Andrea Cruz Castillo
Harini Venkatesan

CS172 Final Project Report
Tweety-project

Part 1. Crawler:

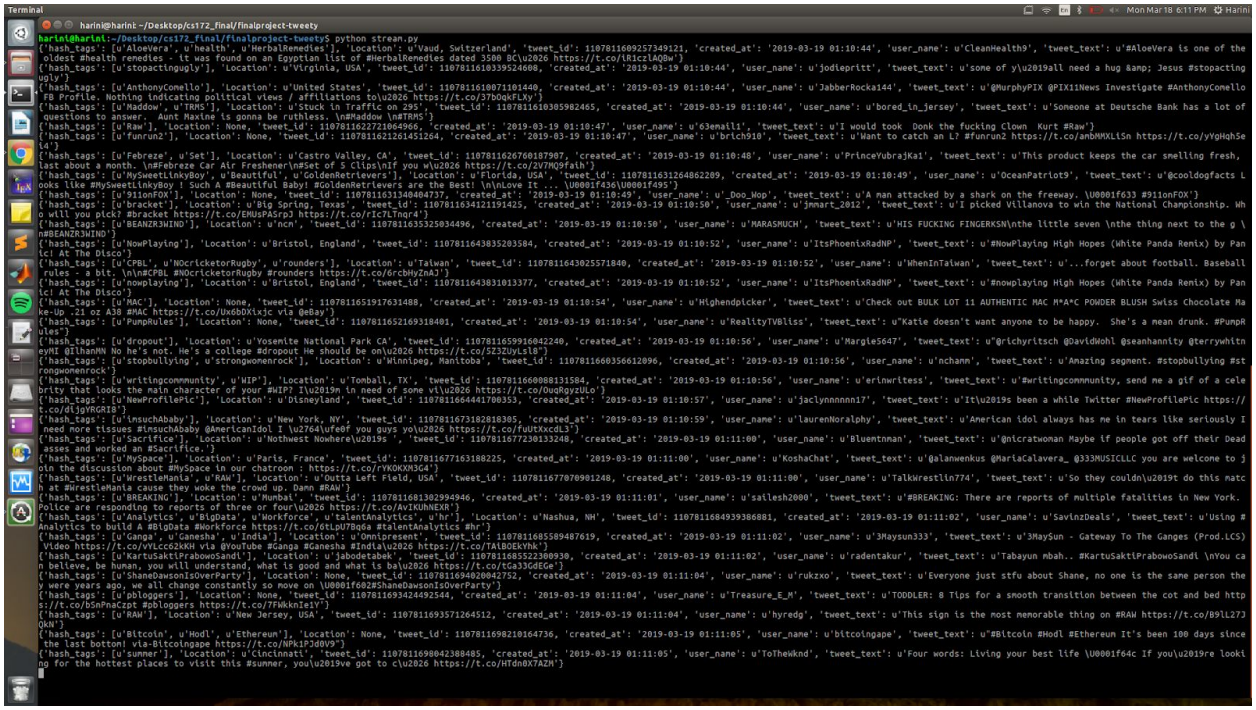
Collaboration Details: Harini wrote a python script that used a twitter streaming API called tweepy. The API keys for our twitter account was inserted in the script and she collected tweets as they were streaming while filtering out the location. This API uses a basic listener class called StdOutListener() which performs streaming and creates json objects which we then dump into a json file. We collected around 100,000 tweets which ended up being 15MB.

Limitations of the system: Filtering using location took a long time since not a lot of users have location enabled on twitter. So we streamed tweets by collecting the user's location and not the lat and long values of the tweet.

Instruction on how to deploy the crawler:

1. Create a twitter account and apply for twitter developer account
2. After creating an account, find your api key, secret key, access token, and secret token under Keys and Access Tokens and add to stream.py
3. `sudo pip install tweepy` to install tweepy
4. Run `python stream.py`
5. Collected tweets will be stored in a file named `CS_data.json'

Screenshots showing the system in action:



Andrea Cruz Castillo
Harini Venkatesan

Part 2 – Indexer:

Collaboration Details: Harini wrote a script that connected to elasticsearch and indexed each tweet one at a time. However, that was not efficient, and we could not figure out how to integrate it to our Django app. Initially, we had included it in out views.py file but that meant that we would be indexing the same documents every time there was a query. Thus, we decided to bulk load our data through the command line instead.

Limitations of system: The indexing has to be done as a separate process after collecting the tweets. The only way we got it to work was by bulk loading data from the command line.

Instructions on how to deploy the system:

1. After collection the tweets, run
    a. `python json_to_es.py` # Input file is the file produced by stream.py
2. This will output a file i.e. tweets.json. From command line run
    a. `curl -X POST "localhost:9200/_bulk" -H "Content-Type: application/json" --data-binary @tweets.json`
3. Then check on your browser that the index was created with:
    a. `localhost:9200/_cat/indices`

Part 3 – Extension:

Collaboration Details: Andrea worked on creating the web interface using Django. She defined the templates and views of each window that the user can interact. She connected the Django application to Elasticsearch and wrote code to perform queries. Andrea also found the Textblob libraries to perform sentiment analysis on the tweets and display them by category. Harini worked on displaying tweets on the map. She used folium and pandas to create a map. She also used a file called worldcities.xlxs which was an open source dataset with lat and long values of cities and towns around the world to parse out the appropriate location of the user, find their geolocation, and display it on the map.

Motivation: We decided to add the features of a map view and sentiment analysis to extend the basic search feature. The map view allows users to make queries and displays the results on the map. This can be a great tool if the users are interested in the geolocation of a specific query. For example, they can search for a movements across the globe like #globalwarming and see how different parts of the world are reacting to it. The sentiment feature was added so that users can determine how the population feels about a certain topic as well. It classifies tweets as negative,

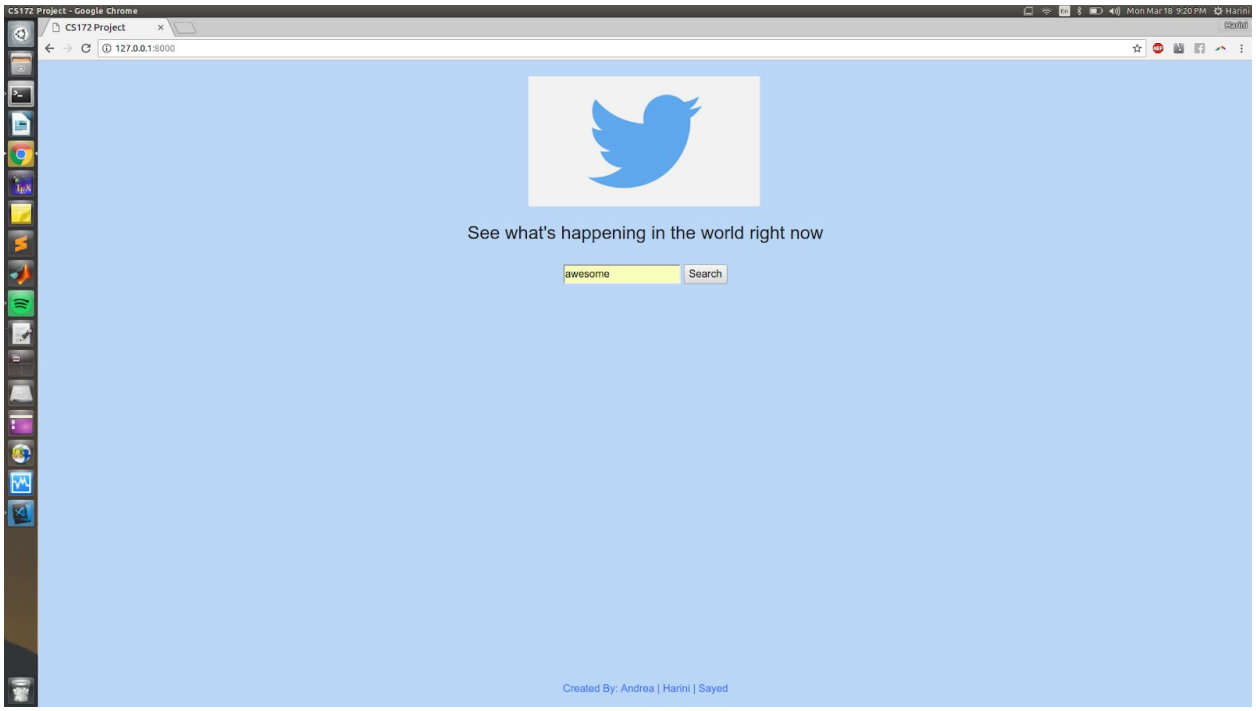
neutral, or positive. Thus, a user can search for a query like #globalwarming and see how people feel about it.

Limitations of System: It would have been nicer to incorporate both features, map view and sentiment, into one view such that the system would display pins on the map with different colors to show the sentiment of the tweet.

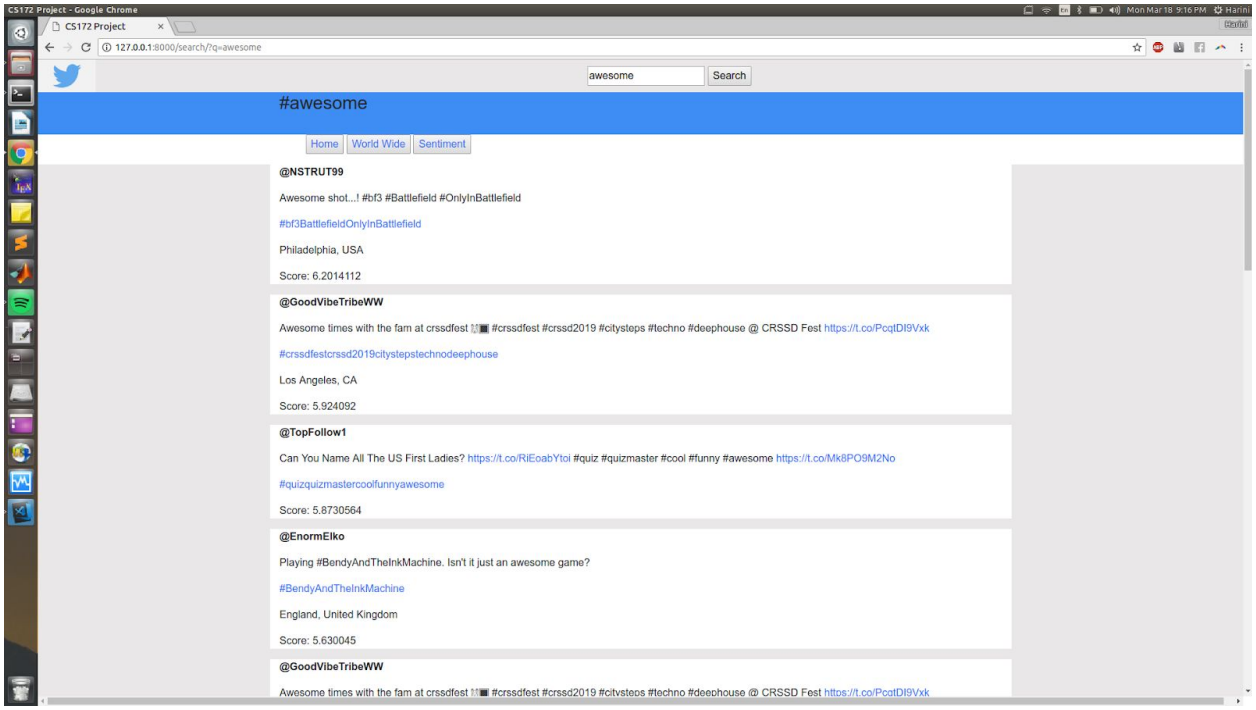
Instructions on how to deploy the system:

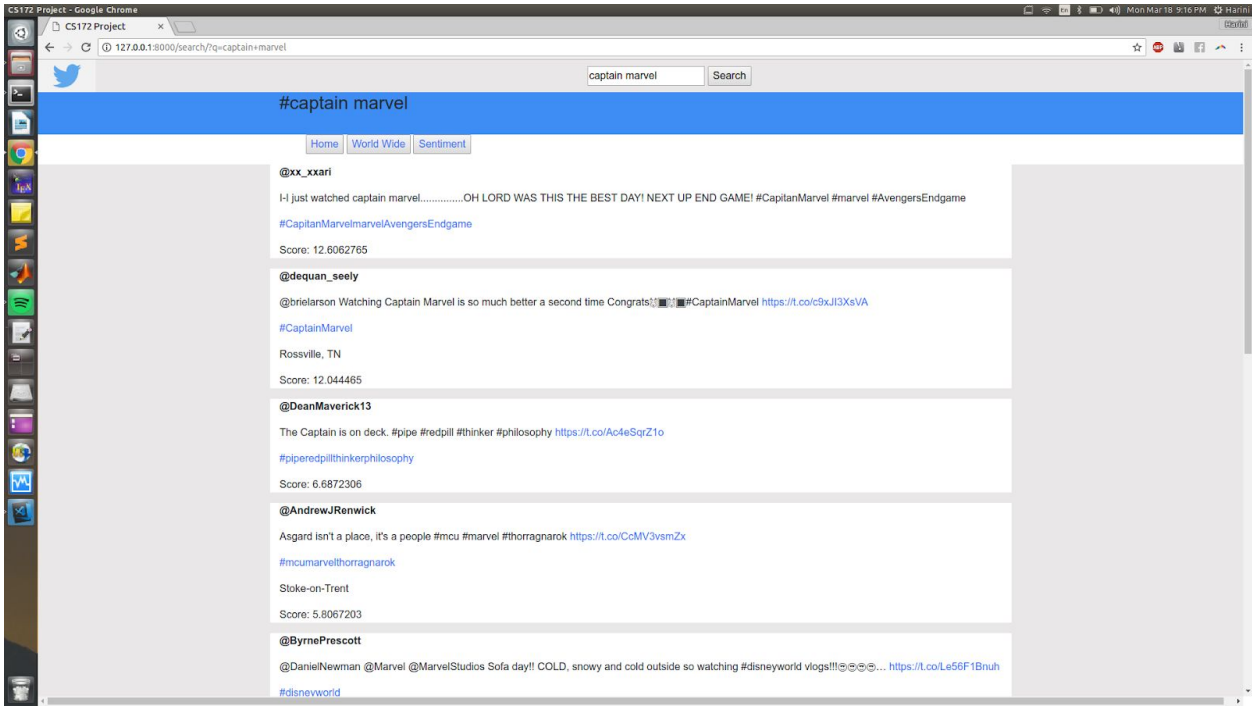
1. Move to CS172_Project:
   Note: there are 2 directories with the same name, one is a subdirectory. You must be in the first CS172_Project directory such that you can see the following files: manage.py, db.sqlite3, main/, *_pycache_*/,CS172_Project/, worldcities.xlxs
2. cd CS172_Project
3. Then run the server with:
4. python3 manage.py runserver
5. Open up your browser at: http://127.0.0.1:8000/

Screenshots of system in action:



Andrea Cruz Castillo
Harini Venkatesan





Andrea Cruz Castillo
Harini Venkatesan