Natural Disasters Classifying natural disaster occurrences, types and urgency in realtime using twitter data

February 18, 2021

Prepared by: Noah Zuckerman, Mathea Stevens, Zachary Katsnelson, Maya Morales

How we collected tweets in realtime using both twitter's API and web scrapers

100 accounts



Leveraged the most popular/followed twitter disaster news and/or relief accounts (Redcross, unicef, disaster update, etc)

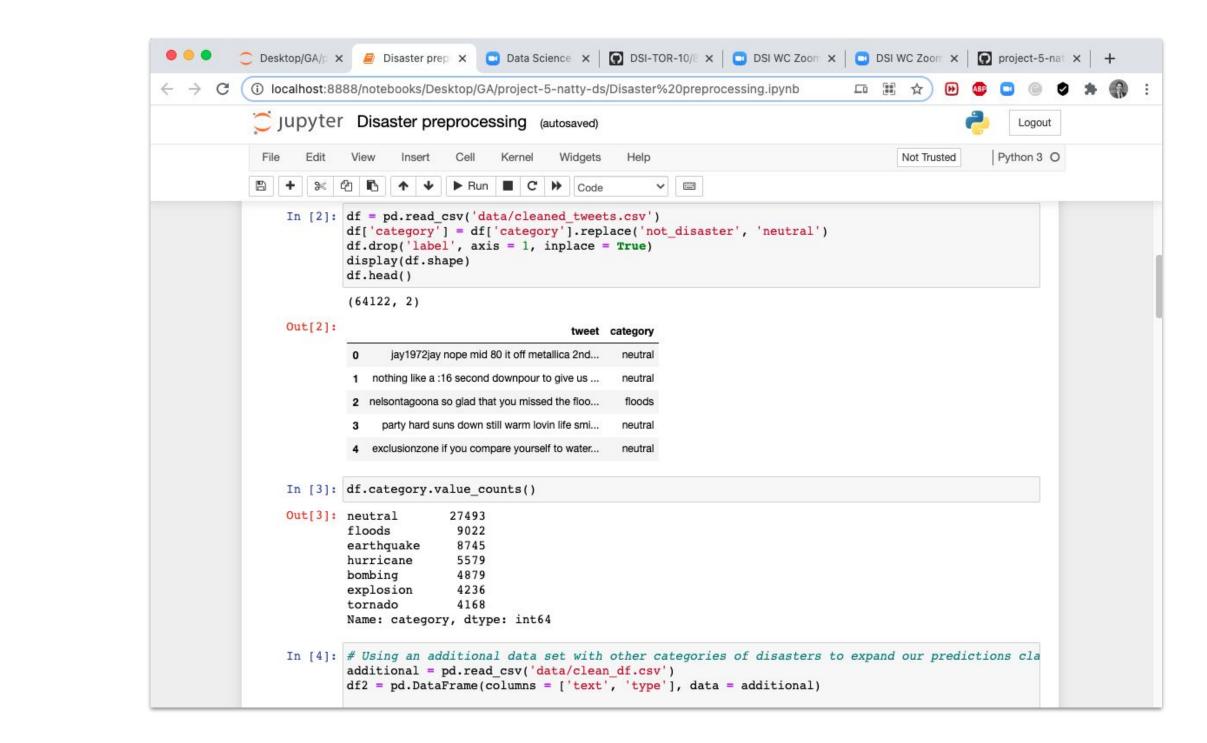
Used Tweepy module with Twitter
API credentials to pull the most
recent tweets and trending
hashtags from these accounts

70 keywords

Used the closest **Word2Vec** key word matches to our target natural disaster labels (earthquake, hurricane, etc)




Implemented **SNSscrape** module to query twitters front end for any posts mentioning our **70 relevant** keywords and trending hashtags



- Built our API calls, data processing, location gathering, model predictions and key insights into a single application
- New data is fed into the model every hour to be pre-processed, modelled and displayed
- End user sees the most relevant disasters types along with the location where they are occuring

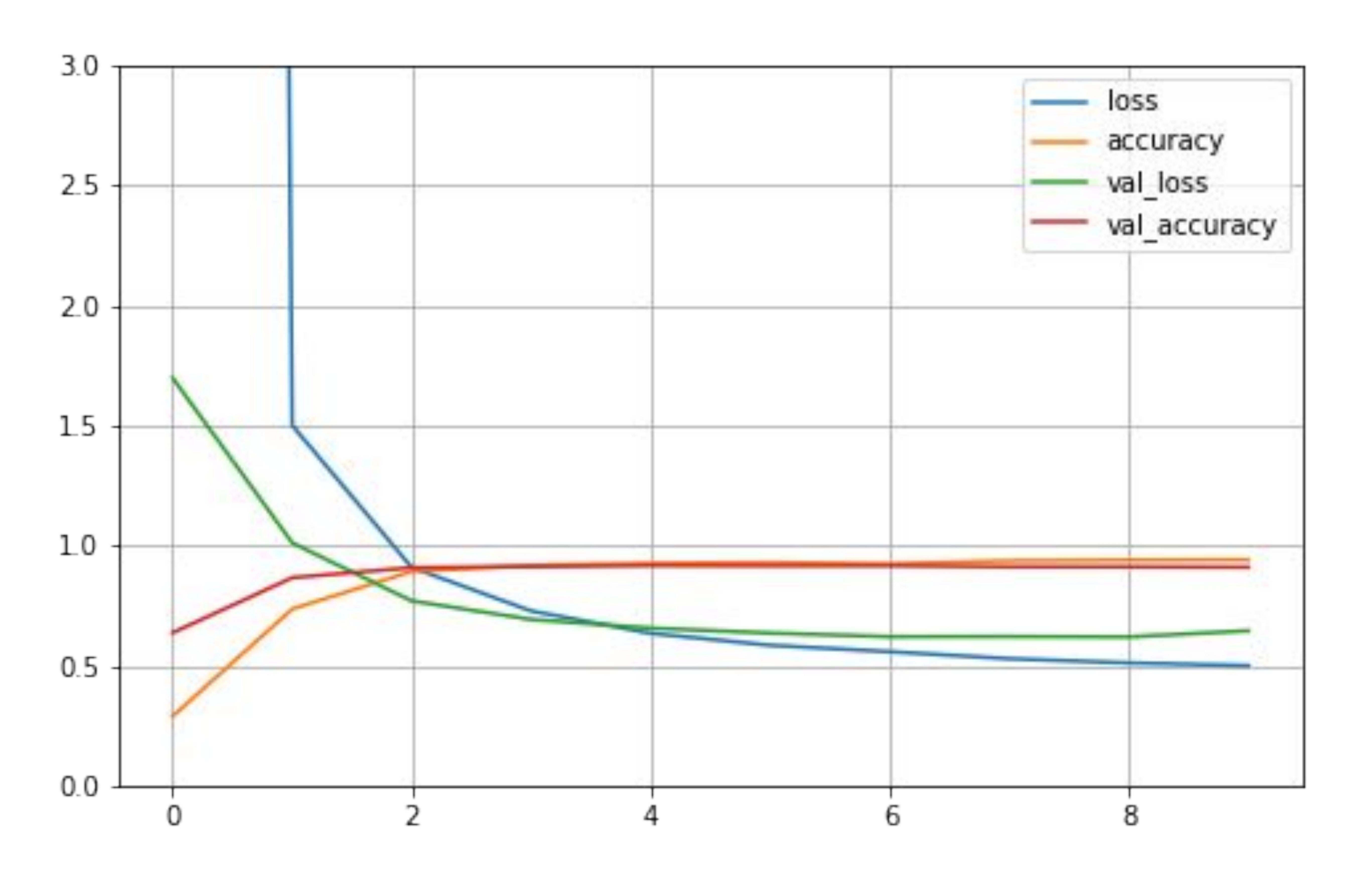
Build a classification model to accurately predict disaster types from tweets in real time

```
Time: 19:37:12 | {'covid19': 48, 'earthquake': 17, 'resilient': 10, 'cities': 10, 'iati': 6} | MODEL PREDICTIONS:
[['There are 675 mentions of hurricane'], ['There are 291 mentions of mudslide'], ['There are 226 mentions of ear
thquake'], ['There are 201 mentions of neutral'], ['There are 189 mentions of tornado'], ['There are 95 mentions of
bombing'], ['There are 64 mentions of noreaster'], ['There are 64 mentions of floods'], ['There are 29 mentions of
wildfire'], ['There are 28 mentions of explosion']],
location
           No Location Available
bombing
            friendly
            pace
            page
earthquake No Location Available
                                    . . .
wildfire
            middle
            mission
            ontario
            union
Length: 115, dtype: int64
```



Model evaluation and performance

Results:



Methodology:

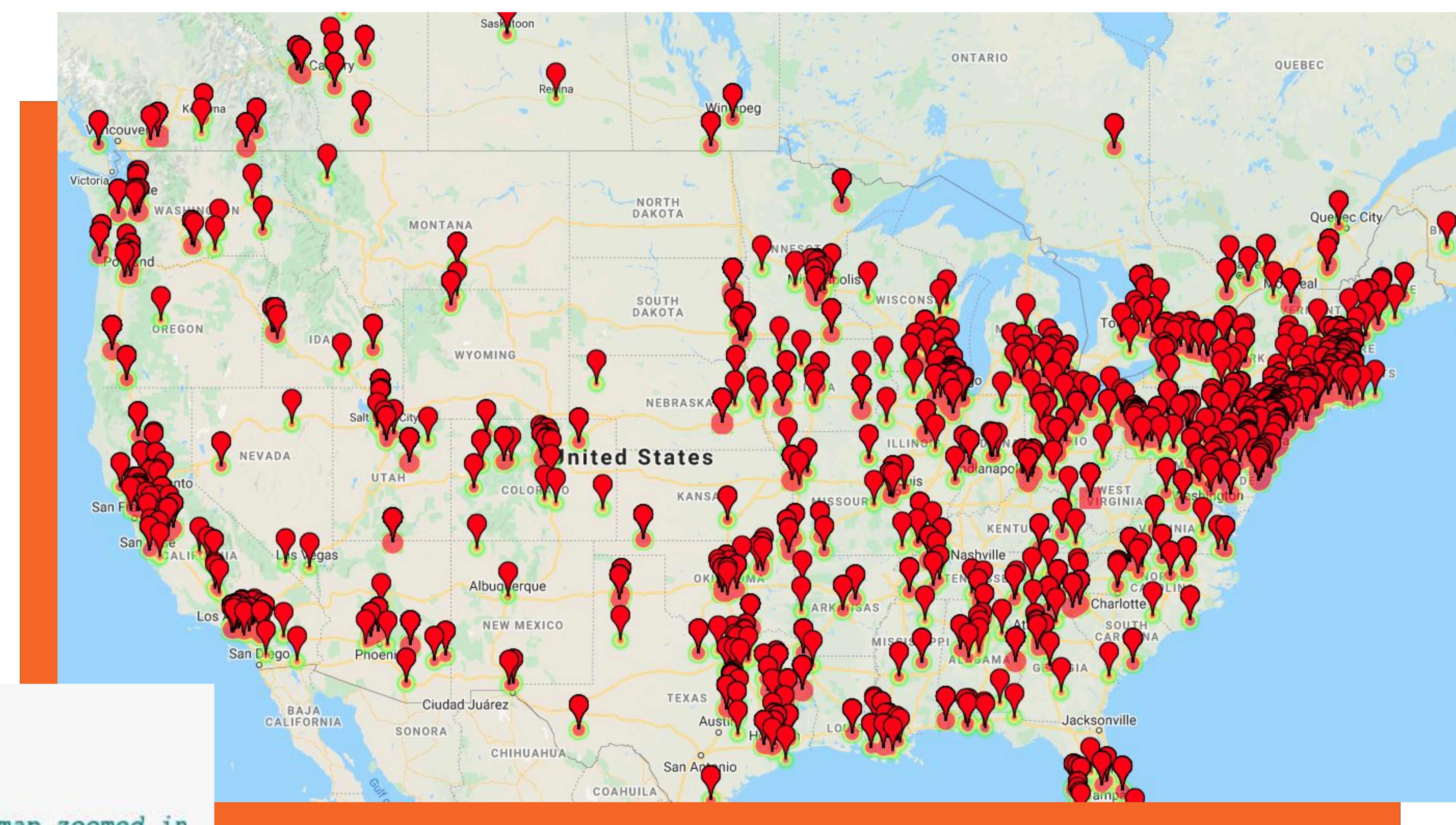
- Tokenized, lemmentized and prepared text data for a Word2Vec embedding layer
- Leveraged the bidirectional learning of recurrent neural network
- Gradually increase model complexity, then implemented regularization and dropout layers to minimize bias.

Mapping locations in tweets

```
location = []
array = [[]]
for tweet in df_dis['processed_tweets']:
    for word in tweet:
        try:
            city names[word]
            location.append(tweet)
            location.append(word)
            array.append(location)
            location = []
        except:
            location.append(tweet)
            location.append('No Location Available')
            array.append(location)
            location = []
```

```
lat = []
long = []
latitude list = []
longitude list = []
for word in X['location']:
        coords dict[word]
        lat.append(coords_dict[word][0])
        latitude list.append(lat)
        long.append(coords_dict[word][1])
        longitude list.append(long)
        lat = []
        long = []
    except:
        lat = []
        long =[]
```

Visualizing the natural disasters in real time...



```
#Import important libraries
import gmplot
import numpy as np

# declare the center of the map, and how much we want the map zoomed in
gmap = gmplot.GoogleMapPlotter(0, 0, 2)
# plot heatmap
gmap.heatmap(flat_lat, flat_long)
gmap.scatter(flat_lat, flat_long, c='r', marker=True)
#Your Google_API_Key
gmap.apikey = "AIzaSyA5D2bG0St11zmF66yo84ZXg1sEufc-4RM"
# save it to html
gmap.draw("./maps/country_heatmap2.html")
```

Next steps and improvements

Given more resources and time the following next steps would be taken in order to improve the applications functionality and accuracy:

- Continue to gather all of our analyzed tweets in a database along with our predictions for post-mortem analysis on our models accuracy
 - With more resources, we could have an offshore team work through our scrapped tweets to check labels and re-label and misclassified tweets
- Continue to fit and train our model on our growing database of tweets (leveraging other pre-labeled disaster tweet datasets wherever possible)
- Experiment with more model architectures (including pre-trained models) on our historical dataset to identify improvements to be made to our core prediction model
- Implement a flask GUI interface on top of our application giving the end user more control over the interface (location filtering, real-time map, email notifications etc) as well as a better user experience
- Leverage twitters paid API account to expand the scope of our daily api calls as well as the tweet parameters and metadata feeding into our application