Twitter EmotiMap

Members:

*Matthew Gross, Andrew Mahan, Brian McWilliams, Dillon Fancher, Varsha Devnani, & Max Trotter*

# Abstract

Today, we find that a large portion of human interaction is centered in social computing. As society moves away from pen and paper and toward the Internet to express thought, the field of computer science finds itself confronted with an assortment of social and psychological data and trends at its disposal. In the following project, we will attempt to harness the enormous stream of Twitter statuses, known as “tweets,” in order to analyze the mood demonstrated by a population. Furthermore, we will group these trends in accordance with their geographical attributes when possible so that a regional mood analysis may also be shown.

# Introduction

In the ensuing project, a variety of APIs will be utilized in order to collect, filter, and display Twitter data. First and foremost, we must address the Twitter fire hose, which shall be used in order to collect live Tweets. Initially, tweets will be filtered so that only those with tweet content in English and geolocation data readily available in their metadata will be collected for analysis. After this is solidified, we may expand our geolocation criteria so that, if the geolocation data is not in the tweet’s metadata, the user’s profile will be parsed, if possible, and their profile location will be used as a proxy. Additionally, as we expand our dictionary database, we may also incorporate languages other than English for which to propagate key emotional words.

Once tweets are collected, we will analyze the textual content of each tweet for a variety of key emotional words. A data set of “emotion words” will be drawn up for which to be used in an analysis against each tweet’s content. These emotion words will be assigned a numerical quantity, which scales to a certain emotion. All emotional words hashed from a tweet will then be averaged together to achieve a composite emotional score for that tweet. The scale to be used as well as mathematical processes associated with this data matching will be further expounded upon in the *Methodology* section of this report.

Next, based upon the collected geolocation data for each tweet, the data will be displayed on a map, using Google Maps API, in order to demonstrate regional mood trends. A color scheme will be assigned to moods so that the maps may provide a more visual representation of mood rather than boring textual data accumulations. Further analysis in regards to trends over time and local tendencies will also be considered before a final feature presentation. All analysis and visuals will be wrapped within a website built upon the Bootstrap framework.

# Methodology

## Tweet Mining & Geolocation

### Summary

Tweets will be collected on a live basis, via a fire hose, so as to capture the most recent and relevant data. This collection will be performed by script running on remote servers that then store the collected data to other nodes in the network. This aggregated data will then be processed through another set of scripts in order to assign each tweet with a composite “emotional score.” This data will then be fed to the Google Maps API. A producer-consumer method will be needed in order to efficiently and safely manage the concurrent handling of mining and analysis.

### Twitter API & Tweet Structure

Tweets will be collected on a live basis, via a fire hose, so as to capture the most recent and relevant data. In addition to basic sentence structure, hashtags will also be parsed for discernable meaning. Twitter’s RESTful API will be used in order to parse streaming tweets into JSON objects to be queried into storage. Tweets’ metadata will be used to filter out useless tweets and analyze those tweets that fit the parsing criteria specified in the following section.

### Parsing Parameters & Methodology

Tweets will be filtered by the following attributes:

1. Content
2. Coordinates
3. Language

Initial effort will be geared toward analyzing English tweets with populated geolocation metadata. This will be later expanded to profile location mining and possibly additional language support.

Parsing will be performing using Google’s Twitter Python library. A very simple, example filter can be viewed in Figure 1 of the Appendix.

### Amazon Web Services (AWS) – Storage & Computation

Amazon Web Services (AWS) harness the robustness of Amazon’s vast server enterprises and make them available to the public. Not only do the servers allow for large scale storage, but they also provide an easy means for cloud computing. Both services will be essential to the collection and analysis of Twitter data.

Seeing as the data collected will be consistent in structure, and fairly simple, it seems suitable to select relational databases as a mean for storing tweet data such as content, coordinates, and location – along with other features. MySQL appears to be the most reasonable management system. Data may need to be sharded across multiple servers as it grows; however, this spread should still maintain its consistency, as little to no replication will be needed – (the value of a single tweet does not warrant a replication, thus taking the spot of another tweet that is just as valuable). Additionally, an expiration policy will be considered for tweets so that antiquated data is unnecessarily stored, given the limited resources available to us at this time. The deletion of expired data would indeed have an adverse effect on the availability of the data to an audience as it grows. At this juncture, however, we find it an acceptable compromise to perform updates at times of low use. As an audience and the volume of data grows, it may be reasonable to reevaluate the validity of a relational database structure; however, it is most practical at this point in time to begin with a simple and efficient, short-term solution.

In addition to storage, AWS cloud services will also be needed for the mining computations. Python scripts will be run from multiple nodes in order to collect live data. With this concurrent operation comes the risk of duplication. **<HOW WILL WE ADDRESS THIS?>** The data will then be pushed to the appropriate databases where the tweets may then be analyzed by other nodes which will then enable the rendering of the results onto the front end of our application.

## Mood Rating & Associated Mathematical Models

The main incentive behind collecting live tweets is to analyze the associated emotion that is being demonstrated on a large scale. A large list of words will be **<WHEN/IF WE FIND AN EMOTION-TO-WORDS DB, LET’S MENTION IT HERE>** generated so in order to be hashed against incoming tweets in order to gauge the emotional trend of each one. Once a mood rating has been determined, a certain color will be assigned to a pushpin and that pin will be placed at the location from which the tweet was made. One challenge to be acknowledged is the abundant use of inside humor and sarcasm in tweets. Our system will analyze words objectively for mood, however it is entirely possible that a word may be misrepresented as a result – and thus further investigation will be made towards a possible solution to handling metaphorical and satirical writing.

A mathematical model will be drawn up in order to assign a numerical value to a tweet with regards to the emotions that its content displays. This algorithm will be performed on all collected tweets and then assign a permanent value to each tweet. **<DYLAN MAY WANT TO FURTHER DESCRIBE THE MATHEMATICAL MODELS WE WILL USE>**

## Google Maps & Geolocation Visualizations

The latitude and longitude of all collected tweets will be parsed in order to display the tweet on a map once its emotional scale has been determined. If a tweet does not have geolocation data, the location of the user’s profile, if available, will be retrieved and used, instead. Additionally, tweets will be stored for a certain amount of time in order to enable the user to view a historical map of data. Thus, along with a current rendition of the public’s mood, a user may also access past emotional trends – enabling a multitude of possible analytics options with respect to change in mood over time.

All geographical data will be displayed on a browser client via the Google Maps Browser API. This will allow for in-frame rendition and a smoother, cleaner experience for the user, as they will be presented with a friendly UI, rather than raw data and analytics. Google Maps will be implemented using v3 in Javascript, but its use shall be restricted solely to the domain of the front end host so that the data may not be abused by a third party.

## User Interface

<LET’S FIGURE OUT WHICH TECHNOLOGY WE WANT TO USE TO STREAM SERVER DATA TO THE STATIC WEBPAGE MAP API>

# Results

Results

# Analysis

Analysis

# Conclusion

Conclusion

# References

Google Python Twitter Library - <https://code.google.com/p/python-twitter/>

Twitter API (Tweets) - <https://dev.twitter.com/docs/platform-objects/tweets>

Amazon Web Services - <https://aws.amazon.com/>

Google Maps Browser API - <https://developers.google.com/maps/documentation/javascript/reference>

Bootstrap - <http://getbootstrap.com/>

# Appendix

## Figure 1

import twitter

import json

# API Credentials

api = twitter.Api(consumer\_key='None',consumer\_secret='None', access\_token\_key='None', access\_token\_secret='None',cache=None)

statuses = api.GetStreamSample()

# Iterate through all geolocated tweets

for obj in statuses:

if 'text' in obj and 'coordinates' in obj and 'lang' in obj:

if obj['text'] != None and obj['coordinates'] != None and obj['lang'] == "en":

coordinates = obj['coordinates']['coordinates']

print "Tweet: %s\nLocation: (%f, %f)" % (obj['text'], coordinates[0], coordinates[1])

# If the place is named, list it

if 'place' in obj and obj['place'] != None:

print "Place: %s" % obj['place']['full\_name']

continue