

Ling 388 Final Project: Twitter Sentiment Analysis

Summary

So, for this final project, I decided to program a Twitter Web Scraper that searches for a particular word that's included inside of a user's tweet. This is particularly useful if we would like to do sentiment analysis based on some variable X that we would like to study.

And for the problem that I decided to research into, I choose the word 'corona' and decided to sample tweets from different major cities in the United States. If we do sentiment analysis on those tweets over time, we can start to formulate an opinion on how the nature of how corona is affecting people's lives in different areas.

- Towards that end, I decided to grab 100 tweets from 6 major cities in the United States (Tucson, Los Angeles, Miami, New York, Seattle, Chicago). I then pass that data into Google's Natural Language API to grab sentiment data to compare and contrast fluctuations of emotions regarding COVID-19 in those cities.

Procedure

1st Step: First we need some way to actually grab those tweets from the cities we want to research. To do that, we first need an API key from Twitter so that we can access their database of tweets.

- To get the API key from twitter, you need to apply through their website here. Make sure you have a twitter account first before you apply.

<https://dev.twitter.com/apps/new>

- You'll need four critical pieces of information from this. The consumer_key, consumer_secret, access_token, and access_token_secret. Write those down somewhere since we'll need them to use in the next part.

2nd Step: Now that we have the keys, we need some way to physical grab the tweets themselves. There exists a lot of API's to serve that purpose on the web, but for this project, I used tweepy, an API that we can use to develop on Python.

- To install it, just do a pip install of tweepy on your command line. There might be other dependencies that I don't remember installing, but just pip install your way through them until the error messages disappear. At the end of all that, you will want to have a setup like what I have here.

Ryan Luu
5/06/2020

- If you want to find geolocation for a specific area, you can either use google maps or a website like this one: <https://www.latlong.net>
- I also decided to cap out MAX_TWEETS at a 100 (could be higher) since Twitter will block you if you keep spamming their servers to grab tweets with the basic developer key.
- Any other documentation on Tweepy can be read here: <http://docs.tweepy.org/en/latest/>

```
tweepy_test.py - /Users/HuRuuCorporations/git/ling388-twitter/tweepy_test.py (3.8.1)

import tweepy
import time
import datetime
import json
import passwords

# Twitter API keys, I stored them in a separate file for security reasons
consumer_key = passwords.consumer_key
consumer_secret = passwords.consumer_secret
access_token = passwords.access_token
access_token_secret = passwords.access_token_secret

# Setting up the tweepy_api based on twitter keys
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

# Constants used to modify what data is grabbed and stored
# currentdate grabs the current time that the script is ran. Used in helping sort when data is grabbed by
# appending that information to the filename.
phrase_to_search = 'Corona'
MAX_TWEETS = 100
currentDT = datetime.datetime.now()
currentdate = str(currentDT.month) + "-" + str(currentDT.day) + "-" + str(currentDT.year)

# Geocode location of cities we want to look at.
# Format is Latitude, Longitude, Radius in either mi or km. 2nd Tuple information is used simply for file naming
# Also since the twitter api is dumb, don't put spaces between geolocation string information (1st item in Tuple)
tucson_geocode = ("32.22174,-110.92648,50mi", 'Tucson')
los_angeles_geocode = ("34.052235,-118.243683,50mi", 'Los Angeles')
new_york_geocode = ("40.712776,-74.005974,50mi", 'New York')
miami_geocode = ("25.761681,-80.191788,50mi", 'Miami')
chicago_geocode = ("41.881832,-87.623177,50mi", 'Chicago')
seattle_geocode = ("47.608013,-122.335167,50mi", 'Seattle')

cities = (tucson_geocode, los_angeles_geocode, new_york_geocode, miami_geocode, chicago_geocode, seattle_geocode)
```

Now that the setup is all done, we write some code to get us the tweets we want. It's simple and straightforward. It loops through the cities we want to look into (declared earlier on previous picture), grabs 100 tweets in a 50 mile radius of that geolocation, then outputs it to a JSON file for sentiment analysis later.

Here is the relevant code that does that.

Ryan Luu
5/06/2020

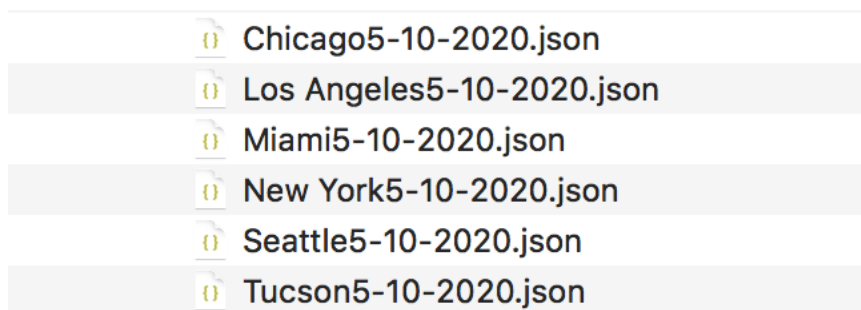
```
# Calls the API to search
try:
    for city in cities:
        tweet_string = ""
        for tweet in api.search(q=phrase_to_search, count=MAX_TWEETS, geocode=city[0], tweet_mode='extended'):
            tweet_string += str(tweet.full_text)

        # Format the json string for the google natural language api to process
        data = {
            "document":{
                "type":"PLAIN_TEXT",
                "content": tweet_string
            },
            "encodingType": "UTF8"
        }

        # Every filename will be different depending on city
        filename = str(city[1]) + currentdate + ".json"
        # Writes out the tweet list to a json file with the relevant city and the current date grabbed
        with open(filename, 'w') as filehandle:
            json.dump(data, filehandle, ensure_ascii=False)

except BaseException as e:
    print("Failed with exception: " + str(e))
    time.sleep(3)
```

If done correctly, these six files should appear in the same directory as your program.



Now we need a middle man to do our sentiment analysis for us. For that, I decided to use Google's Cloud's Natural Language API.

- There's a lot of hassle with getting this setup since it's a IaaS (Infrastructure as a Service). That means you normally have to pay to use their services. I'm a bit lucky on that end since I had to setup an account through my cloud computing course already, but don't fear since they offer apparently 300\$ worth of free credit for use in the first month if you just want to try it out.

- Then for the actually getting the Natural Language API to work with you, there are even more steps and hops to go through. I found this article extremely useful towards that end.

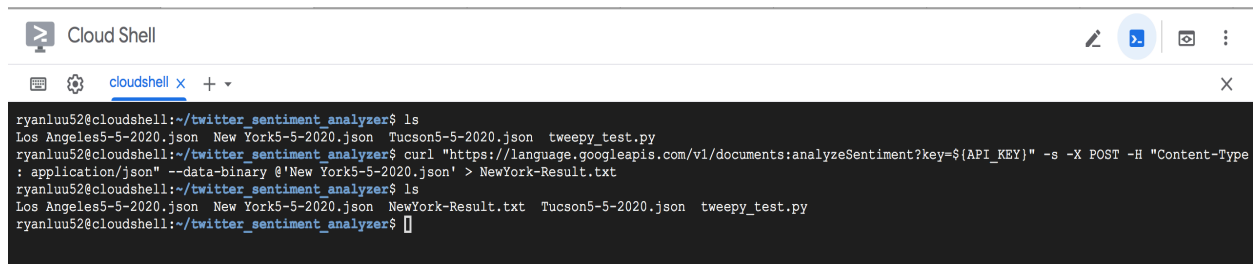
Ryan Luu
5/06/2020

<https://medium.com/google-cloud/sentiment-analysis-using-google-cloud-machine-learning-552be9b9c39b>

NOTE* If you wrote the code I mentioned above locally on your machine you need some way to get your files onto this Google Cloud platform. I used GitHub to migrate my code and files, but there exist alternatives such as doing an scp command on the files to the ip address of the instance or simply even just developing the python script itself within the cloud shell using your favorite text editor (Nano, Vim, Emacs, etc.). If you don't like those text editors, there also exists plugin on VS code that allow you to develop on google cloud as well. Either way, just find some way to get your files on there if they did not exist there already.

The Cloud shell comes prebuilt with python2 and python3, so you should be able to run the program the moment you get it on there. It will generate the files

Here's my relevant shell output after I managed to get everything setup. What we're looking to do is pass the JSON file we generated earlier into the API through the curl command. We then pipe that result to a text file that we can analyze.

A screenshot of a Google Cloud Shell terminal window. The window has a title bar that says "Cloud Shell" and a toolbar with icons for editing, saving, and other functions. Below the toolbar, there's a tab labeled "cloudshell" with a close button and a dropdown arrow. The terminal itself shows a series of commands and their outputs. The first command is "ls", which lists several files: "Los Angeles5-5-2020.json", "New York5-5-2020.json", "Tucson5-5-2020.json", and "tweepy_test.py". The second command is a "curl" command that sends a POST request to a Google Cloud API endpoint, passing a JSON file as data. The output of the curl command is piped into a file named "NewYork-Result.txt". The third command is another "ls", which now includes "NewYork-Result.txt" in the list of files. The prompt is "ryanluu52@cloudshell:~/twitter_sentiment_analyzer\$".

```
ryanluu52@cloudshell:~/twitter_sentiment_analyzer$ ls
Los Angeles5-5-2020.json  New York5-5-2020.json  Tucson5-5-2020.json  tweepy_test.py
ryanluu52@cloudshell:~/twitter_sentiment_analyzer$ curl "https://language.googleapis.com/v1/documents:analyzeSentiment?key=${API_KEY}" -s -X POST -H "Content-Type: application/json" --data-binary @'New York5-5-2020.json' > NewYork-Result.txt
ryanluu52@cloudshell:~/twitter_sentiment_analyzer$ ls
Los Angeles5-5-2020.json  New York5-5-2020.json  NewYork-Result.txt  Tucson5-5-2020.json  tweepy_test.py
ryanluu52@cloudshell:~/twitter_sentiment_analyzer$
```

If we open up one of those text files, we are greeted with something like this in nice JSON format.

```

Tucson-Result.txt — Edited
{
  "documentSentiment": {
    "magnitude": 14.7,
    "score": -0.1
  },
  "language": "en",
  "sentences": [
    {
      "text": {
        "content": "@2020BlueTexas #DisinfectantDonnie visits AZ on Cinco de Mayo encouraging the spread of a virus named Corona just t... https://t.co/ccQLXbHqnr@nemavand I felt so guilty getting corona at the store😞 but i did😞😞",
        "beginOffset": 0
      },
      "sentiment": {
        "magnitude": 0,
        "score": 0
      }
    },
    {
      "text": {
        "content": "Coaches are thinking about what that means when their teams..RT @FrankFigliuzzi1: Join me in the 4p ET hour for the 'China Trap': Pompeo Ties Coronavirus to China Lab, Despite Spy Agencies' Uncertain...If largemouth bass carry Corona virus, I plan on risking my life next week.",
        "beginOffset": 1547
      },
      "sentiment": {
        "magnitude": 0.1,
        "score": 0.1
      }
    },
    {
      "text": {
        "content": "We are all gonna get corona at this point.",
        "beginOffset": 4695
      },
      "sentiment": {
        "magnitude": 0,
        "score": 0
      }
    }
  ]
}

```

Now to actually explain what those values mean. The things that we'll primarily be looking at are the 'score' and 'magnitude' categories since that's what the Google Natural Language API returns to us.

Here is Google's documentation for what the values returned means.

These field values are described below:

- **documentSentiment** contains the overall sentiment of the document, which consists of the following fields:
 - **score** of the sentiment ranges between **-1.0** (negative) and **1.0** (positive) and corresponds to the overall emotional leaning of the text.
 - **magnitude** indicates the overall strength of emotion (both positive and negative) within the given text, between **0.0** and **+inf**. Unlike **score**, **magnitude** is not normalized; each expression of emotion within the text (both positive and negative) contributes to the text's **magnitude** (so longer text blocks may have greater magnitudes).
- **language** contains the language of the document, either passed in the initial request, or automatically detected if absent.
- **sentences** contains a list of the sentences extracted from the original document, which contains:
 - **sentiment** contains the *sentence level sentiment* values attached to each sentence, which contain **score** and **magnitude** values as described above.

Ryan Luu
5/06/2020

Overall while there exist sentiment scores for every individual tweet, what we're interested in are the scores found near the top under documentSentiment. That's because those values tell us the overall evaluation of the entire document (in this case all 100 tweets).

```
{
  "documentSentiment": {
    "magnitude": 14.7,
    "score": -0.1
  },
  "language": "en",
  "sentences": [
    {
      "text": {
        "content": "@2020BlueTexas #DisinfectantDonnie visits AZ on Cinco de Mayo encouraging the spread of a virus named Corona just t...
https://t.co/ccQlXbHqnr@nemavand I felt so guilty getting corona at the store😞 but i did😞😞",
        "beginOffset": 0
      },
      "sentiment": {
        "magnitude": 0,
        "score": 0
      }
    },
    {
      "text": {
        "content": "Pro tip: you have to forgive your failed leade...",
        "beginOffset": 220
      },
      "sentiment": {
        "magnitude": 0.4,
        "score": -0.4
      }
    },
    {
      "text": {
        "content": "https://t.co/qXhCkxbx7GRT @FrankFigliuzzil: Can't have him spouting all that scary science stuff: White House blocking Fauci
from testifying before Congress about...meu deus tô com dor nos olhos e dor de cabeça",
        "beginOffset": 270
      },
      "sentiment": {
        "magnitude": 0,
        "score": 0
      }
    }
  ]
}
```

Overall Score of all 100 Tweets:
Magnitude and Score

Individual Tweet's:
Magnitude and Score

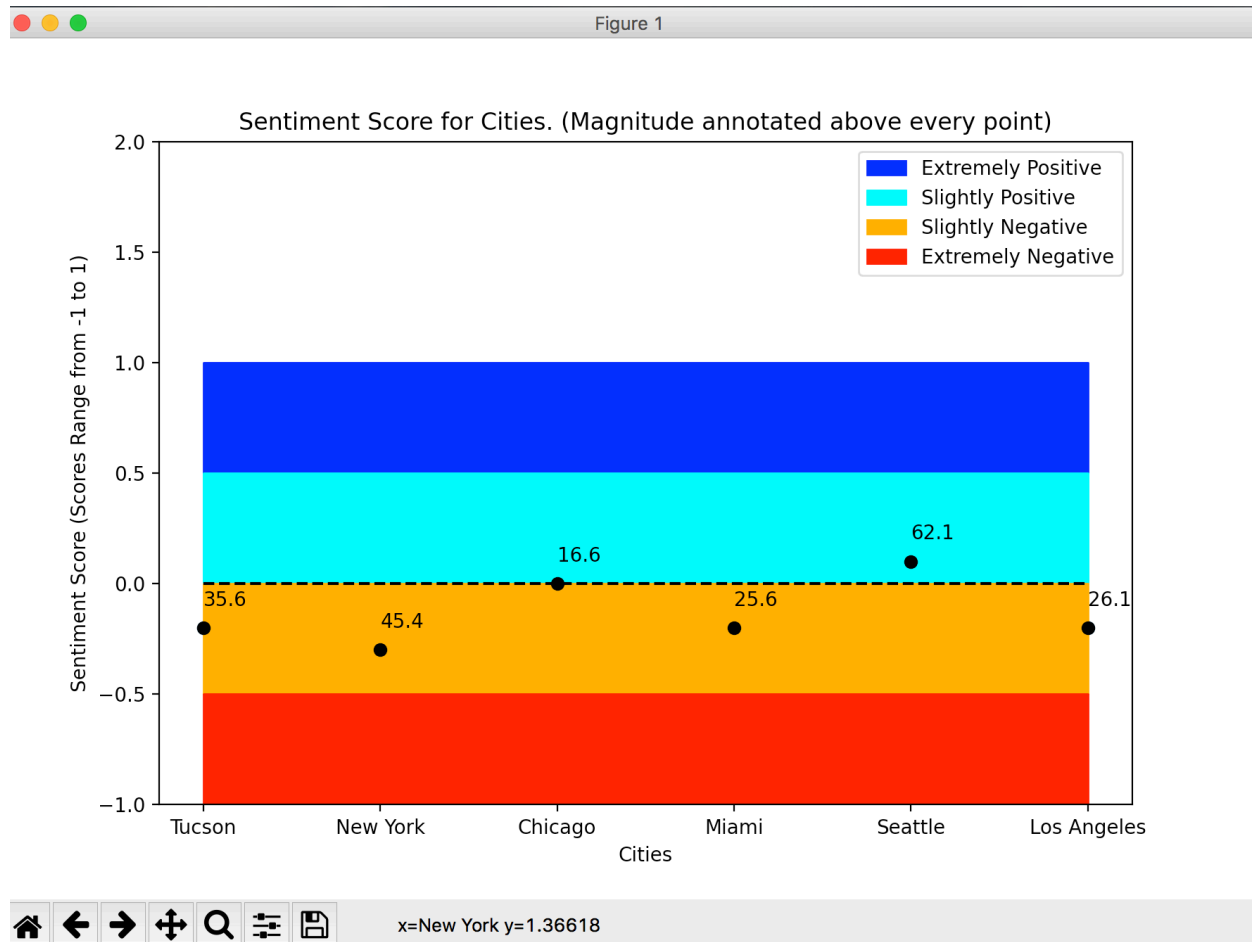
Individual Tweet's:
Magnitude and Score

The score of a documentSentiment tells us the overall emotion of the document and the magnitude tells how much emotional content there is within the document. It's important to note however that the magnitude is often proportional to the length (which is what it means by not normalized in the documentation).

Overall score is more important to look at, but we can use the magnitude to help us differentiate between texts of different lengths or when the 'scores' are neutral. We don't need to worry about different lengths since we grabbed 100 tweets per city, but I'll explain a bit more about the 2nd point in a bit since it is relevant.

For now, though, I wrote quick python script that parsed the JSON file and plotted the overall scores for each of the 6 cities I grabbed data from using matplotlib.

Results based on 100 Tweets that included the word Corona in them.



I color coded the varying degrees of emotions, plotted the overall scores (y axis values), and annotated the magnitudes above each of the scores.

Analysis of Results/ Limitations

Overall, there are some conclusions that we can draw from this plot in particular. First is that we can tell that many of the scores present are negative, which indicate that people are overall unhappy with the corona situation and are more consistently using negative words in their tweets. New York being at the lowest rating for score also makes sense considering the number of cases there as well.

However, we can also tell that those scores are relatively close to 0, which can also potentially imply one of two things. Either neutral feelings/low emotions or mixed emotions. This is where we can use the magnitude scores to help us analyze these grey area cases. For example, we

Ryan Luu
5/06/2020

would generally think that most people are unsatisfied with the corona situation. However, if we look at Seattle, they have an overall score of 0.1. Being above 0 means that it reflects positive emotions.

However, they also happen to have a very high magnitude score. A higher magnitude score in this case then would reflect mixed emotions found within the tweets. Meaning that Seattle being slightly positive doesn't necessarily mean they have a positive outlook on the COVID-19 situation. It's instead rather mixed, with a large pool of negative and positive tweets.

Here is Google's documentation about how to interpret magnitude in these scenarios.

A document with a neutral score (around 0.0) may indicate a low-emotion document, or may indicate mixed emotions, with both high positive and negative values which cancel each out. Generally, you can use magnitude values to disambiguate these cases, as truly neutral documents will have a low magnitude value, while mixed documents will have higher magnitude values.

In retrospect, there do exist some limitations with this project. First would probably have to do with the fact that I could only sample a meager 100 tweets for only 1 day. While I was lucky to witness some deviations, I'm not confident that a single day of information is enough to draw any concrete evidence of any sort. If we wanted to compare fluctuation of emotions from city to city, then we would need a lot more than a 100 tweets, and a lot more than simply 1 day. If this data was taken over, say a 3-5 month period, then I could more confidently say that most people are feeling the same levels of emotion about the Corona situation.

Afterword

This turned out to be a really fun project and I was glad to be able to work with new API's I've never used before. There certainly exists a wealth of things I would like to go back and fix/refine when I have the time, but I felt like I was overall able to do what I set out to do. While I'm not the biggest fan of the Twitter API (due to them constantly timing me out), using Google's Natural Language API turned out to be a lot of fun. I realized that there exists a lot of potential for analyzing data through that particular service. I'm a bit curious as to how they actually end up tagging the data and calculating the sentiment scores themselves, but in terms of how it functioned as a tool; it was surprisingly easy to work with. Overall I can say that it was a fun project to tackle.