DISASTER RELIEF

LOCATION

a GA capstone by naren rajasekaran

What is DISASTER RELIEF LOCATION about ?

disaster

dɪˈzɑːstə/
a sudden event resulting in great damage and loss of life

With the recent events around the world with regards to natural disasters (Hurricane Harvey,Irma) as well as terror attacks in Paris/London, peope are increasingly turning to social media to report events and to request for help.

My project was to reduce the noise from one of these platforms (Twitter), provide tracking (approximate) and to assign locations for these relief workers.

Dataset

I used the Twitter API services to scrape revelant topics which were widely spoken of (Trending) and did my best to bring some form of localisation for it. This is to simulate the closeness it will have to disasters where it is usually based in a single location (region, state) and not the entire country. As such, i chose to collect tweets by reducing the geo-range to within the boundaries of the USA by providing the neccesary bounding box coordinates.

Problems with my dataset

DATE October 3, 2017

SUBJECT Re: Can't find coordinates in tweet

EMAIL

customersupport@twitter.com

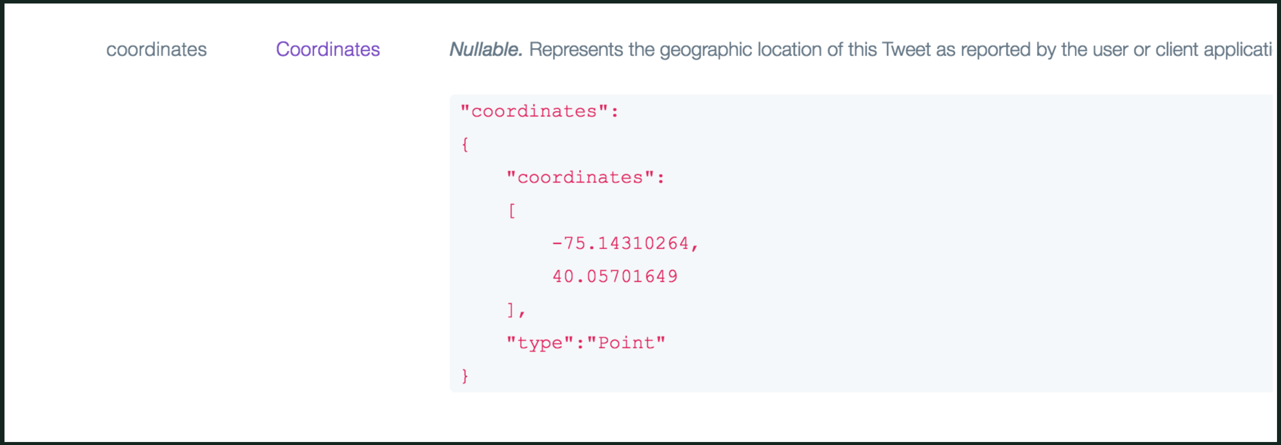
Hi Customer Support Guy,
im using tweets for a datascience project but most tweets don't have geolocations. what gives?

DATE October 3, 2017

SUBJECT Re: Can't find coordinates in tweet

Hi Naren,
sucks to be you. lol

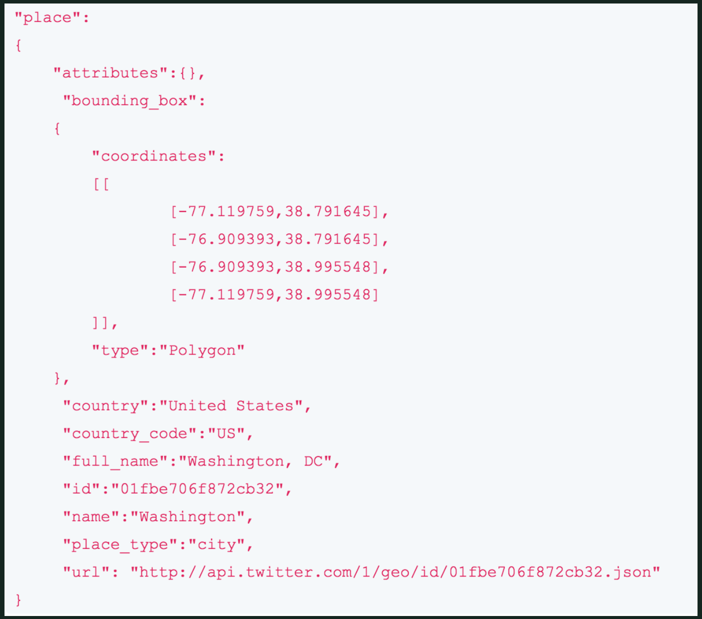
A DRAMATIC RETELLING OF EVENTS



coordinates Coordinates *Nullable.* Represents the geographic location of this Tweet as reported by the user or client applicati

```
"coordinates":
{
    "coordinates":
    [
        -75.14310264,
        40.05701649
    ],
    "type":"Point"
}
```

Coordinates are opt-in per tweet. Every tweet has to be configured to send current location while tweeting.

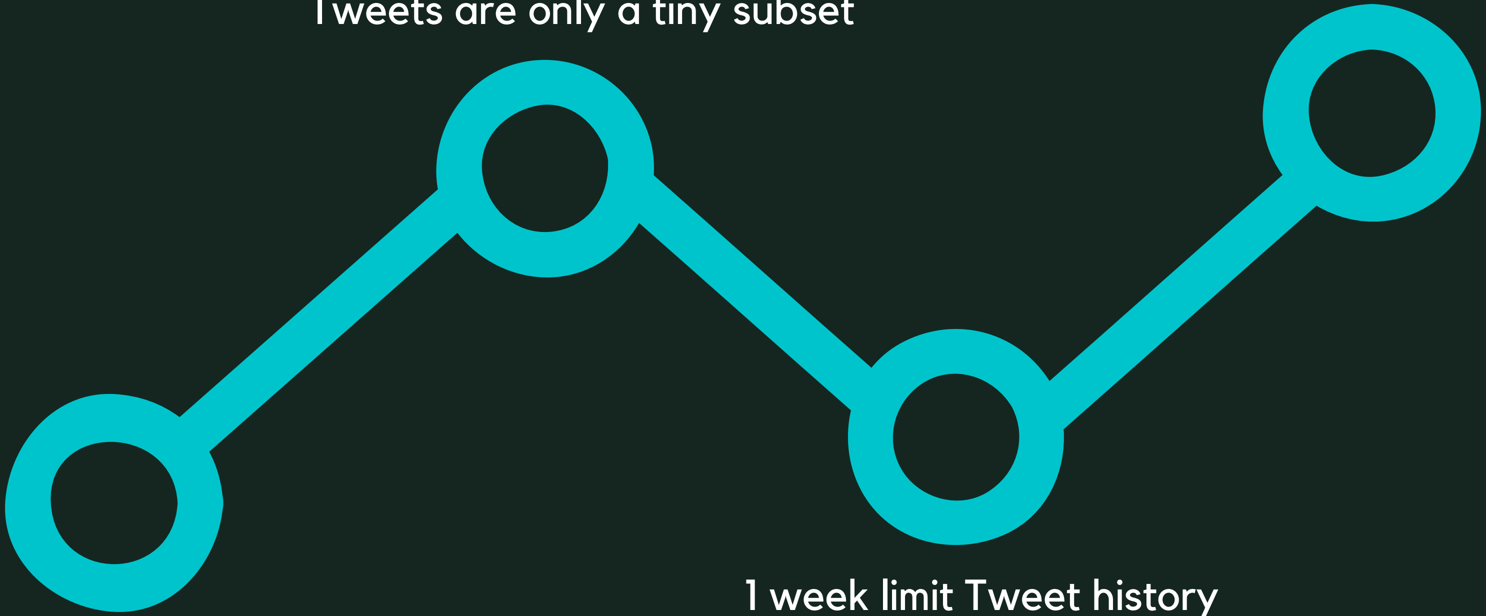


```
"place":
{
    "attributes":{},
     "bounding_box":
    {
        "coordinates":
        [[
                [-77.119759,38.791645],
                [-76.909393,38.791645],
                [-76.909393,38.995548],
                [-77.119759,38.995548]
        ]],
        "type":"Polygon"
    },
     "country":"United States",
     "country_code":"US",
     "full_name":"Washington, DC",
     "id":"01fbe706f872cb32",
     "name":"Washington",
     "place_type":"city",
     "url": "http://api.twitter.com/1/geo/id/01fbe706f872cb32.json"
}
```

An approximate location is tied to the tweets by default. We are given bounding boxes and the state of the origin of the tweet.

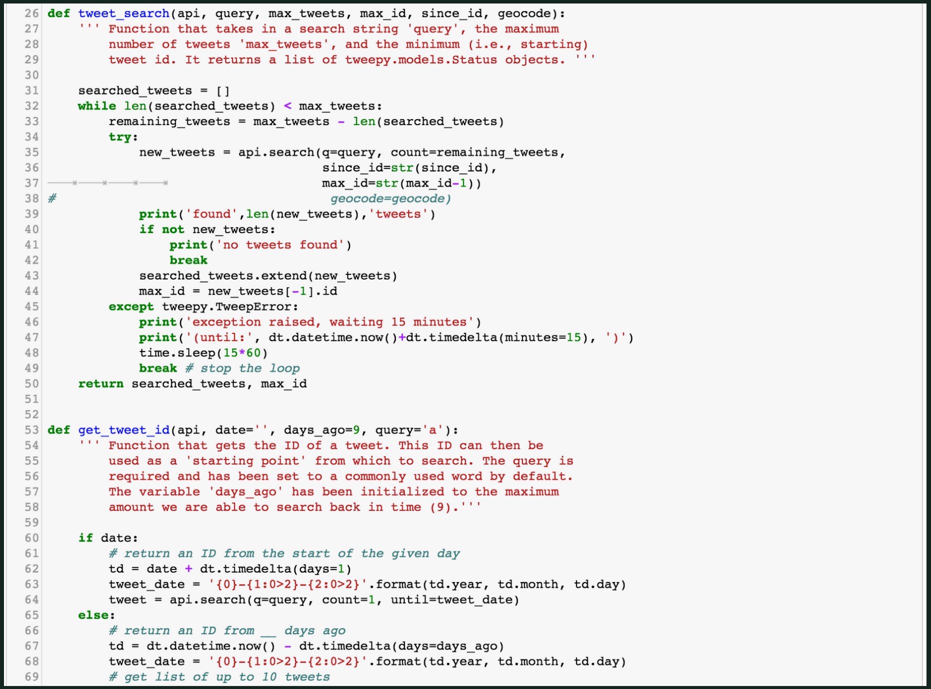
This ended up being a blessing in disguise as in most rescue efforts especially involving people who have cellphone connectivity, they tend to turn off non-essential services including locations as they tend to drain the most battery.

Other issues encountered


bounding boxes out of range of country limits.
Tweets are only a tiny subset
1 week limit Tweet history
Tweets are rate limited

chronic laziness

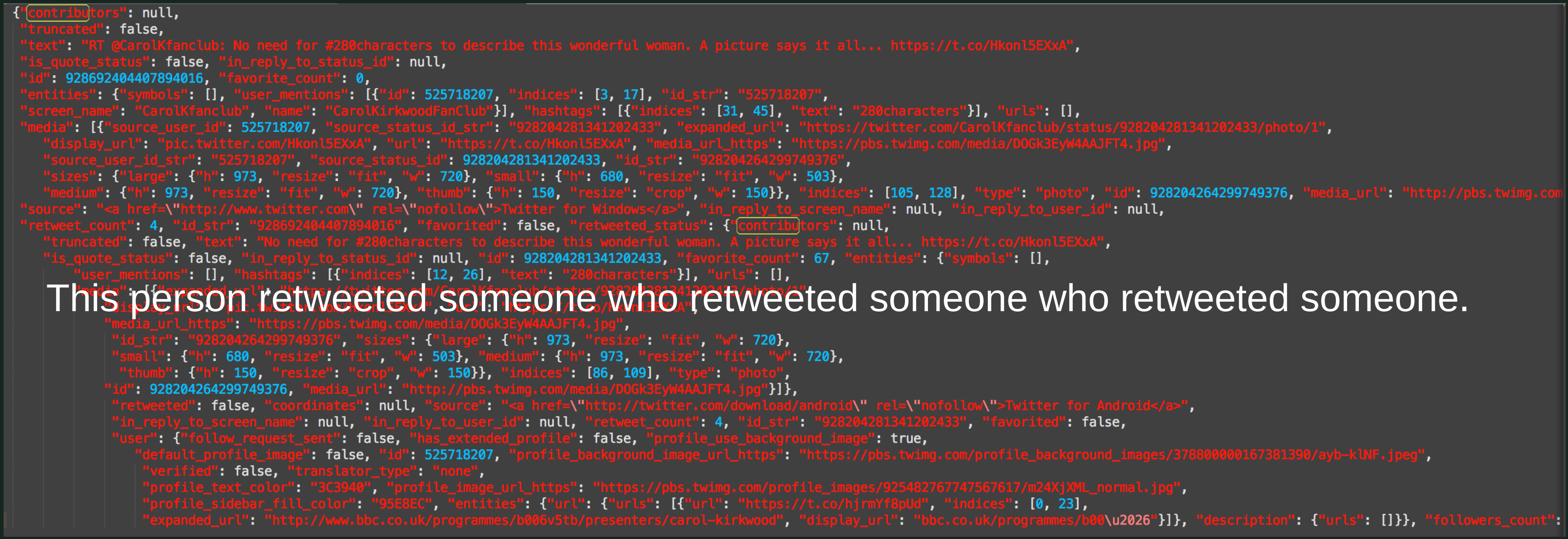
API Calling and Cleaning



```python
def tweet_search(api, query, max_tweets, max_id, since_id, geocode):
    ''' Function that takes in a search string 'query', the maximum
        number of tweets 'max_tweets', and the minimum (i.e., starting)
        tweet id. It returns a list of tweepy.models.Status objects. '''

    searched_tweets = []
    while len(searched_tweets) < max_tweets:
        remaining_tweets = max_tweets - len(searched_tweets)
        try:
            new_tweets = api.search(q=query, count=remaining_tweets,
                                    since_id=str(since_id),
                                    max_id=str(max_id-1))
#                                    geocode=geocode)
            print('found',len(new_tweets),'tweets')
            if not new_tweets:
                print('no tweets found')
                break
            searched_tweets.extend(new_tweets)
            max_id = new_tweets[-1].id
        except tweepy.TweepError:
            print('exception raised, waiting 15 minutes')
            print('(until:', dt.datetime.now()+dt.timedelta(minutes=15), ')')
            time.sleep(15*60)
            break # stop the loop
    return searched_tweets, max_id


def get_tweet_id(api, date='', days_ago=9, query='a'):
    ''' Function that gets the ID of a tweet. This ID can then be
        used as a 'starting point' from which to search. The query is
        required and has been set to a commonly used word by default.
        The variable 'days_ago' has been initialized to the maximum
        amount we are able to search back in time (9).'''

    if date:
        # return an ID from the start of the given day
        td = date + dt.timedelta(days=1)
        tweet_date = '{0}-{1:0>2}-{2:0>2}'.format(td.year, td.month, td.day)
        tweet = api.search(q=query, count=1, until=tweet_date)
    else:
        # return an ID from __ days ago
        td = dt.datetime.now() - dt.timedelta(days=days_ago)
        tweet_date = '{0}-{1:0>2}-{2:0>2}'.format(td.year, td.month, td.day)
        # get list of up to 10 tweets
```

my precious code to call Twitter API

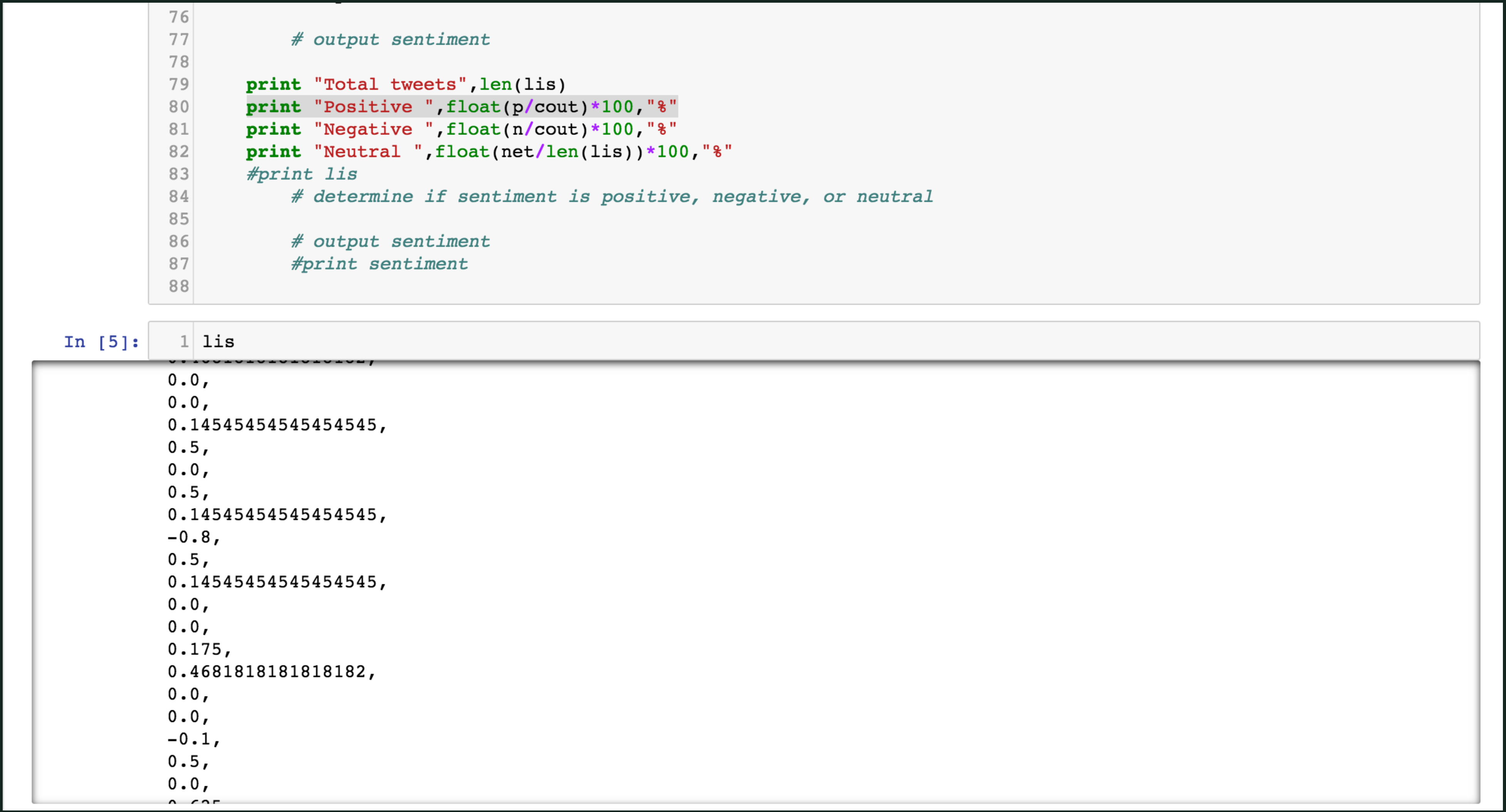
the reply i get



This person retweeted someone who retweeted someone.

```
{"contributors": null,
 "truncated": false,
 "text": "RT @CarolKfanclub: No need for #280characters to describe this wonderful woman. A picture says it all... https://t.co/Hkonl5EXxA",
 "is_quote_status": false, "in_reply_to_status_id": null,
 "id": 928692404407894016, "favorite_count": 0,
 "entities": {"symbols": [], "user_mentions": [{"id": 525718207, "indices": [3, 17], "id_str": "525718207",
 "screen_name": "CarolKfanclub", "name": "CarolKirkwoodFanClub"}], "hashtags": [{"indices": [31, 45], "text": "280characters"}], "urls": [],
 "media": [{"source_user_id": 525718207, "source_status_id_str": "928204281341202433", "expanded_url": "https://twitter.com/CarolKfanclub/status/928204281341202433/photo/1",
   "display_url": "pic.twitter.com/Hkonl5EXxA", "url": "https://t.co/Hkonl5EXxA", "media_url_https": "https://pbs.twimg.com/media/DOGk3EyW4AAJFT4.jpg",
   "source_user_id_str": "525718207", "source_status_id": 928204281341202433, "id_str": "928204264299749376",
   "sizes": {"large": {"h": 973, "resize": "fit", "w": 720}, "small": {"h": 680, "resize": "fit", "w": 503},
   "medium": {"h": 973, "resize": "fit", "w": 720}, "thumb": {"h": 150, "resize": "crop", "w": 150}}, "indices": [105, 128], "type": "photo", "id": 928204264299749376, "media_url": "http://pbs.twimg.com
 "source": "<a href=\"http://www.twitter.com\" rel=\"nofollow\">Twitter for Windows</a>", "in_reply_to_screen_name": null, "in_reply_to_user_id": null,
 "retweet_count": 4, "id_str": "928692404407894016", "favorited": false, "retweeted_status": {"contributors": null,
   "truncated": false, "text": "No need for #280characters to describe this wonderful woman. A picture says it all... https://t.co/Hkonl5EXxA",
   "is_quote_status": false, "in_reply_to_status_id": null, "id": 928204281341202433, "favorite_count": 67, "entities": {"symbols": [],
       "user_mentions": [], "hashtags": [{"indices": [12, 26], "text": "280characters"}], "urls": [],
       "media": [{"expanded_url": "https://twitter.com/CarolKfanclub/status/928204281341202433/photo/1",
         "display_url": "pic.twitter.com/Hkonl5EXxA", "url": "https://t.co/Hkonl5EXxA",
         "media_url_https": "https://pbs.twimg.com/media/DOGk3EyW4AAJFT4.jpg",
          "id_str": "928204264299749376", "sizes": {"large": {"h": 973, "resize": "fit", "w": 720},
          "small": {"h": 680, "resize": "fit", "w": 503}, "medium": {"h": 973, "resize": "fit", "w": 720},
           "thumb": {"h": 150, "resize": "crop", "w": 150}}, "indices": [86, 109], "type": "photo",
         "id": 928204264299749376, "media_url": "http://pbs.twimg.com/media/DOGk3EyW4AAJFT4.jpg"}]},
          "retweeted": false, "coordinates": null, "source": "<a href=\"http://twitter.com/download/android\" rel=\"nofollow\">Twitter for Android</a>",
          "in_reply_to_screen_name": null, "in_reply_to_user_id": null, "retweet_count": 4, "id_str": "928204281341202433", "favorited": false,
          "user": {"follow_request_sent": false, "has_extended_profile": false, "profile_use_background_image": true,
             "default_profile_image": false, "id": 525718207, "profile_background_image_url_https": "https://pbs.twimg.com/profile_background_images/378800000167381390/ayb-klNF.jpeg",
              "verified": false, "translator_type": "none",
              "profile_text_color": "3C3940", "profile_image_url_https": "https://pbs.twimg.com/profile_images/925482767747567617/m24XjXML_normal.jpg",
              "profile_sidebar_fill_color": "95E8EC", "entities": {"url": {"urls": [{"url": "https://t.co/hjrmYf8pUd", "indices": [0, 23],
              "expanded_url": "http://www.bbc.co.uk/programmes/b006v5tb/presenters/carol-kirkwood", "display_url": "bbc.co.uk/programmes/b00\u2026"}]}, "description": {"urls": []}}, "followers_count":
```

Preprocessing / NLTK



```
        # output sentiment

    print "Total tweets",len(lis)
    print "Positive ",float(p/cout)*100,"%"
    print "Negative ",float(n/cout)*100,"%"
    print "Neutral ",float(net/len(lis))*100,"%"
    #print lis
        # determine if sentiment is positive, negative, or neutral

        # output sentiment
        #print sentiment

```

```
In [5]: lis
```

```
0.0,
0.0,
0.14545454545454545,
0.5,
0.0,
0.5,
0.14545454545454545,
-0.8,
0.5,
0.14545454545454545,
0.0,
0.0,
0.175,
0.4681818181818182,
0.0,
0.0,
-0.1,
0.5,
0.0,
```

Ignore positive tweets/retweets/focus on negative

EDA \\\

36000 TWEETS PER HASHTAG

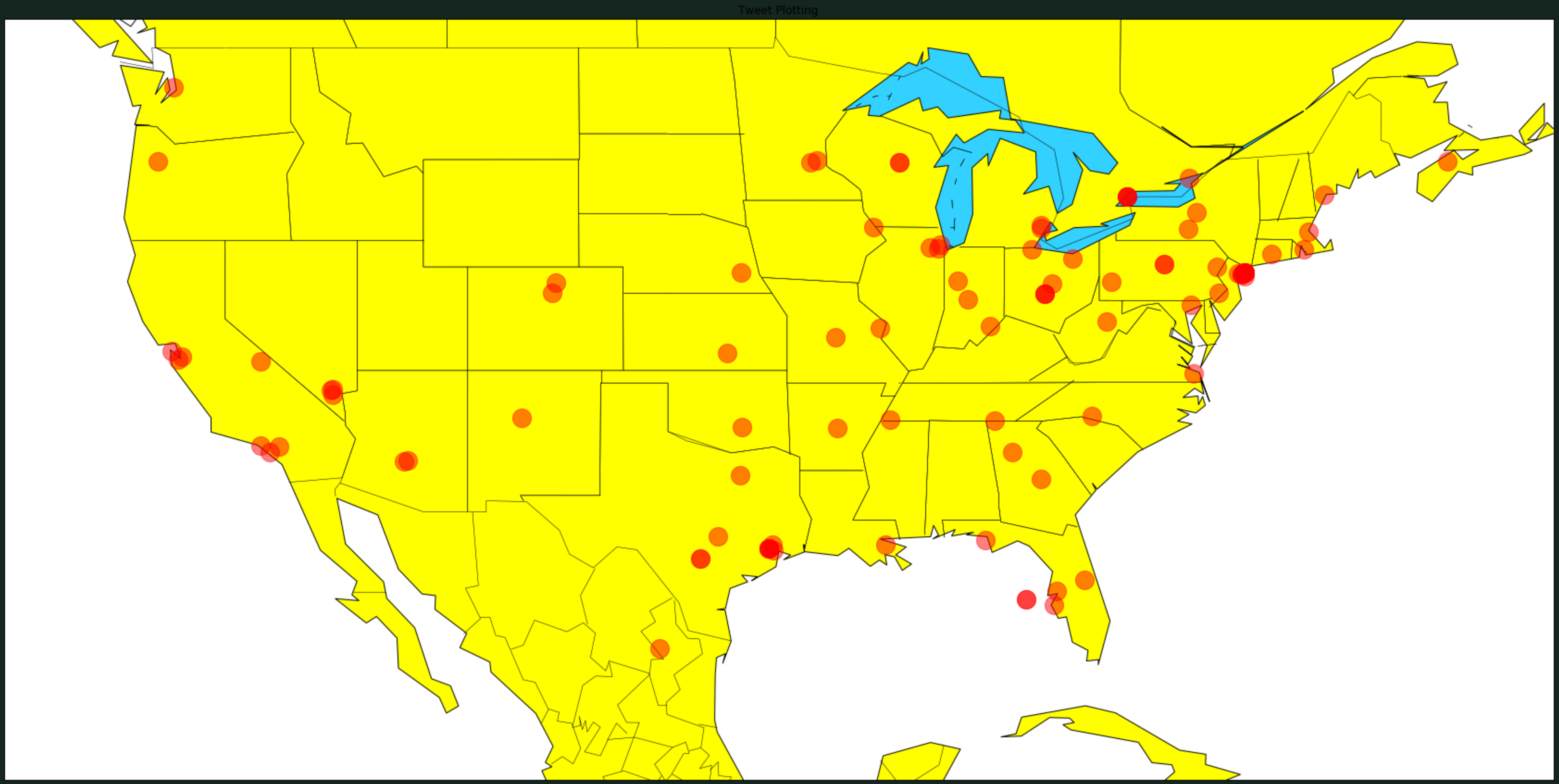
3%

WITH COORDINATES

Trimming - We are left with

- id
- message
- place {approx coordinates}
- retweet count
- user

Plotting Disaster locations



Best way to reach a disaster cluster

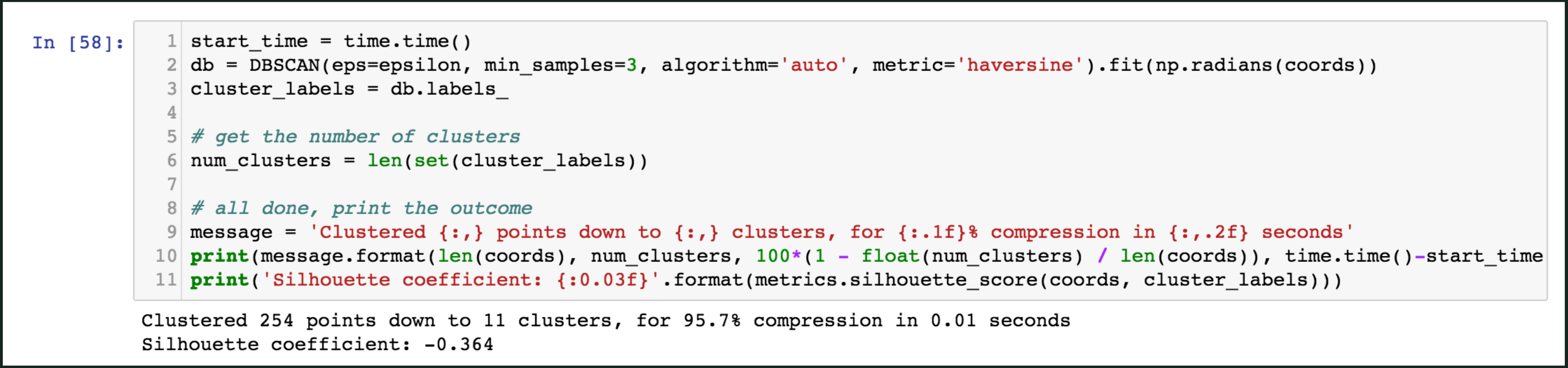
- drive a truck through the zones (road conditions, avoid danger)
- airdrop supplies (subject to getting stolen, accuracy of drop)

Clustering methods to consider

K-means
DBScan
optical cluster

K-means wins ?

Due to the nature of the tweets , DBScan gave very low silhouette scores. This is due to the fact that in this scenario, we are looking to cluster points and each "rescuer" can only accomodate so many rescuees. As such, it is easier for us to approach clustering using K-means where we can set the number of clusters (rescuers) and their size.

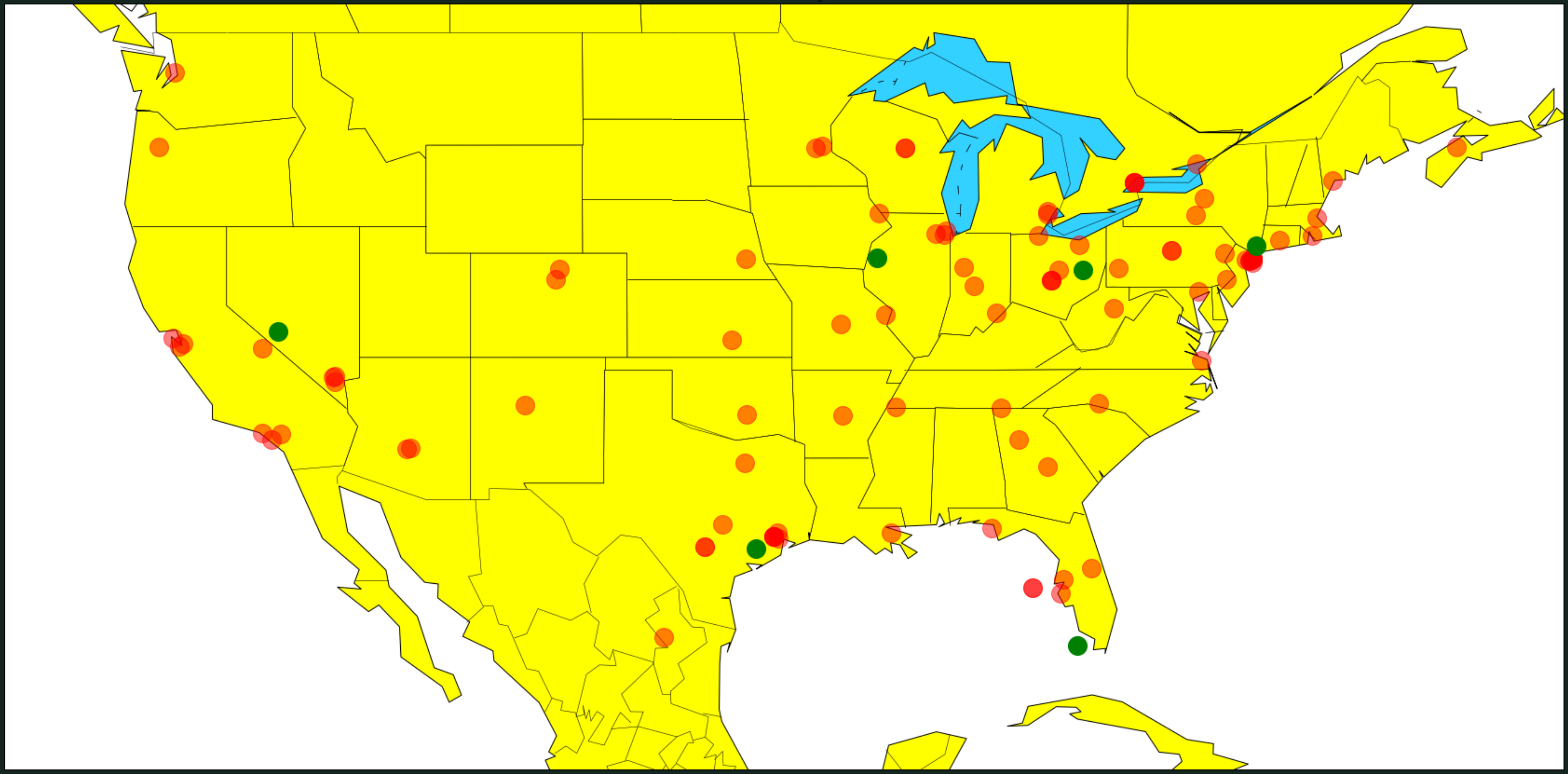


In [58]:
```python
start_time = time.time()
db = DBSCAN(eps=epsilon, min_samples=3, algorithm='auto', metric='haversine').fit(np.radians(coords))
cluster_labels = db.labels_

# get the number of clusters
num_clusters = len(set(cluster_labels))

# all done, print the outcome
message = 'Clustered {:,} points down to {:,} clusters, for {:.1f}% compression in {:,.2f} seconds'
print(message.format(len(coords), num_clusters, 100*(1 - float(num_clusters) / len(coords)), time.time()-start_time
print('Silhouette coefficient: {:0.03f}'.format(metrics.silhouette_score(coords, cluster_labels)))
```

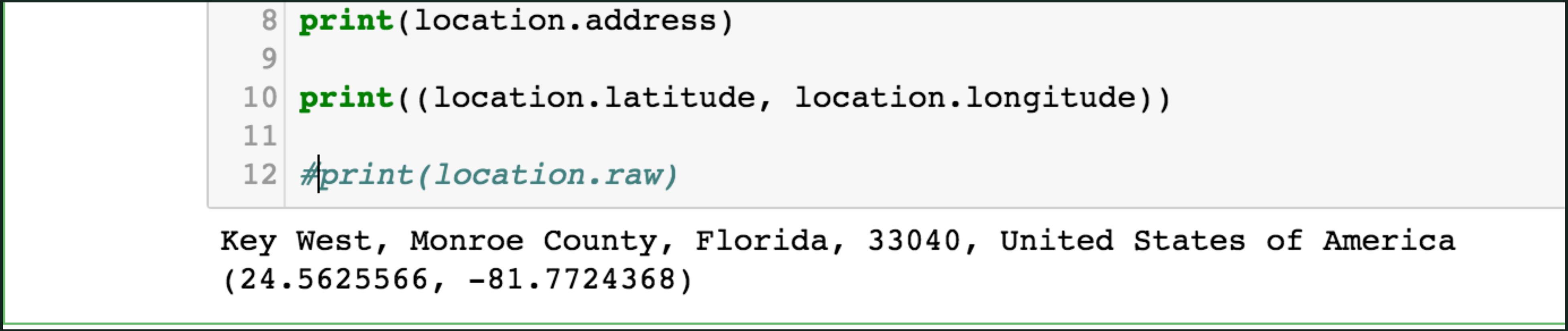
```
Clustered 254 points down to 11 clusters, for 95.7% compression in 0.01 seconds
Silhouette coefficient: -0.364
```

Rescue point plotting

Tweet Plotting



Reasonably accurate address to begin rescue operations



```
print(location.address)

print((location.latitude, location.longitude))

#print(location.raw)
```

```
Key West, Monroe County, Florida, 33040, United States of America
(24.5625566, -81.7724368)
```

The main rescue point for the points in cluster 15