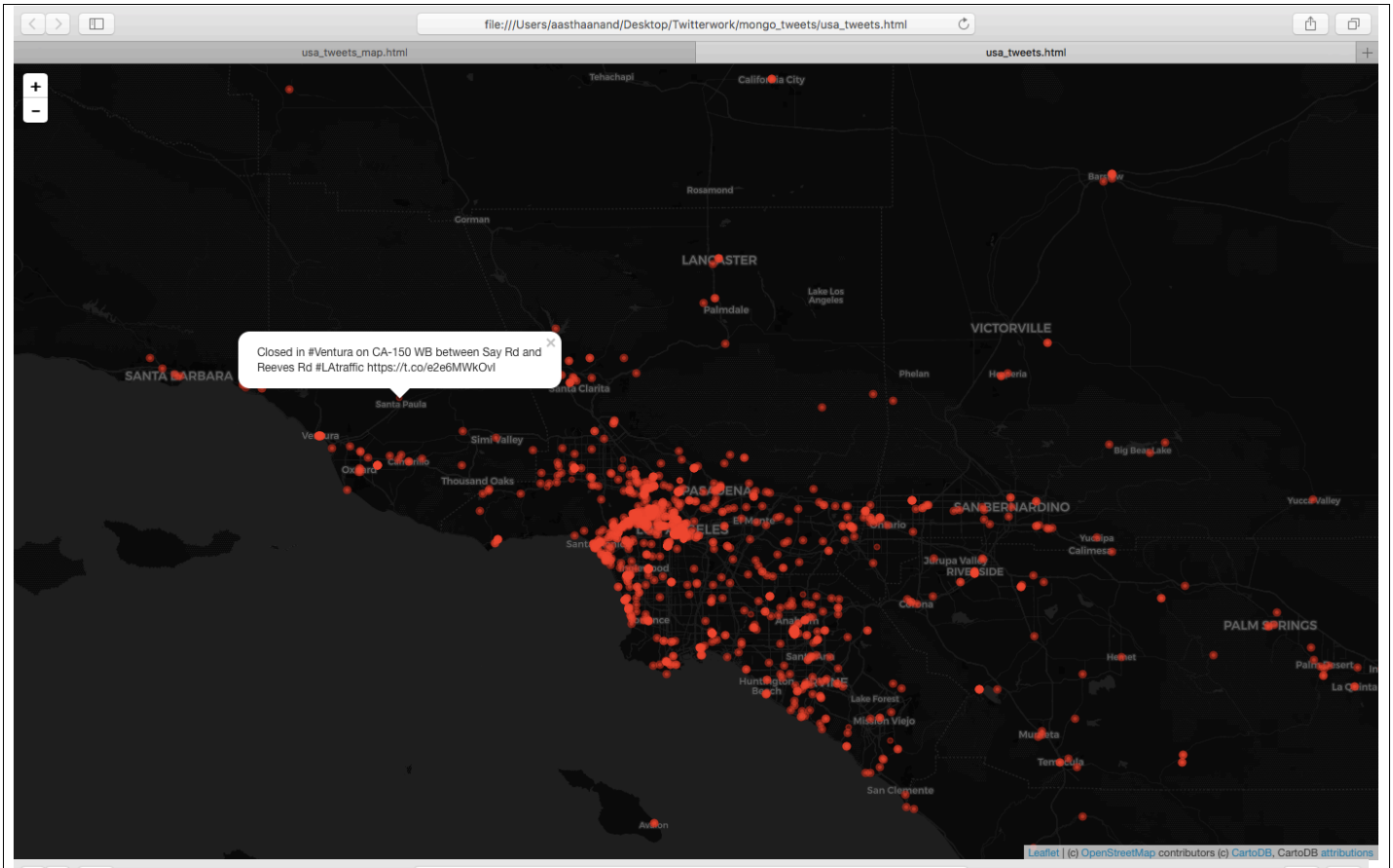
Part 2D

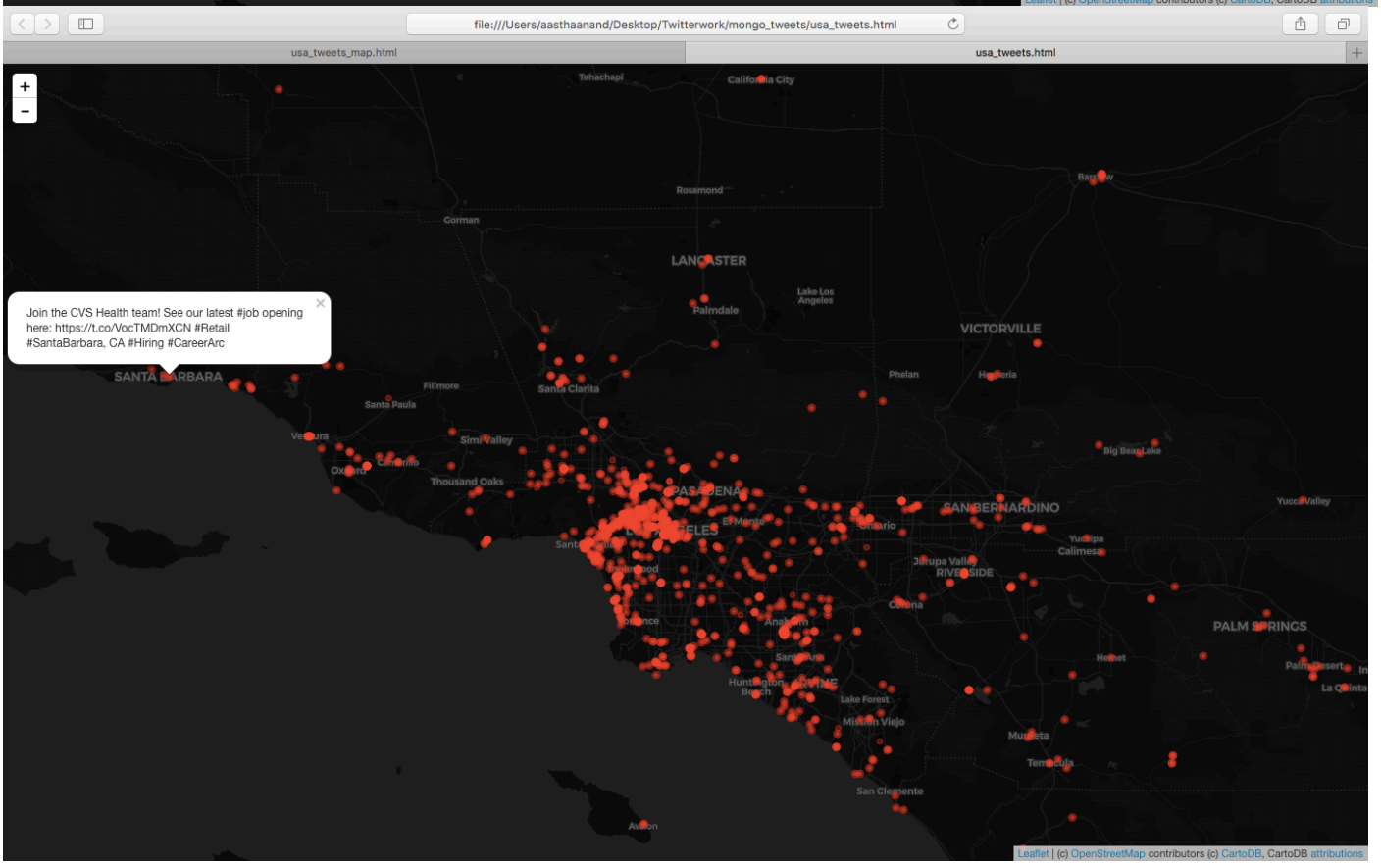
- The partD.py contains all the code for this part. This was by far my **favorite** part of the project and I tried different ways of using **folium** to get my map. Trust me I did not do it only for the brownie points. ;)
- I was also able to work up the pop up and when we zoom in and click a point in the map it pops up with the tweet (text) sent from that location.

```
import pandas
import folium
import ast

file = pandas.read_csv('usa_tweets.csv')
#print(file)
map = folium.Map(location=[48, -102], zoom_start=2, tiles='CartoDB dark_matter')
# slicing it to get the text part of the tweet
texts = file['text'].tolist()

#print(tweets)
geo_location = (file['geo'])
#print(geo_location)
counter=0
for line in geo_location:
    # making a dict to store the key coordinates and getting the longitude and lattitude
    dic_geo = ast.literal_eval(line)
    coord = dic_geo['coordinates']
    #print(coord)
    folium.CircleMarker(coord, radius=500, color='red', fill_color='red', popup=texts[counter]).add_to(map)
    counter = counter + 1

map.save('map.html')
```




file:///Users/aasthaanand/Desktop/Twitterwork/mongo_tweets/usa_tweets.html
usa_tweets_map.html
usa_tweets.html
CANADA
MEXICO
CUBA
GUATEMALA
PANAMA
VENEZUELA
COLOMBIA
SURINAME
ECUADOR
PERU
BRAZIL
BOLIVIA
Pacific Ocean
Atlantic Ocean
ICELAND
FINLAND
SWEDEN
UNITED KINGDOM
IRELAND
BELARUS
POLAND
BELGIUM
UKRAINE
FRANCE
AUSTRIA
ROMANIA
ITALY
BULGARIA
PORTUGAL
GREECE
TURKEY
TURKMENISTAN
TUNISIA
SYRIA
IRAQ
IRAN
AFGHANISTAN
MOROCCO
JORDAN
ALGERIA
LIBYA
EGYPT
SAUDI ARABIA
UNITED ARAB EMIRATES
MAURITANIA
MALI
NIGER
CHAD
SUDAN
ERITREA
YEMEN
SENEGAL
BURKINA FASO
GUINEA
IVORY COAST
NIGERIA
CENTRAL AFRICAN REPUBLIC
SOUTH SUDAN
ETHIOPIA
KENYA
GABON
DEMOCRATIC REPUBLIC OF THE CONGO
ANGOLA
ZAMBIA
ZIMBABWE
MADAGASCAR
Leaflet | (c) OpenStreetMap contributors (c) CartoDB, CartoDB attributions
Calgary
Regina
Winnipeg
VANCOUVER
Victoria
Thunder Bay
Quebec
Montreal
TORONTO
Helena
Bismarck
Pierre
Minneapolis
Boise
Salt Lake City
Cheyenne
Omaha
Des Moines
CHICAGO
Buffalo
Pittsburgh
NEW YORK
WASHINGTON
DENVER
UNITED STATES OF AMERICA
St. Louis
SAN FRANCISCO
Norfolk
Nashville
Las Vegas
Santa Fe
Oklahoma City
Memphis
LOS ANGELES
Phoenix
San Diego
ATLANTA
Jackson
Hamilton
San Antonio
HOUSTON
New Orleans
Chihuahua
MONTERREY
MIAMI
Nassau


file:///Users/aasthaanand/Desktop/Twitterwork/mongo_tweets/usa_tweets.html
usa_tweets_map.html
usa_tweets.html
Closed in #Ventura on CA-150 WB between Say Rd and Reeves Rd #LAtraffic https://t.co/e2e6MWkOvl
Leaflet | (c) OpenStreetMap contributors (c) CartoDB, CartoDB attributions

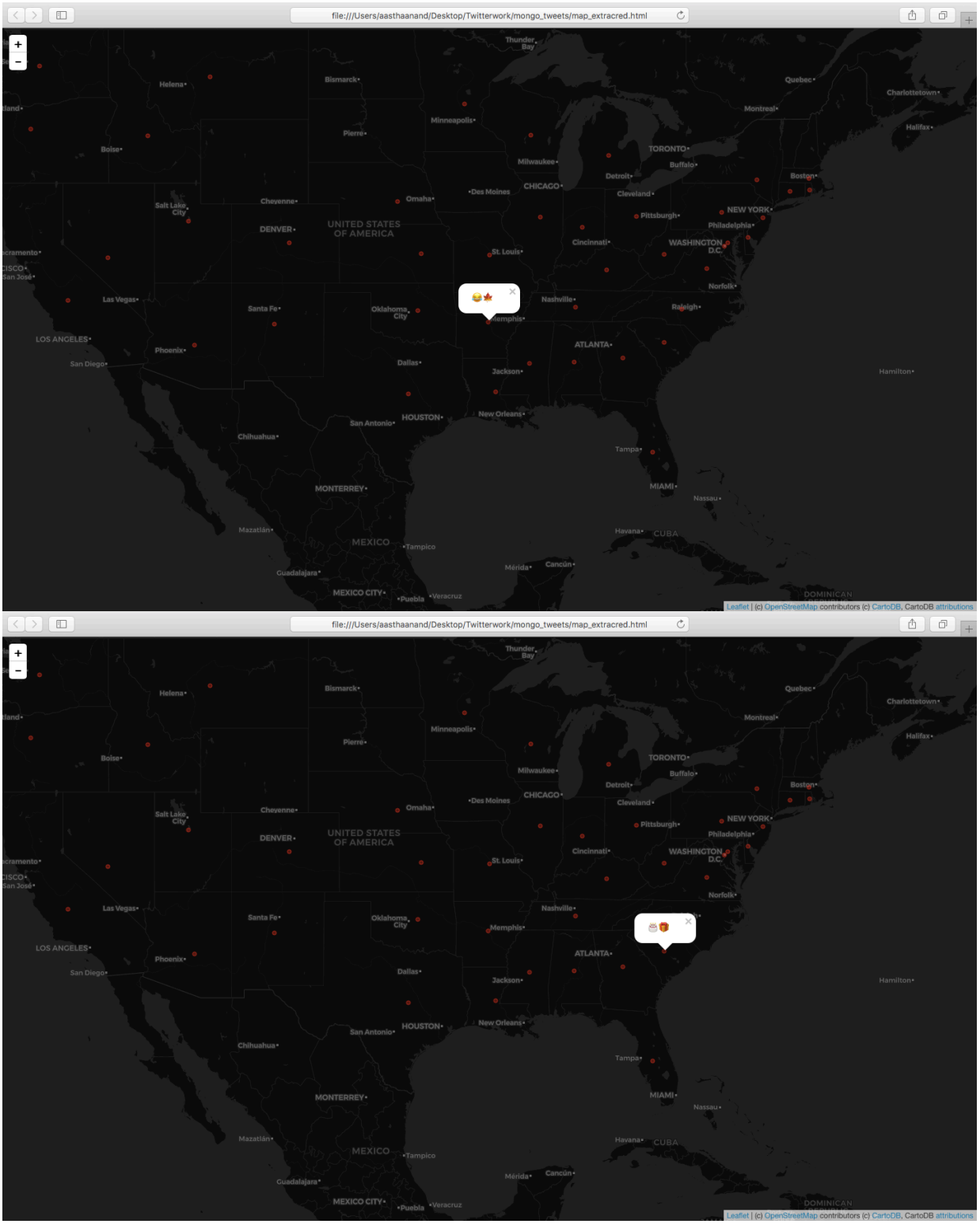

file:///Users/aasthaanand/Desktop/Twitterwork/mongo_tweets/usa_tweets.html
usa_tweets_map.html
usa_tweets.html
Join the CVS Health team! See our latest #job opening here: https://t.co/VocTMDmXCN #Retail #SantaBarbara, CA #Hiring #CareerArc
Leaflet | (c) OpenStreetMap contributors (c) CartoDB, CartoDB attributions

Extra credit

This involved a little elaboration with the part 2B third question. With the same emoji finder and involved some coding and manipulation of dictionaries and tuples as follows:

```
for k,v in p3.items():
    sorted_p = sorted(p3[k].items(), key=operator.itemgetter(1))
    top2=sorted_p[-2:]
    final[k] = {}
    final[k] = top2[::-1]
# my top 2 emojis in each state
print(final)

state_coord = {
        'AK': [61.37,-152.404],
        'AL': [32.806,-86.791130],
        'AR': [34.969704,-92.373123],
        'AZ': [33.729759,-111.431221],
        'CA': [36.116203,-119.681564],
        'CO': [39.059811,-105.311104],
        'CT': [41.597782,-72.755371],
        'DC': [38.897438,-77.026817],
        'DE': [39.318523,-75.507141],
        'FL': [27.766279,-81.686783],
        'GA': [33.040619,-83.643074],
        'HI': [21.094318,-157.498337],
        'IA': [42.011539,-93.210526],
        'ID': [44.240459,-114.478828],
        'IL': [40.349457,-88.986137],
        'IN': [39.849426,-86.258278],
        'KS': [38.526600,-96.726486],
        'KY': [37.668140,-84.670067],
        'LA': [31.169546,-91.867805],
        'MA': [42.230171,-71.530106],
        'MD': [39.063946,-76.802101],
        'ME': [44.693947,-69.381927],
        'MI': [43.326618,-84.536095],
        'MN': [45.694454,-93.900192],
        'MO': [38.456085,-92.288368],
        'MS': [32.741646,-89.678696],
        'MT': [46.921925,-110.454353],
        'NC': [35.630066,-79.806419],
        'ND': [47.528912,-99.784012],
        'NE': [41.125370,-98.268082],
        'NH': [43.452492,-71.563896],
        'NJ': [40.298904,-74.521011],
        'NM': [34.840515,-106.248482],
        'NV': [38.313515,-117.055374],
        'NY': [42.165726,-74.948051],
        'OH': [40.388783,-82.764915],
        'OK': [35.565342,-96.928917],
        'OR': [44.572021,-122.070938],
        'PA': [40.590752,-77.209755],
        'RI': [41.680893,-71.511780],
        'SC': [33.856892,-80.945007],
        'SD': [44.299782,-99.438828],
        'TN': [35.747845,-86.692345],
        'NH': [43.452492,-71.563896],
        'NJ': [40.298904,-74.521011],
        'NM': [34.840515,-106.248482],
        'NV': [38.313515,-117.055374],
        'NY': [42.165726,-74.948051],
        'OH': [40.388783,-82.764915],
        'OK': [35.565342,-96.928917],
        'OR': [44.572021,-122.070938],
        'PA': [40.590752,-77.209755],
        'RI': [41.680893,-71.511780],
        'SC': [33.856892,-80.945007],
        'SD': [44.299782,-99.438828],
        'TN': [35.747845,-86.692345],
        'TX': [31.054487,-97.563461],
        'UT': [40.150032,-111.862434],
        'VA': [37.769337,-78.169968],
        'VT': [44.045876,-72.710686],
        'WA': [47.400902,-121.490494],
        'WI': [44.268543,-89.616508],
        'WV': [38.491226,-80.954453],
        'WY': [42.755966,-107.302490]
}

map = folium.Map(location=[48, -102], zoom_start=2, tiles='CartoDB dark_matter')
del final['nx']
del final['es']

def cvsr(arg):
    valueString = [elem[0] for elem in arg]
    str = ''.join(valueString)
    return str

for key, value in final.items():
    coord = state_coord[key]
    #print(coord)
    text = cvsr(value)
    #print(type(text))
    folium.CircleMarker(coord, radius=500, color='red', fill_color='red',popup=text).add_to(map)

map.save('map_extracred.html')
```

The map obtained is below with each state in my tweets having **top 2 emojis** and when the point is clicked the top 2 emojis are popped up as follows: **file is map_extracred.html**



file:///Users/aasthaanand/Desktop/Twitterwork/mongo_tweets/map_extracred.html