1 Overview

For this assignment you are going to work with one of your classmates and write a Python program which will calculate the sentiments in real twitter data for a given keyword. I gathered twitter data on February 19 and February 25 for this analysis. Your program will read in each tweet in both files. It will associate the tweet with its corresponding state. Then it will evaluate the "sentiment" of each tweet using a primitive technique. Finally it will produce a map that displays the mood for each state for a given term. Note that I did not gather tweets from Alaska or Hawaii.

DISCLAIMER: This projet has you working with real data from Twitter. Thus, there could be foul language, inappropriate comments, links to malicious web sites or lewd pictures. In fact, I shouldn't even say "could be" in the previous sentence, I would bet my hard earned money that there are examples of all of those things! I think it is extremely valuable from an educational standpoint to be using a real data set for this project but that comes with a certain downside that, as adults, I hope that we can all put up with.

NOTE: Twitter does not allow the publication of collected tweets, thus, you are not allowed to post the data files I provided you to a public forum. You may display the maps you create and discuss the analysis publicly but you are forbidden from posting the raw data.

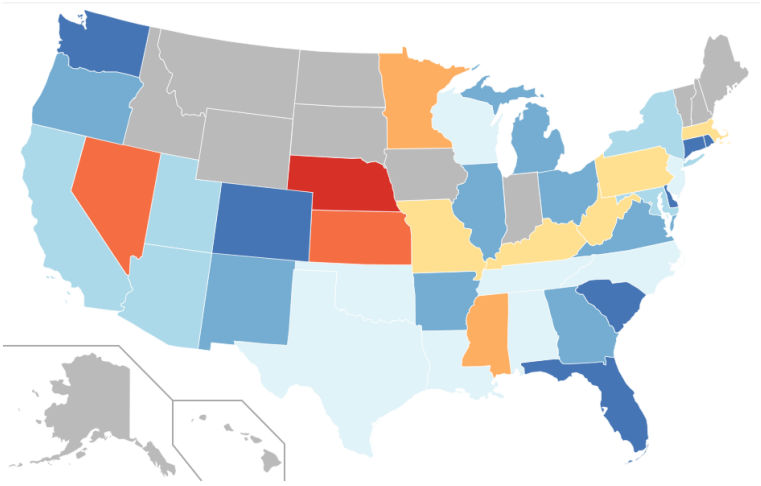

Figure 1: In the map above I searched for Twitter activity on the word "nba" using both days of twitter data. The National Basketball Association held its all-star festivities on the weekend of February 18,19. The red signifies positive tweets regarding the nba, fading to orange and yellow to signify less positive tweets. The gray states had no twitter references to nba. The dark blue signify negative tweets regarding nba, fading to light blue to signify less negative tweets.

2 Files I've Provided

1. Data Files

   Do not alter any of these files at all. The data files are in zip format on Canvas. Specifically, `usaTweetsFeb19.zip` contains `usaTweetsFeb19.json`, `usaTweetsFeb25.zip` contains `usaTweetsFeb25.json`, and the other data files mentioned below are in `hw4OtherDataFiles.zip`

   - `sandwich.json` (JSON stands for JavaScript Object Notation)

     This file contains 11 tweets about sandwiches. I actually just took the first 11 posts from the February 19 file below and changed the tweets to be about sandwiches. This will serve as a good test file as you're developing your code so you don't have to wait for the much larger files mentioned below to load.

   - `usaTweetsFeb19.json` and `usaTweetsFeb25.json`

     These files contain about a half hour's worth of tweets collected on two different days. There are about $50,000$ tweets in each of those files. These files are not nearly large enough to do a statistically significant analysis but they'll suffice for this project.

   - `sentimentsSmall.csv` (csv stands for comma separated values, you can open it with any text editor or Excel).

     This file contains a short list of 8 key words and phrases that have a score in the range -1 to 1 where -1 represents very negative sentiment and 1 represents very positive sentiment. Some of the 8 words / phrases chosen for this file appear in `sandwich.json` so it is also handy while you're developing your code so you can make sure everything is working.

   - `sentimentsFull.csv` This file contains more than 22 thousand words and phrases (all in lower case) with sentiment scores in the range -1 to 1. The scores calculated in this file were developed through a detailed analysis of English words and phrases. Note the scores in this file for some words may differ from the scores in the smaller sentiments file because I altered the scores in the file above for testing purposes.

2. Python Files and Supporting Files for Drawing the Map

   - `simplemapplot.tar.gz` This file contains some helper functions to draw the map at the end. Copy this file to the directory you want to work in (it's best to make a special hw4 subdirectory for this project since there are so many supporting files).
     - If you're working in Linux you type `tar xvf simplemapplot.tar.gz` and it will extract files and create a subdirectory that will contain all the helper files for drawing the map.
     - If you're working in Windows in the labs then you can right click on the file on go to `7zip` and click `Extract Here` which will create `simplemapplot.tar` then right click on that file and do the same thing and that will create a subdirectory called `simplemapplot` with all the necessary files in it.
   - `hw4_skeleton.py` This is a Python file with some helper functions and some other definitions for you. You should add all of your code to this file and then rename it when you turn it in according to the directions below. Let's go through this file in more detail so you know what's what.
     - Do not touch the import statements at the top.
     - The variable `SENTIMENTCOLORS` is a list that contains strings representing a gradient of colors from red to blue with the middle one being gray. Each of these strings is

actually a base sixteen RGB color. Do not alter the colors. I used Color Brewer (`colorbrewer2.org`) to pick the colors.

- The variable `stateAbbrevList` is a list that contains the abbreviations of the 50 states. I use this when I read a tweet to make sure that the tweet was tagged with a valid state.
- `readTweetFile` takes an empty list and a string which is a file name. It opens the file specified and then creates a list where each item of the list is a tweet read from the file that meets my minimum qualifications (i.e. the tweet was located in the continental USA and it had a state tag on it). It uses `parseTweet` as a helper function.

I also defined some variables in the main program for you to use.

- `tweetList` is a list. Once the file has been read, each list item is itself a list of 7 strings indexed as follows: [0] date, [1] identification number, [2] text of the tweet, [3] number of followers of the person posting, [4] number of friends of the person posting, [5] two letter country code, [6] city, state. Note that this last item is literally a city name followed by a comma, followed by a space, followed by a two letter state code (e.g. "Salt Lake City, UT", "Seattle, WA", etc.)
- `sentimentDict` will be a dictionary. key: string, value: sentiment value from -1 to 1. You have to populate this file from the sentiment files.
- `stateCountDict` will be a dictionary. key: two letter state code, value: count of the number of tweets from each state that your keyword appears. I included a loop in the main program to initialize all values to 0.
- `stateSentimentScoreDict` will be a dictionary. key: two letter state code, value: sentiment score for that state. Initially, this will hold a raw sentiment score for each state, then it will hold the average sentiment score for each state, then finally it will hold an index into the SENTIMENTCOLORS list. I wrote a loop in the main program to initialize all values to 0.

3 How to Get Started

We're going to assign you a partner in lab this week. The first thing you should do is review the sample code from Monday's class on dictionaries together as a team.

Then review this document section by section as a team to make sure you're on the same page. Help out your partner if they don't understand something and similarly, ask questions of your partner if you don't understand something. Be diligent here. There is a lot to piece together for this assignment and the better view of the big picture you have the easier this will be.

The code I'm giving you reads in the tweet file (the small one about sandwiches) and stores several fields into tweetList. You can start without your partner by just printing that list out so you're familiar with how it is structured. Look at the class examples and the book if you need to.

You have to write a function to read in the sentiment file. You can start with that once you have a partner. You can print out that list to make sure it was read in correctly.

At this point you can prompt the user for a word of interest (for initial testing use the word "sandwich").

Then write a loop in the main program to examine the items (i.e. tweets with metadata) from `tweetList`. For each item see if the word of interest is in the current tweet you're examining (the

`in` operator is your friend!). If the word is in the tweet, then you'll have to extract the state of the tweet and add one to that state's counter in `stateCountDict`.

If you run this loop on the tweets in `sandwich.json` you should get the following counts (all state counts not shown below are zero):

```
AZ 1
CA 1
CO 1
MS 1
NJ 2
NY 2
TX 3
```

Your final program should not print these out but it's helpful to do so here to check on this small file to make sure it's working.

Next you'll want to determine the sentiment of the tweet and add the appropriate sentiment score to the current value in `stateSentimentScoreDict`.

If you print out the `stateSentimentScoreDict` here you should see raw sentiment scores for each state. Again, working on the small file of sandwich tweets you should get the following (all state scores not shown below are zero):

```
AZ 0.375
CA -0.125
CO -1.5
MS -0.125
NJ -2.25
NY 2.25
TX 0.625
```

Next you should average the sentiment score for each state. This is as simple as taking the raw score and dividing by the count. At this point you should get the following (again, all state scores not shown are zero):

```
AZ 0.375
CA -0.125
CO -1.5
MS -0.125
NJ -1.125
NY 1.125
TX 0.20833333333333334
```

4 Home Stretch

You've done the hardest part. All that is left to do is assign colors to each state based on their sentiment. Call this function `assignSentimentColors`. It should take a dictionary of state:score pairs (state is the two letter abbreviation and score is a float containing that state's average sentiment score for the given word).

The idea of this function is that you'll first determine the minimum sentiment score and the maximum sentiment score. Then assign colors to the states based on how close they are to the min and max. For negative scores, we'll use color 0 for those that are less than 60% of the minimum score, color 1 for those that are less than 30% of the minimum score, color 2 for those that are less than 15% of the minimum score and color three for the rest of the negative scores. For states with a sentiment of zero we'll use color 4. Then we'll do a similar thing for positive scores. Color 8 will be for those that are greater than 60% of the maximum score, color 7 for those greater than 30% of the maximum, color 6 for those greater than 15% of the maximum and color 5 for the rest of the positive scores. This is just one big honking if statement!

Then you should be all ready to call the `make_us_state_map` function at the bottom of the main program. I wrote the function call for you but left it commented out in the skeleton program. It will create a file in your directory and you can open that with a web browser like Firefox or Chrome and you'll see your map.

Using the small files (`sandwich.json` and `sentimentSmall.csv`) you should get the following map when you use "sandwich" as the keyword:

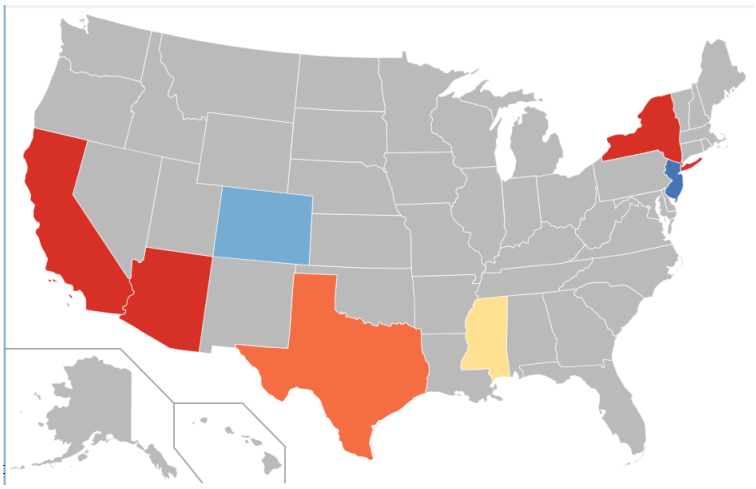

Figure 2: This map is for "sandwich" on the small data files

If this worked then it's time to unleash this puppy on the big data files! The picture on the first page is for nba on the combination of both of the big data files. Note that the way the function `readTweetFile` works is that it will append new tweets onto the end of the existing `tweetList` so you can just call that function twice, once for each file of tweets. You do not need to modify that function at all.

Some more pictures are included below.

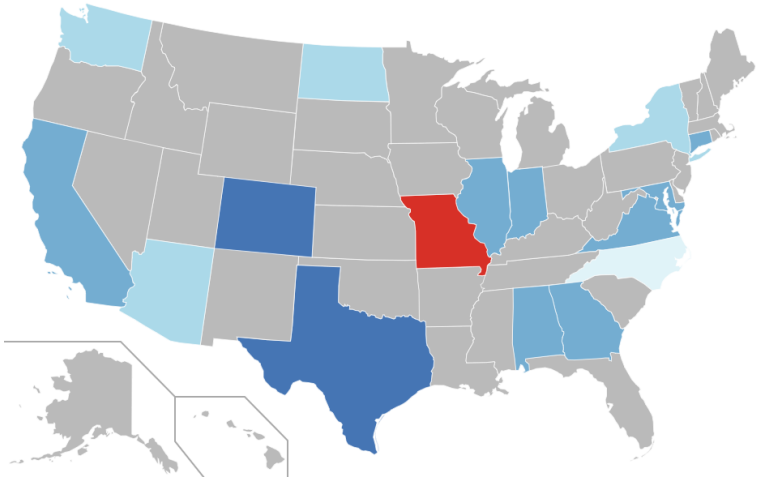

Figure 3: This is the map for "immigration". While this is showing that people around the country are tweeting negatively about immigration there were only dozens of such tweets in the time period collected. That's why I said above that the size of the data files are not suitable to make statistically relevant conclusions.

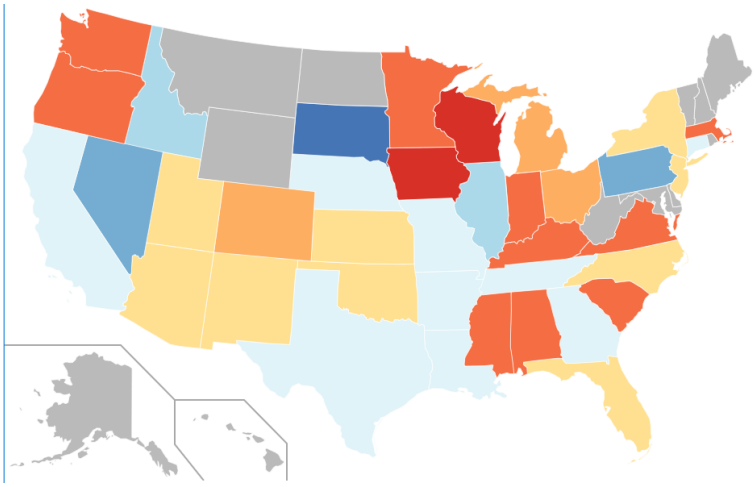

Figure 4: This is the map for "baseball". Spring training commenced in February for professional baseball teams and this map may suggest who is optimistic about their team.

5 A Note on Paired Programming

Pair programming is a software development technique where two programmers work together in front of one keyboard. One partner types code while the other is suggesting and/or reviewing every line of code as it is being typed. The person typing is called the driver. The person reviewing and/or suggesting code is called the navigator. The two programmers should switch roles frequently (such as every 20 minutes). For this to be a successful technique the team needs to start with a good program design so they are on the same page when it is finally time to start typing on the computer. **No designing or programming is to be done without both partners present!** You should individually read this assignment sheet at least a few times before starting and read it with your friends and talk about it too! The better big picture view you have going in the better partner you'll be.

Pair programming has been shown to increase productivity in industry (I once was on a paired team in industry where we finished a project in 10 days that we were allotted 30 days to complete! And we would have been allotted 60 days if only one of us was assigned it alone. Afterward, we both felt that it would have taken us each more than 60 days if we had not worked together) but it is actually being used in this class for other reasons. First, I am including this assignment to increase collaboration – too much of the introductory sequence in computer science is a "go it alone" endeavor which does not accurately reflect what goes on in upper division classes or in industry. The other reason is that it will be a good teaching tool. Inevitably, in each pairing the partners will have different styles and abilities (for instance, one person may be better at seeing the big picture while the other is better at finding detailed bugs or one person might like to code on paper first while the other likes to type it in and try it out). Because of that you will have to learn to adjust to another person's style and ideally you will meet each other half way when there are differences in approach. It's important that each person completely understands the program and so both parties need to be assertive. Be sure to explain your ideas carefully and ask questions when you are confused. Also it is crucial that you be patient! There is plenty of time allotted to complete this assignment as long as you proceed at a steady pace. Ask for help from me or your TA if you need it.

You will use pair programming for this assignment. You will be assigned a partner from your lab period and those pairings will be announced this week. You will spend lab time in the final two weeks working on this assignment as a team. **For this reason attendance at lab the last two weeks is mandatory!** Your last ten points of your lab grade are based on attendance in the final two weeks. Failure to show up for lab will also result in you individually losing significant points this homework assignment. You will need to contact me ASAP if you miss lab. You will also need to spend time outside of lab to work together on this assignment.

Let me stress again that **no designing or programming is to be done without both partners present!** If I determine this happens I will fail you for this assignment.

6 Grading

Name your program `lastname1_lastname2_hw4.py` and please put the alphabetically earlier name first in the list for your TA's sake. Only one copy has to be turned in for both partners.

- Style and Clarity (20 points)
  - Meaningful well-designed functions. At least 3, no more than 5 in addition to the ones I provided (8 points)
  - Clear docstrings on the functions you wrote (4 points)
  - Meaningful variable names (4 points)
  - Use of for loops, with good loop iterator variable names, to iterate through dictionaries and lists (4 points)
- Correctness (25 points)
  - Your program should properly output a map into the same folder where the code resides that is colored based on the sentiment of the tweets relating to the keyword given by the user. (15 points)
  - In addition to outputting the map you should also print out to the screen your final dictionary that contain sentiment scores and tweet counts that contain the given word. **These should be in sorted order by state**. Your TA will pick one of the words or phrases (baseball, nba, immigration) to test your program. Make sure it works on all of them! (10 points)

7 Last thoughts on the assignment

I hope that this is an empowering assignment. There are so many available libraries and modules out there that it makes accomplishing something quite complex like this possible even by an introductory student.

I'll be available for help on this assignment but you have to make a good faith effort to understand what is going on. I know this handout is long but that's because there is a lot going on. So read it again and again before you start to get a basic grasp of what you need to do.

Then of course once it's working you can try out whatever terms you want! It's kind of interesting as it is but there are a zillion possible ways you could improve this program. Talk to me about those if you're interested.