Symptom Tracker

Priya Balachandran Mary
Department of Computer Science
State University of New York at Albany
Albany, New York 12222, USA
pbalachandranmary@albany.edu

Abstract - **Symptom Tracker is an online tool that helps a user garner knowledge about possible disease outbreaks in the United States from tweets in the twitter space.**

The Symptom tracker helps detect outbreaks by providing statistical data and visuals on the number of people tweeting about a disease symptom.

The tracker takes a symptom and a time range as input and tracks the number of hits for the selected symptom for 1217 major cities. It generates heat maps, word clouds, scatter charts and clusters of closely located cities with high tweet count. We discuss the same through the remainder of this paper.

Keywords – clusters

I. INTRODUCTION

A. *Motivation*

Detecting disease outbreaks ahead of time helps in planning and taking proactive measures to avoid the risk of a disease spreading and prevent epidemics.

B. *Problem Statement*

To detect a disease outbreak, government health officials would need to record the possible occurrence of each illness and symptom for almost every individual. The task of manually collecting such data would be cumbersome and costly.

However if we can harness a public source like social networking sites, where almost every individual can state if he/ she has a symptom, things would be a lot simpler and would provide a better chance at forecasting of a possible disease outbreak. This would address the issue of data collection.

C. *Significance of the Problem*

Social networking sites like Twitter, Facebook etc. have taken center stage in today's world. With many tweeting and posting very random tweets and statuses about symptoms like headaches and the flu, some of this data can be converted and treated as a source of information when it comes to disease outbreak detection. If health officials garner such data possible epidemics can be detected and avoided by different measures.

This information can also help in analyzing trends over time to gain knowledge about seasonal symptoms like the flu and which location require attention and thereby better costing decisions for a government when it comes to providing medical care during an outbreak

We introduce a tool called the 'Symptom Tracker to gather information regarding symptoms and location from tweets posted by twitter users space and utilize the data to provide visualizations that help convey the possibility of the start of an epidemic or an outbreak.

II. RELATED WORK

A. *WebMD Symptom Checker*

The WebMD Symptom checker tracks the cold and flu symptoms for the whole of United States. The WebMD Symptom checker covers all cities in the United States which is much more than the cities we cover in our proposed approach.

However, WebMD Symptom checker does not utilize twitter data to mine data related to flu or cold. Instead WebMD asks the user logged in permission to access his location and asks the user if the user feels sick.

WebMD Symptom checker also assigns color keys to each city to indicate if the city has mild, moderate or severe cold and flu symptoms.

We track the same in our system with the help of tweet counts in city maps and heat maps.

Unlike the WebMD Symptom checker our system does not limit its scope to just the cold and flu symptoms. It allows the user to send queries for a predefined set of common symptoms mentioned later in the paper.

III. PROPOSED APPROACH

The Symptom tracker takes in tweets from the backend system data collection system mentioned in IV. These collected twitter set consists of tweets from 1217 different cities in the United States with any keyword

relevant to a symptom mentioned in a table in the backend system. The symptom table is constantly queried for symptoms and the location for locations of the 1217 cities.

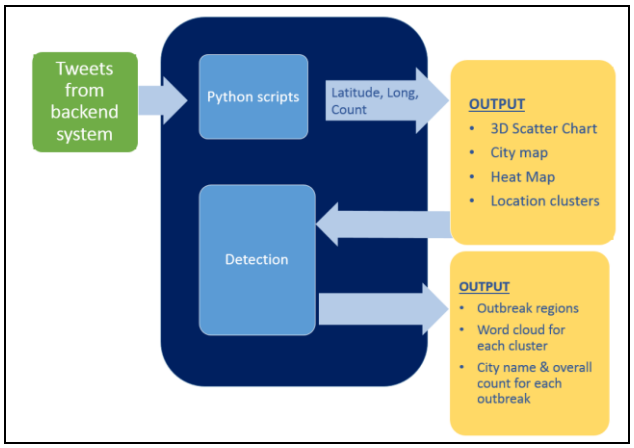

Figure 1: Proposed Plan Layout

These tweets are processed with the help of python scripts to make 3d visualizations and charts to study outbreaks and analyze trends if any.

The python scripts generate city maps, heat maps, 3d scatter plots, word clouds and location clusters for each symptom.

The output can be later analyzed and studied for different trends for each of the symptoms.

IV. IMPLEMENTATION

The Symptom Tracker system architecture comprises of a front end system which allows the user to query the twitter space for relevant symptom tweets and a backend system to support this operation.

A. Front end system

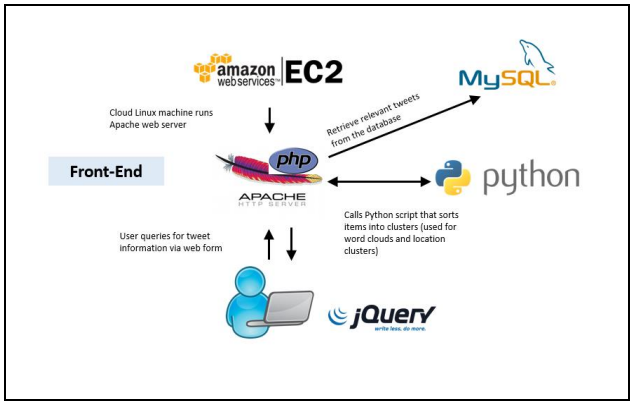

Figure 2: Frontend System

The Symptom tracker system provides its user with the web page which allows the user to choose from a set of different symptoms like flu, fever, cough, cramps, diarrhea, hangover, headache, nose bleed, runny nose, sore throat, vomiting and work-related stress.

The web page also provides input fields for start date and end date which the user can utilize to view symptom tracker results between the specified span of time using the tweets collected by the Amazon EC2 engine.

The input from the user is then sent to an Apache web server run by the Amazon EC2 machine. Based on user input the web server gets the relevant tweets from the database.

The web server also calls a python script to form clusters of closely located cities based on latitude, longitude information if the user calls for it explicitly.

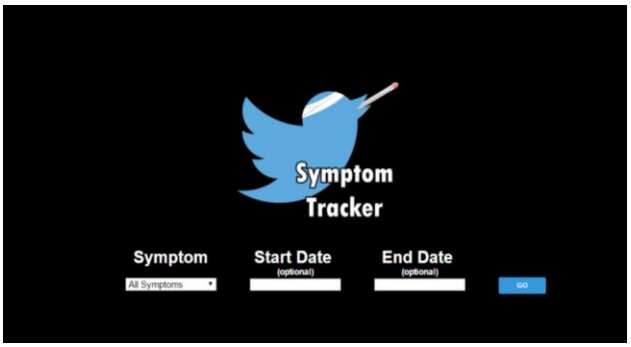

Figure 3: Front end page for end user to post symptom query

B. Back end system

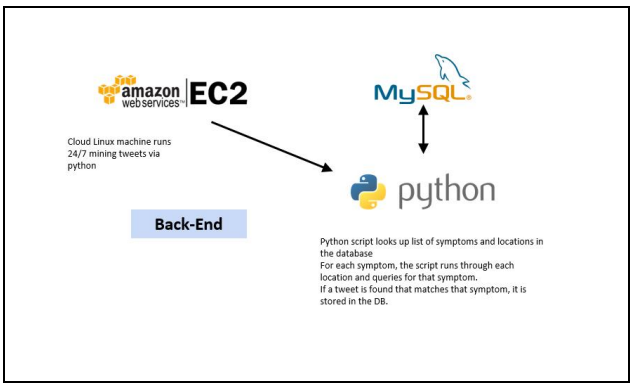

Figure 4: Backend System

The backend system consists of the following components:

1. Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) [2] is a web service running on Linux. This machine runs a python script which mines for relevant tweets in the twitter space 24x7.

2. MySQL Server instance

The MySQL Server instance is used to collect the relevant tweets, storing locations, symptoms and for accessing Twitter API keys.

The database consists of tables for the following:

a) Locations: The database stores information related to 1217 major cities in the United States. The fields in the table include:

1. loc_id: A unique location id is assigned to each city.
2. city: This fields stores the name of the city
3. state: This field stores the a two letter abbreviation for the state under for the city.
4. lat: This field stores the latitude for each city. The latitude is queried by the python script to pass to the REST API search function to fetch tweets relevant to a specific location.
5. lon: This field stores the longitude for each city. The longitude is queried again by the python script to serve as a parameter for geocode for the REST API query in order to fetch tweets relevant to a specific location.
6. radius: The radius field is used to define the radius around the specified lat lon pair for a specific location and to fetch tweets around that location. The default radius for each location is set to 5 miles.

b) Symptoms: The symptoms table consists of:

1. symptom_id: unique symptom id
2. label: label for each symptom
3. query: query for each symptom passed to the REST API
4. disabled: to enable and disable search for a symptom.

c) Tweets: The tweets table consists of the following fields which is used by the python scripts later. The tweets mined by the Amazon EC2 machine is stored here.

1. text: stores the tweet text based on the input provided to the search function.
2. loc_id: location id for the tweet
3. created_at: This field stores the tweet creation date. This field is later used to fetch tweets between the time range provided as input by the user in the front end.

d) Twitter developer keys: This table stores a set of 3 twitter application developer keys which are used in succession after every 15 minutes to retrieve relevant tweets.

3. Python scripts for data collection

The python scripts for collecting tweets are run on the Amazon EC2 machine.

The python script takes in input from the database for symptoms and location and utilizes the set of 3 different developer keys stored in the database in intervals of 15 minutes to get relevant tweets.

The script runs queries for each symptom recorded in the database for all 1217 locations by utilizing the query text from the symptoms table in the backend.

V. DATA COLLECTION STRATEGY

A. Initial data Collection plan: The initial data collection plan involved capturing tweets from last winter till the end of May to observe trends in flu statistics using the twitter REST API.

B. Constraints: The twitter search API limits search for past queries. The API allows to track tweets from the past week and not more

C. Current data collection strategy: The current strategy stores tweets from the commencement of data collection till the end of May. We utilize this data for for our project.

The data collection strategy setup now has collected only a few tweets which gives us required result but has become a major setup when it comes to trend analysis for a symptom especially to track changes in trend of symptom tweets with respect to change in season and climate.

VI. DISCUSSION

The following discussion considers ‘flu’ as the symptom provided as an input by the user.

A. Results

a) Word cloud: The Symptom tracker generates word cloud for the symptom selected by the user from the front end. It is programmed to generate 3 word

clouds based on the collection of tweet text file given as input to the python file. The word cloud allows us to gather information about keywords that frequently appear in the tweets for a particular symptoms.

Findings: As we designed the word clouds we found that the keyword 'Flu', 'Bird', 'USDA' appeared prominently in the word cloud when there was a talk about bird flu in the twitter space.

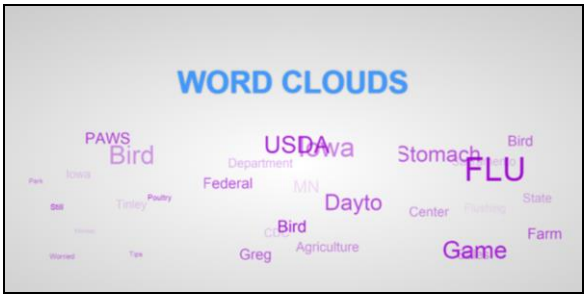

Figure 5: Word clouds generated for symptom 'flu'

b) City Map:

The city map is a map generated by the system using the 3rd party tools called High Charts. The city map gives us the total number of tweets, information about the city and state, latitude and longitude information.

It also gives us the tweet count for the symptom provided by the user as input and for all the 1217 cities stored in the database.

Findings: A few cities like Chicago, Buffalo showed high tweet counts for the symptom flu but we weren't able to decide on a trend here since our data collection strategy could not account for the tweets that could have been utilized to analyze trends in flu outbreak detection since [1] shows prominent rise in flu during the months of December, January and February.

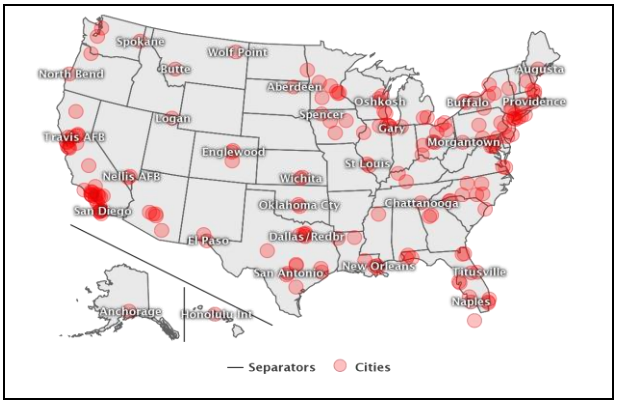

Figure 6: City map generated for symptom 'flu'

c) 3D Scatter Chart

The symptom tracker utilizes high charts again for plotting a 3D scatter chart. This chart shows a visualization of the tweet count with respect to latitude longitude information about each of the 1217 cities.

Findings: We noticed that for the selected symptom (flu) some cities show higher tweets counts. Also for the symptom flu we see that the latitude and longitude does not seem to show a trend for cases when the tweet count is exceptionally high.

The reason is because our data collection strategy could not collect data from the past during the winter when the tweets could have been a rich resource for detection of the flu symptom.

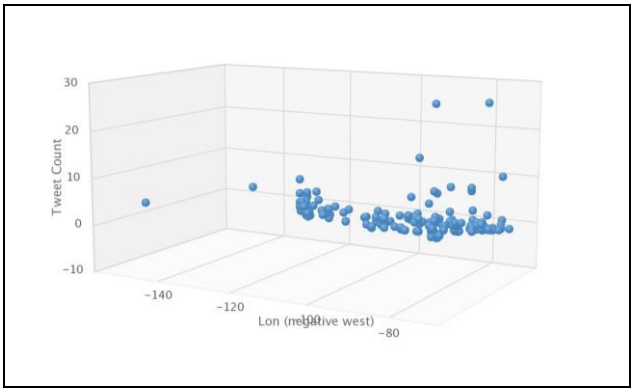

Figure 7: 3d scatter chart generated for symptom 'flu'

d) Heat Map

The heat map shows how often a city tweets about a symptom. The color legend at the bottom of the chart shows how frequently a city has quoted the symptom in its tweets.

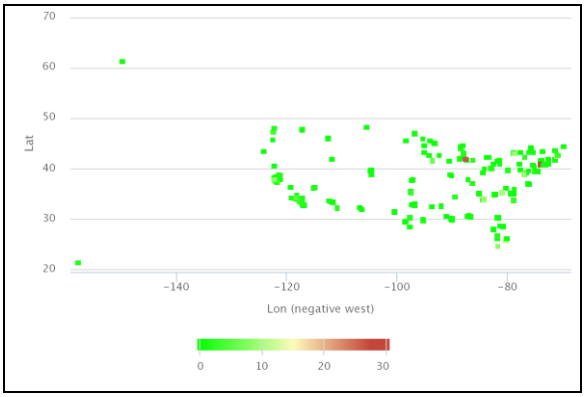

Figure 8: Heat map generated for symptom 'flu'

Findings: The heat map showed almost similar color information for all cities when it came to the flu since the dataset we had was not good enough. A dataset of tweets spread over the all seasons would have been the perfect dataset for such trend analysis.

e) Location Clusters

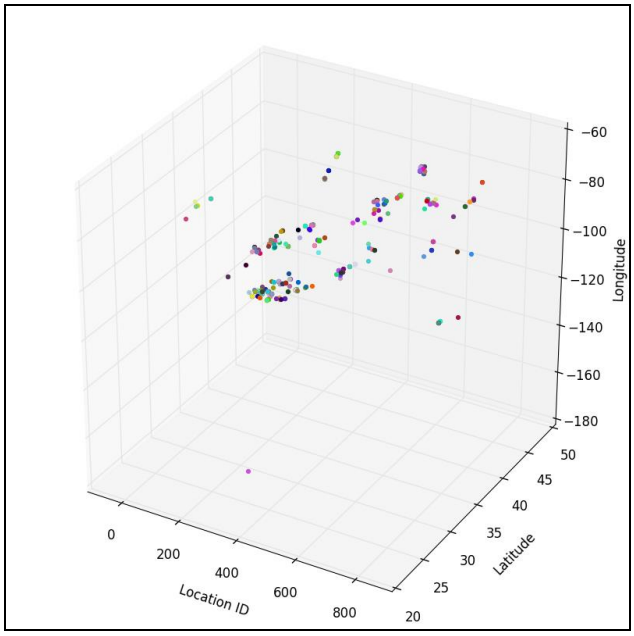

Figure 9: Location clusters generated for symptom 'flu'

Lastly, the apache web server calls a python script to run over all locations and calculate location clusters. Each location cluster represents a set of tweets for each city. The plot however helps us visualize clusters differently.

Upon eyeballing, some of the closely located cities form a cluster of cluster where the main cluster consists of tweets from a set of closely located cities and each sub cluster represents a set of tweets tweeted from that location. Each point in the cluster represents a tweets.

The main clusters utilize the location ID, latitude and longitude information to form the cluster visually.

However through the python script we can only generate a cluster for each location and not for a set of location.

The Symptom tracker also provides an option to download output file which contains latitude longitude information of each tweet for each location cluster.

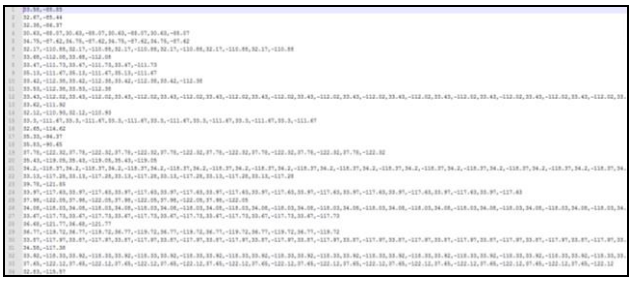

Figure 10: Location clusters output file for each unique location ID

f) Symptom clusters

This section was scraped from the. The symptom clusters were clusters generated for every symptom in the database.

Tweets from the database was passing to the k means clustering algorithm with the number clusters set to the number of symptoms the Symptom tracker tracks.

Upon passing the tweets the k means algorithm we derived 10 centroids one for each symptom. These centroids were plotted and the remaining tweets were visualized in the plot around the centroids by computing sentiment score for each tweet.

The sentiment score for each tweets was incorporated with the centroids calculated earlier for each symptom to plot the tweets related to a symptom around its centroid.

The formula utilized for deriving points to represent tweets for each of these clusters is as follows:

tweet_point = k_means_symptom_centroid + sentiment_score

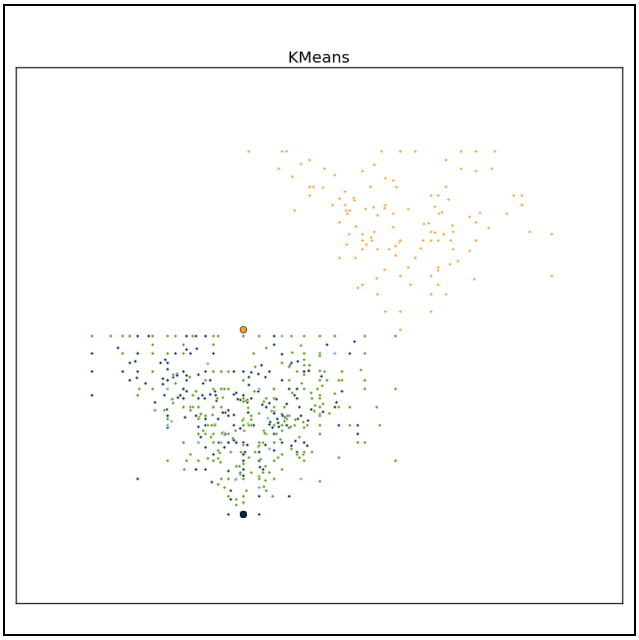

Fig 11: Symptom clusters generated using k means clustering algorithm

VII. FUTURE WORK

Our current system does not check if a tweet is relevant. This leads to collection of some irrelevant tweets in the database from the Amazon EC2 machine. We plan to counter this problem by using SVM learning on a set of positive tweets and mine more accurate tweets.

Our system does not restrict a user from tweeting about the flu multiple times in a small time frame. This could increase the tweet count by a huge value and give away false heat map, scatter map and tweet count information. We plan to work on a foolproof solution for this and work around this issue.

Currently our system has a small dataset to work on for outbreak detection. If the system continues to collect data over all seasons in a year we could probably provide estimates over possible outbreaks in a city or for a group of closely located cities.

REFERENCES

[1] Google Flu Trends:
https://www.google.org/flutrends/us/#US

[2] Amazon web services EC2
http://aws.amazon.com/ec2/

[3] CDC weekly flu data

http://www.cdc.gov/flu/weekly/

[4] WebMD Symptom Checker
http://symptoms.webmd.com/cold-and-flu-map-tool/default.htm

[5] K means clustering – sklearn kit

http://scikit-learn.org/stable/modules/clustering.html