Using IBM's Tonality Analysis of Language and Geolocated Tweets to Map Emotional Intensity.

Isaac Callison (ic2d@mtmail.mtsu.edu)
Middle Tennessee State University

April 11, 2020

Abstract

The corpus of social-media data as a whole is growing at an exponential rate and extraction of useful information from that data is exploding. Big Data focused companies and organizations are competing at a furious rate to glean advantages from ever-expanding datasets. One such voluminous and continuously expanding dataset are tweets from the social-media giant Twitter. Tweets from various regions were collected in real-time using Tweepy. Geolocation was obtained through location parsing and geocoding. Using IBM's tone analyzer, the emotional nature of these tweets was analyzed and the predominant emotion and intensity of each was extracted. The results were mapped onto a Google Heatmap.

1 Introduction

There is a convergence of powerful technologies that allow for near- instantaneous notification of current events using available from social media platforms. In some cases these technology tools are the medium by which revolutions are fomented and driven[1]. This is the era of "Big Data" that measures in the realm of zettabytes, the unspoken hypothesis is that one can presume a positive correlation between the growing data and the usefulness that can be extracted from said data[10].

One social media platform for which, at least the possibility of obtaining relevant data is present, is Twitter. This reality is not lost on Twitter. In fact, part of Twitter's business model is selling user data through premium API's[1].

The goal of this project was to gather and analyze tweets for emotive tonality, then display this on a "heatmap"[2] of emotive intensity for a specific region. To this end three separate technologies were utilized. For this project these technologies were interdependent. First, using Twitter's developer API, geolocated tweets were collected from a specific region, then pre-processed to extract latitude, longitude, and text. Secondly, using IBM's Watson and natural language processing capabilities, the tweets were assessed for emotive tonality. Finally, a Google map was displayed with a heatmap layer graphing the intensity of these emotions.

In a perfect world with enough data coming in of the type referenced above, a real-time map of the emotional state of a town or city could be analyzed. The reality fell far short of what was envisioned and the successes and shortcomings will be documented herein.

2 Background

As stated above, there were several interdependent moving parts with this project. With regards to the development environment, a Jupyter Notebook was used for this project to pull in modules, access API's, and make GET and POST requests to those APIs.

The project's datasets for the natural language processing came from Twitter. The Twitter API allows for the triangulation of tweets provided certain data[8]. In general, tweets have a great deal of metadata bundled into their JSON objects.

[1] https://developer.twitter.com/en/premium-apis

[2] A heatmap is a graphical representation of data that uses a system of color-coding to represent different values.

A key part of this project was the triangulation data associated with tweets. This is certainly not the first project to leverage geotagged tweets for Big Data. Projects have used it for situational awareness in diasters[9] and for tracking tourism globally [4], to name just two. Papers in the recent past have created high-quality mappings for geo-tagged tourist tweets[5]. However, the ability to extract out latitude and longitude from a tweet, which was previously available, was culled in June of 2019[3]. In response, other methods were used to triangulate tweets.

Using developer authentication, and a GET request with certain location parameters, one can obtain a list of current tweets within a search radius in JSON format. A great deal of data is returned in this format, but from these tweets one can glean a myriad of information, including up until recently, the latitude and longitude of the tweet.

The second part of this project was natural language processing with IBM's Watson using tonality analysis. IBM has a cloud computing program with various machine learning capabilities[6], one of which is tonality analysis. The natural language tonality processing that Watson offers can, among other things, extract emotion from a corpus. In this specific case, a variety of emotional states could be extracted including: fear, joy, analytical, confident, sadness, tentative, and anger[4]. Further, the intensity of a specific emotion can be derived at the sentence and document level.

As a graphical representation of the data collected, a heatmap was used. A heatmap overlay is a feature offered by Google Maps. It can create a visualization to depict the magnitude variation of data at a range of latitudinal and longitudinal points. A heatmap is very useful when you have a great deal of data points of varying magnitude and their geographic position. One type of dataset that lends itself well to such graphical representation is eathquake data[5]. When the heatmap layer is enabled, a colored overlay will appear on top of the map. By default, areas of higher intensity or magnitude will be colored red, and areas of lower intensity will appear green[3].

3 Research Method

The consoles and APIs of Twitter, IBM's Watson, and Google were accessed. The three aformentioned services required developer accounts to be used. Twitter's API required an application and pre-approval. These credentials were secured and inserted directly into the code for ease. Initially, separate Notebook's were used to test the various API's before integration into the main Notebook. These accounts were free to use, up to a point. After some extensive testing with IBM's Watson an upgraded account had to be setup to continue.

All work was done on a Linux desktop environment. Code was written in Python 3. Anaconda was used to launch the Jupyter environment. Postman, a REST API testing software was used to experiment with Twitter's API[7] before integrating Tweepy's tweet streaming code into the main Notebook.

Using the Python Tweepy[6] module GET requests were made to stream a set number of tweets for analysis. The original plan was to extract the latitude and longitude directly from the tweet, and of course the text of the tweet itself. Because Twitter no longer provides the latitude and longitude

[3]The author was not aware of this change when launching this project. Please see: https://www.engadget.com/2019-06-19-twitter-removes- precise-geo-tagging.html

[4]These are the exact tags assigned to individual sentences by IBM's analyzer.

[5]An earthquake dataset is included in the gmaps package and is used for illustration and tutorials

[6]https://www.tweepy.org/

of tweets a workaround was required. Fortunately, some of the tweets also had embedded address locations. An open-source API called Nominatum[7] was used to reverse geocode addresses which were still embedded in tweet data.

Tweepy allows for filtering by a bounding-box defined by four coordinates delineated by two latitude points and two longitude points. Several of these regions were set into variables for use including the rough bounding-boxes of New York City, Nashville, Hawaii, and The United States of America.

Using the modified method with Nominatum, the Tweets pulled in by the Notebook and subjected to significant pre-processing. Of the multitude of tweets that were pulled in, only a fraction made it through the several filters set up.

Tweets less than 45 characters in length were excluded as it was presumed no meaningful emotional context could be derived from a shorter tweet. Tweets were excluded if the tweet metadata indicated it was in any language other than English to avoid confusing the Tone Analyzer. Tweets with incomplete addresses were filtered out as well as the geocoder would only work with complete addresses. Initially retweets were excluded, then allowed on the theory that a retweet would mirror the retweeter's mood and allow for more data collection. Finally, a regex expression was used to strip out extraneous symbols from the tweet.

The cleaned text from the obtained tweets were passed into Watson for tonality processing. IBM provides a Watson Software Development Kit for integration into Python and Jupyter[8].

A Google Map displayed a heatmap layer with the attendant emotion and intensity. For instance, in areas of a region where the tweets have low sadness, green shading predominated, changing to red as the intensity of that emotion increased.

4 Results and Analysis

A description of the results gleaned through this process is best addressed through an analysis of each API service. As the project progressed, three interdependent API services became four interdependent APIs. Initially considered were, Twitter, IBM Watson, and Google. Once it became clear that latitudinal and longitudinal information could not be pulled directly from twitter data, a fourth API, Nominatum was brought into the fray for geocoding.

Pulling tweets using Tweepy and Twitter's API proved to be fairly straight- forward once the authentication code was properly set up. As previously lamented, the fine-grain location data of each tweet was removed by Twitter in mid-2019. As a result, tweets could be pulled from a specific region through Tweepy, but the exact latitude and longitude of that tweet was not discernable. What was initially envisioned was a fine-grain heatmap showing data across a city, but this was not to be the result.

There were two crippling realities grappled with concerning Twitter's API. The first is that though thousands of tweets could be pulled in a matter of seconds, after filtering by length, language, and geocoding through Nominatum, attrition was extremely high. A flood of tweets was turned into a trickle whereby a new usable tweet would come only after several seconds.

The second issue was rate-limiting [9]. With all the filters in place, less than 300 tweets could be

[7] https://nominatim.org/

[8] https://cloud.ibm.com/docs/services/ watson?topic=watson-using-sdks

[9] https://developer.twitter.com/en/ docs/basics/rate-limiting

pulled in during a half-hour period. At that point an error would be returned by the Twitter API. This exception was caught in the code and an exponential backoff was implemented to attempt the stream again at a later point. By relaxing a few of the filters, including allowing retweets, the number of tweets available before rate-limiting kicked in was increased to around 1200. However, it could take easily an hour to pull in that many tweets, which defeated the ”real-time” nature of the project. The usable tweets and their geocodes were loaded into a Pandas dataframe object for manipulation.

Watson’s Tone Analysis worked with very few caveats. The free account was limited to 2500 API calls per month. As such a premium account was set up to continue its use. The tone analysis worked well at classification, but the documentation gave little guidance on how to parse the data returned and offered no function calls that could be used to pull out specific data objects.

The cumbersome JSON data object returned was converted to Python nested dictionary objects and parsed. Often more than one emotive quality was returned but sometimes the API could not determine the emotive nature of the tweet and would return null. The tweet was classified by the predominant emotional quality returned. The aforementioned Pandas dataframe was amended to include these attributes and the magnitude of these attributes as returned by the tone analyzer.

	date_obj	tweet	latitude	longitude	magnitude	emotion
0	2020-04-11 15:16:27	JIMBRO GOT MARRIED love you guys so much holy ...	42.360253	-71.058291	0.786025	joy
1	2020-04-11 15:16:31	Lol my brother is successful business man but ...	40.789624	-73.959894	0.696755	joy
2	2020-04-11 15:16:37	Not unrelated Excellent proximity work Timelin...	40.728158	-74.077642	0.801827	analytical
3	2020-04-11 15:16:42	Every corpse that was registered Democrat shou...	40.730309	-73.326559	0.660207	confident
4	2020-04-11 15:16:48	Those who do lazy work end up having to do it ...	40.846651	-73.878594	0.587205	sadness
5	2020-04-11 15:16:54	Covid19 I didn t mean to infect him he caught ...	40.749824	-73.797634	0.589295	analytical
6	2020-04-11 15:16:54	If we are including women then Laura Ingraham ...	40.949172	-74.237680	0.889390	tentative
7	2020-04-11 15:17:08	You beat me to it with this op ed I am more th...	40.743307	-74.032375	0.000000	null
8	2020-04-11 15:17:08	SupaDupaASS Lol it solid but 9months straight ...	43.157285	-77.615214	0.727798	confident

Figure 1: Selected rows of Pandas Datframe

A Google API key was used to configure gmaps and display a heatmap overlay. Of all the API’s this one, once setup[10], worked the most reliably. The heatmap was populated using the dataframe. However, to clarify the type of emotion that was represented, only those rows that had the selected emotion were displayed. For instance, the following code could be used to load all rows with the emotion ”joy” for later display:

```
// Heatmap Notebook
// Defining a variable df_joy from the df object wherein all rows selected have
//"joy" as the requsite emotion

df_joy = df.loc[df['emotion'] == 'joy']
```

Figure 2: Dataframe object with certain rows selected.

The code was adjusted to pull in one-thousand tweets. The total time to complete this task

[10]See the environment setup in the appendix.

was roughly nintey-minutes. As the time signatures of each usable tweet was saved in the Pandas dataframe, the time between the first and last tweet pulled can be illustrated:

```
In [31]: print(df.head()["date_obj"])
         print(df.tail()["date_obj"])

0    2020-04-11 17:29:39
1    2020-04-11 17:29:44
2    2020-04-11 17:29:49
3    2020-04-11 17:29:49
4    2020-04-11 17:29:55
Name: date_obj, dtype: object
995    2020-04-11 18:58:50
996    2020-04-11 18:58:51
997    2020-04-11 18:58:51
998    2020-04-11 18:58:52
999    2020-04-11 18:58:59
Name: date_obj, dtype: object
```

Figure 3: Time required to glean one-thousand usable tweets.

Of the one-thousand usable tweets pulled from the New York City region, and displayed by the variable emotion, this result is produced the following heatmap:

Figure 4: Heatmap view of New York City and surrounding regions.

This is promising until one scrolls in closer. Instead of a multitude of tweets of varying magnitude spread all over the city, we see a few clusters:

Figure 5: Close up view of New York City Heatmap.

That brings us to the last API utilized, Nominatum. It seems from the above result that Nominatum has, through its geocoding services, assigned all of the tweets to one of the five New York City boroughs. Nominatum had the minor, yet crucial task of geocoding the addresses pulled from usable tweets. In fact, whether a tweet had a fairly complete address for geocoding purposes was one of the criteria for whether or not a tweet was usable. While this open-source API provided a work-around for the lack of Twitter embedded coordinates, it was not completely accurate, as illustrated above in figures 4 and 5. If a partial address was enough to provide coordinates, Nominatum was used. The poor results are not necessarily the fault of Nominatum. Nominatum was merely returning coordinates from the addresses embedded in the tweets, and the addresses appear to be of such general nature that instead of an even dispersal of points we have a few hotspots.

5 Conclusion and Future Work

We live in an age of ever-expanding data, ”Big Data” as it is often called. As the volume of this data increases, data scientists will seek to harness this information through new technologies, for better or worse. In this project, an attempt was made to sift a vast amount of Twitter data in real-time to graph the emotional state of particular regions. Unfortunately, the particular type of implementation envisioned was ultimately unsuccessful due to the removal of precise tweet geotagging. Graphing the emotional intensity of differing regions based on twitter content proved intractable due to minuscule amount of usable and still geographically imprecise tweets that could actually be assessed for emotional content.

References

[1] Alhindi, W., Talha, M., and Sulong, G. The role of modern technology in arab spring. *Archives des sciences 1661-464X*, 65:1661–464, 09 2012.

[2] Callison, I. Jupyter Notebook. `https://github.com/cthulhu1988/SelTopicsAI/tree/master/NLPpaper`, 2020. [Online; accessed 9-March-2020].

[3] Google,. Heatmap API. `https://developers.google.com/maps/documentation/javascript/heatmaplayer`, 2020. [Online; accessed 10-March-2020].

[4] Hawelka, B., Sitko, I., Beinat, E., Sobolevsky, S., Kazakopoulos, P., and Ratti, C. Geo-located twitter as proxy for global mobility patterns. *Cartography and Geographic Information Science*, 41, 11 2013.

[5] Hu, F., Li, Z., Yang, C., and Jiang, Y. A graph-based approach to detecting the tourist movement patterns using social media data. *Cartography and Geographic Information Science*, 05 2018.

[6] IBM Incorporated,. IBM's Watson: Cloud Computing. `https://cloud.ibm.com/apidocs/tone-analyzer`, 2020. [Online; accessed 10-March-2020].

[7] Postman Incorporated,. Postman API. `https://www.postman.com/`, 2020. [Online; accessed 29-February-2020].

[8] Twitter Incorporated,. Twitter Documentation: Geocode API. `https://developer.twitter.com/en/docs/geo/places-near-location/api-reference/get-geo-search`, 2020. [Online; accessed 29-February-2020].

[9] Verma, S., Vieweg, S., Corvey, W., Palen, L., Martin, J., Palmer, M., Schram, A., and Anderson, K. Natural language processing to the rescue? extracting "situational awareness" tweets during mass emergency. 01 2011.

[10] Villars, R. L., Olofson, C. W., and Eastwood, M. Big data: What it is and why you should care. *White Paper, IDC*, 14:1–14, 2011.

A Coding Environment

All code was run on Ubuntu 18.04 LTS. Anaconda was installed and Jupyter Notebooks were utilized. All code was written in Python 3. There was some difficulty in getting Google Maps to display in JupyterLab and instead Jupyter Notebook 6.0.0 was utilized. Several modules were required to run the various APIs utilized in this project. The following process was used to install Anaconda and the modules used:

```
// From a bash command prompt:
curl -O https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh

// Enter command and follow installation prompts
bash Anaconda3-5.2.0-Linux-x86_64.sh

// After installation of Anaconda, these modules should be installed:
pip install ibm_watson
pip install tweepy
pip install geopy

// To get gmaps to display properly:
jupyter nbextension enable --py --sys-prefix widgetsnbextension
pip install gmaps
jupyter nbextension enable --py --sys-prefix gmaps
```

Figure 6: Anaconda & Module Setup