# Emotion Analysis on Twitter

Contributors:

ARTIMIZIA DIAS

NIDHI KARGATHRA

## CIS 600: Principles of Social Media and Data Mining

December 14, 2018

Fall 2018

### abstract

This document contains information pertaining to Emotion Analysis on Twitter application submitted as the final project for CIS 600 - Principles of Social Media and Data Mining. All code, presentation slides and documentation are created by Nidhi Kargathra and Artimizia Dias and submitted under their names as the members of the project team. The content contained within this document include the following:

1. Application Overview
2. Runtime Instructions
3. Technical Components
4. User Interface

Contents

[Chapter 1: APPLICATION OVERVIEW 4](#_Toc532558123)

[1.1 Overview 4](#_Toc532558124)

[1.2 Limitations 5](#_Toc532558125)

[1.2.1 Bokeh Server 5](#_Toc532558126)

[1.2.2 Python-Twitter API 5](#_Toc532558127)

[1.2.3 Google Geocoding API 5](#_Toc532558128)

[Chapter 2: RUNTIME INSTRUCTIONS 6](#_Toc532558129)

[2.1 Configuration 6](#_Toc532558130)

[2.1.1 API Accounts 6](#_Toc532558131)

[2.1.2 Authentication File 6](#_Toc532558132)

[2.1.3 Supporting Data 6](#_Toc532558133)

[2.2 Running the Code 6](#_Toc532558134)

[Chapter 3: TECHNICAL COMPONENTS 7](#_Toc532558135)

[3.1 Understanding the Dataset 7](#_Toc532558136)

[3.2 Data Preprocessing 8](#_Toc532558137)

[3.3 Classification 8](#_Toc532558138)

[3.3.1 Term Frequency-Inverse Document Frequency 8](#_Toc532558139)

[3.3.2 Problem of Imbalanced Data 8](#_Toc532558140)

[3.3.3 Selection of Classifier 9](#_Toc532558141)

[3.4 Visualizations 10](#_Toc532558142)

[Chapter 4: USER INTERFACE 10](#_Toc532558143)

[4.1 User Input 10](#_Toc532558144)

[4.1.1 Location Input 10](#_Toc532558145)

[4.1.2 emotion Filter 10](#_Toc532558146)

[4.2 Visualizations 11](#_Toc532558147)

[4.2.1 Google Map 11](#_Toc532558148)

[4.2.2 Pie chart 11](#_Toc532558149)

## Chapter 1

Application Overview

# Overview

This application will collect data from twitter and run emotion analysis on the tweets. Each tweet will be rated on its primary emotional content out of 8 emotional choices, based on Plutchik’s wheel of emotions. Based on the location and the emotion/s selected by the user, the application will fetch tweets in real-time. Interactive charts will be used to summarize the emotion of the tweets. The output will also be displayed in the form of colour-coded plots on a geographical map. The classification is implemented using Random Forest Classifier on ISEAR ([The International Survey on Emotion Antecedents and Reactions](https://www.affective-sciences.org/research/materials-and-online-research/research-material/)) dataset.

The classes of emotions in the dataset are anger, disgust, fear, guilt, joy, neutral, sadness and shame.

Filtering of tweets will be made available to the user based on two parameters:

1. Location: User can select the geographic location and tweets geo-tagged with that location will be fetched.
2. Emotion: User will be able to select one or more emotions to filter the tweets.

Upon submitting a request, tweets at the specified location are harvested from twitter, classified into one of the 8 emotions, and returned visually to the user on the fly. The data returned includes a geographic plotting of tweets color-coded by emotion and a pie chart reflecting the aggregate mood of the community.

Further non-technical and technical details can be found in the next two chapters.

# Limitations

### 1.2.1 Bokeh Server

Due to limitations with Bokeh Server, the same geographic location cannot be searched multiple times without restarting the Jupyter Kernel. This is a result of the Bokeh's functionality to load specific tiles from a geographic tile provider. Multiple searches can be done but different locations must be used.

### 1.2.2 python-Twitter API

The Python-Twitter API is limited according to the rate limits specified on the Twitter API website. As of the writing of this document the limit for retrieving tweets is 450 requests per 15-minute window. Given that this application will issue up to 50 batched requests to the Twitter API for a given search the application is theoretically limited to 9 searches every 15 minutes. If the limit is hit, the application will return empty results in the format below.

### 1.2.3 Google Geocoding API

The Google geocoding API is also rate-limited but given that this application only issues one API request for each search, the user is much more likely to be constrained by the limits of the Twitter API. A user would have to submit 2500 requests in a given 24-hour period to hit Google's rate limit.

## Chapter 2

Runtime instructions

# 2.1 Configuration

### 2.1.1 API Accounts

In order to use this application, a user must have registered an account to use the Twitter API and the Google Geocoding API. Registering with these services will provide the user with the authentication keys and tokens required to make the API calls in the code.

### 2.1.2 Authentication file

Once accounts are setup with the two API services, the keys and tokens will need to be stored in JSON format in a file called OAuth Keys.json. This file should be placed in the same directory as the Jupyter Notebook. Alternatively, the parameter CREDFILE in the code can be changed to reference a different location and filename for the JSON credential file. A sample file containing dummy keys currently exists in the application zip file.

The file must contain the following key value pairs all at the initial level in the JSON file.

|  |  |
| --- | --- |
| **Key** | **Value** |
| Token | Twitter API token |
| SToken | Twitter API secret token |
| Key | Twitter API key |
| SKey | Twitter API secret key |
| GoogleKey | Google API key |

### 2.1.3 Supporting data

Lastly, the data file must be included with the Jupyter Notebook. This file, included in the submission, contains data for the ISEAR dataset in CSV format. The file is called ISEAR.csv.

# 2.2 running the code

After all of the configuration steps have been completed, the Jupyter Notebook is ready to be run. In order to start the Application, all modules in the Jupyter Notebook must be run in order. The final module starts the Bokeh application and will then be used to interact with the application.

## chapter 3

technical components

The complete process is as outlined below, followed by the detailed technical explanation for each component.

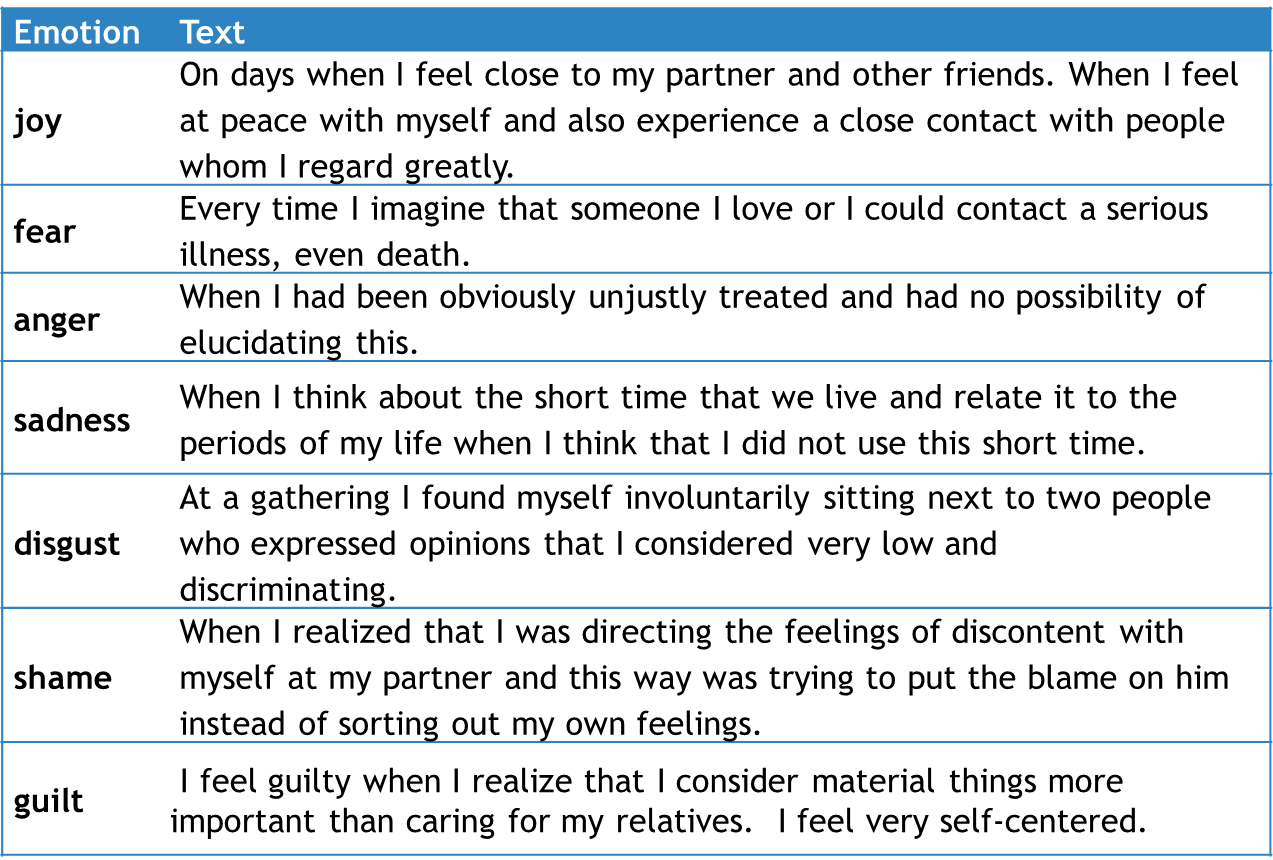
1. Data collection: The ISEAR dataset is split into test and train data after pre-processing.
2. Training the model: Using four different classifiers, we train and test the models. The final model is selected based on the performance of each model on the test data. The finalized models are saved as pickle files.
3. Harvesting twitter data: Using the user input to find tweets from a certain region.
4. Classifying twitter data: After scraping the Tweets from the data sources, the system will perform emotion detection by running the selected classification model and Natural Language Toolkit (NLTK).
5. Compare and Visualize Results: Depict different categories of emotions in interactive charts.

# 3.1 understanding the DAtaset

The dataset is available on [International Survey On Emotion Antecedents And Reactions (ISEAR)](https://www.affective-sciences.org/research/materials-and-online-research/research-material/)

Over a period of many years during the 1990s, a large group of psychologists all over the world collected data in the ISEAR project, directed by Klaus R. Scherer and Harald Wallbott. Student respondents, both psychologists and non-psychologists, were asked to report situations in which they had experienced all of 7 major emotions (joy, fear, anger, sadness, disgust, shame, and guilt). In each case, the questions covered the way they had appraised the situation and how they reacted. The final data set thus contained reports on seven emotions each by close to 3000 respondents in 37 countries on all 5 continents.

Preview of the dataset after cleaning up unwanted columns:



# 3.2 data preprocessing

Before the ISEAR dataset can be used, it needs to be cleaned. Pre-processing of data is an important step in the classification process. Bad quality of data can reduce the classification accuracy. Following steps are involved in this process:

1. Removing punctuations – punctuations are removed as is it not possible to extract much meaning out of it.
2. Removing hashtags and @ mentions – removed because it does not contribute to overall sentiment
3. Removing stopwords – stopwords are frequently occurring words like the, is, if, etc which do not contribute to the semantic meaning
4. Stemming – words are reduced to their roots to simplify the analysis

Data is harvested on-the-go with help of the Python-Twitter API. It is limited according to the rate limits specified on the Twitter API website. As of the writing of this document the limit for retrieving tweets is 450 requests per 15-minute window. Given that this application will issue up to 50 batched requests to the Twitter API for a given search the application is theoretically limited to 9 searches every 15 minutes. The data is stored into a Pandas data frame. The same pre-processing steps are performed on this data.

# 3.3 classification

### 3.3.1 Term frequency-inverse document frequency

Text files are actually series of words (ordered). In order to run machine learning algorithms, we need to convert the text files into numerical feature vectors. We will be using bag of words model for our example. Briefly, we segment each text file into words (for English splitting by space), and count # of times each word occurs in each document and finally assign each word an integer id. Each unique word in our dictionary will correspond to a feature (descriptive feature).

TF: Just counting the number of words in each document has 1 issue: it will give more weightage to longer documents than shorter documents. To avoid this, we can use frequency (TF - Term Frequencies) i.e. #count(word) / #Total words, in each document.

TF-IDF: Finally, we can even reduce the weightage of more common words like (the, is, an etc.) which occurs in all document. This is called as TF-IDF i.e. Term Frequency times inverse document frequency.

### 3.3.2 problem of imbalanced data

There are 7 labels in the training dataset. A binary classifier was constructed for each label. This approach was adopted because a multi-label classifier did not perform well with this data. So instead of one multi-label classifier, we are employing 7 one-vs-rest classifiers. This led to imbalanced data, which is one of the worst problems one can encounter with the data while building a classification system. It can cause the classifier to be biased towards the majority class.

A sophisticated solution to this problem is oversampling. The under-represented data is randomly oversampled using SMOTE. It stands for Synthetic Minority Oversampling Technique. As the name suggests, it creates synthetic samples of the under-represented class in an intelligent manner. When SMOTE creates a new synthetic data, it will choose one data to copy, and look at its k nearest neighbours. Then, on feature space, it will create random values in feature space that is between the original sample and its neighbours.

We implemented SMOTE technique and balanced out our data for maximum accuracy.

### 3.3.3 selection of classifier

We split the data into test and train data using random sampling of 20:80 ratio. The data was trained and tested on 4 classification algorithms. Detailed analysis is given below:

|  |  |
| --- | --- |
| **Classification algorithm** | **Accuracy %** |
| SVM | 73.75 |
| Naïve Bayes | 88 |
| Logistic Regression | 89.32 |
| Random Forest (100 estimators) | 95.31 |

It is clear from the above table that Random Forest Classifier is best suited for our data. Further performance metrics for each of the 7 individual classifiers is given below.

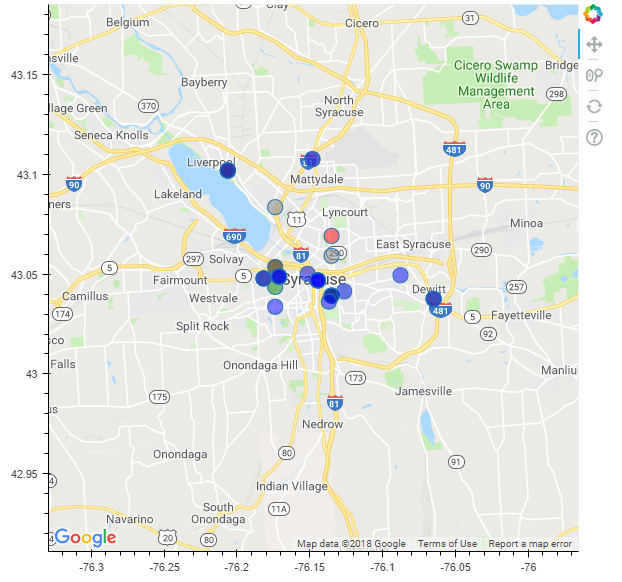
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| joy | 0.95 | 0.97 | 0.96 | 2167 |
| nojoy | 0.97 | 0.95 | 0.96 | 2163 |
| weighted avg | 0.96 | 0.96 | 0.96 | 4330 |
|  |  |  |  |  |
|  | **precision** | **recall** | **f1-score** | **support** |
| fear | 0.94 | 0.96 | 0.95 | 2200 |
| nofear | 0.96 | 0.94 | 0.95 | 2129 |
| weighted avg | 0.95 | 0.95 | 0.95 | 4329 |
|  |  |  |  |  |
|  | **precision** | **recall** | **f1-score** | **support** |
| anger | 0.96 | 0.94 | 0.95 | 2172 |
| noanger | 0.94 | 0.96 | 0.95 | 2157 |
| weighted avg | 0.95 | 0.95 | 0.95 | 4329 |
|  |  |  |  |  |
|  | **precision** | **recall** | **f1-score** | **support** |
| sadness | 0.95 | 0.94 | 0.94 | 2140 |
| nosadness | 0.94 | 0.95 | 0.95 | 2189 |
| weighted avg | 0.95 | 0.95 | 0.95 | 4329 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| disgust | 0.92 | 0.96 | 0.94 | 2177 |
| nodisgust | 0.96 | 0.91 | 0.93 | 2152 |
| weighted avg | 0.94 | 0.94 | 0.94 | 4330 |
|  |  |  |  |  |
|  | **precision** | **recall** | **f1-score** | **support** |
| shame | 0.96 | 0.9 | 0.93 | 2176 |
| noshame | 0.91 | 0.97 | 0.94 | 2153 |
| weighted avg | 0.93 | 0.93 | 0.93 | 4329 |
|  |  |  |  |  |
|  | **precision** | **recall** | **f1-score** | **support** |
| guilt | 0.95 | 0.95 | 0.95 | 2154 |
| noguilt | 0.95 | 0.95 | 0.95 | 2177 |
| weighted avg | 0.95 | 0.95 | 0.95 | 4331 |

Note that we built classifiers for the 7 labels only, and left out the 8th label “NEUTRAL”. Neutral is the absence of all of the above emotions. Thus, if these classifiers do not predict a class as positive with enough probability (>50 %) on a sample input text, i.e. all emotions are negative, we label the input sample as having a neutral emotion.

The trained models are saved as pickle files for all later uses. This saves a lot of training time for future use.

# 3.4 Visualizations



All of the visualizations are plotted using Bokeh. This includes the geographic mapping, tables and charts. Each plot has an associated helper function that is called to manage the data and configure the display of the graph.

## chapter 4

USer interface

The user interface is comprised of two sections, the user input menu on the left and the results that the application displays visually on the right. Initially, since there are no results to display, the right side of the application will just display a message prompting user to supply the input for their query into the men on the left. The initial display is shown below.

When a user submits a request, the display will cycle through a series of processing messages showing the user what is going on behind the scenes before finally displaying the results. An example of the final display is shown below.

# 4.1 User input

The left-hand menu allows users to supply input to the application. This includes the geographic location around which the tweets will be gathered, as well as any emotion user wants to filter. The menu is shown below.

### 4.1.1 Location input

The top section of the menu allows users to specify where geographically they want the application to collect tweets from. The Search Location field is where users enter the location in plaintext. This can be as general as a city or region, or as specific as a neighborhood or street address. This location will be geocoded using the Google Geocoding API to get specific Latitude and Longitude for that location.

### 4.1.2 emotion filter

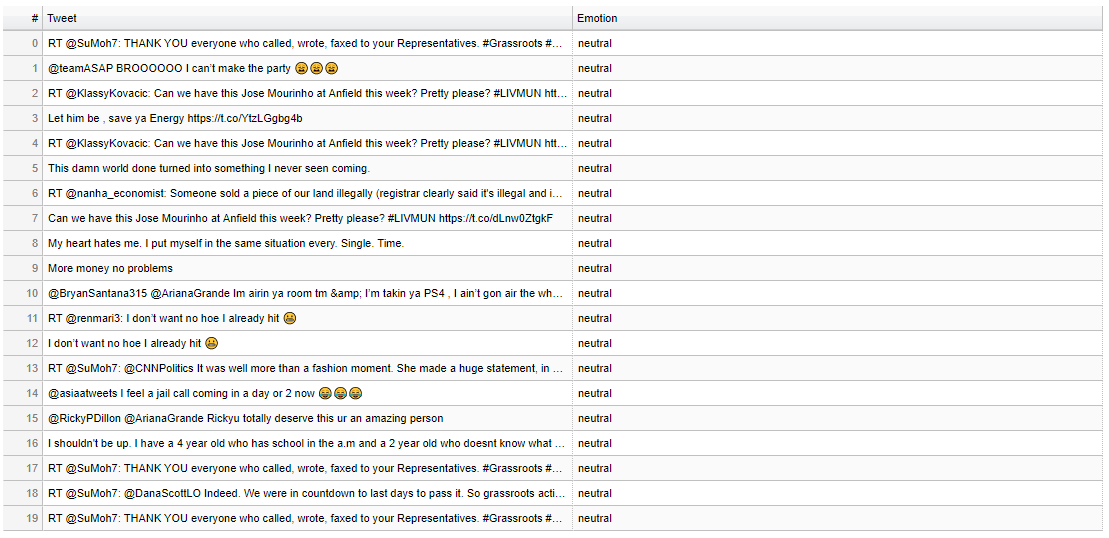
User has an option to display or to not display tweets belonging to certain emotions by selecting or de-selecting the checkboxes corresponding to each emotion.

# 4.2 visualizations

The results of a user request are displayed graphically in two tabs. The first panel maps the location of geocoded tweets, the second displays a pie chart depicting the overall feelings across the location. The panels can be navigated to by clicking on the tabs at the top of the screen.

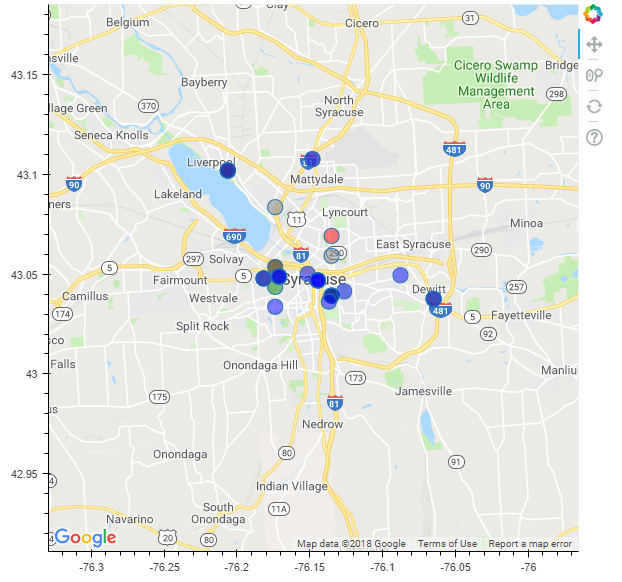
### 4.2.1 Emotion tweet Table:

A table with some tweets and their classified emotion type .This emotion type is either 'joy','fear','anger','sadness','disgust','shame','guilt','neutral'.



### 4.2.1 google map

The first panel plots tweets that have geocoding enabled on a map. Each point is color-coded with the corresponding color for each emotion.



### 4.2.2 pie chart

The second panel displays a pie chart showing the distribution of each emotion and precise counts of each emotion for the tweets extracted at the selected location, which is visible on hover over each section.

