**Twitter Sentiment Analysis of Climate Change**

University of Maryland, Baltimore County

Sairashmika Jammi

DATA 606 Capstone in Data Science

Spring 2020

**Project Overview:**

Twitter is an extremely popular social media platform that allows people from across the world to share their opinions regarding topics that concerns them or post messages via their account. Sentiment analysis is a text classification tool that “analyzes an incoming message and tells whether the underlying sentiment is positive, negative or neutral [1]”. The goal of my capstone project is to analyze twitter data regarding topics relating climate change and create a machine learning algorithm that detects the sentiment of those tweets and classify tweets real time as positive or negative sentiment.

Sentiment analysis can be a powerful tool because the public’s opinion on product or political matter is extremely important and can change at a moment’s notice. It is in the best interest that companies or government understand the publics opinion on such matters. This project can help analyze tweets real time and give an accurate representation on the public’s view on certain topics, such as climate change in a more accurate and time effective way than a traditional human report. Climate change is an extremely important topic, as the real time effects of climate change are being seen across the world.

Additionally, I wanted to conduct further analysis on my dataset to analyze and identify trends in negative or positive tweets and demonstrate these trends in a form of different visualizations such as graphs and word clouds.

**Literature Overview:**

There are various different projects that have conducted Twitter sentiment analysis, using various different techniques to classify their tweets. When analyzing the projects online, I learned that many of the projects used Natural Language Processing [2] for the analysis of tweets, which I plan to use as well.

One project did not only classify the tweet but calculated the sentiment of each tweet and built dashboards to visualize it [4]

The difference between the projects I have researched and mine is that I plan to use not only Natural Language Processing for creating my classification, but also Naïve Bayes. This is so I can experiment with these two popular classifications.

**Dataset Extraction, Cleaning and EDA:**

The primary dataset I will using in this project is a one that I had to capture. I captured tweets relating to climate change for the past month. This dataset will be limited to English speaking tweets; therefore, my dataset is limited to people that can only voice their opinions in English. To capture these tweets, I will use python packages such as Tweepy to access my Twitter API account and stream the tweets and write the data into a CSV file and a data frame.

A screenshot of a cell phone

Description automatically generated

Above is how the data looked initially, before any cleaning. As you can see there is a lot of retweets, link and emoji’s that made it difficult for my ML model to process the tweets. Therefore, I had to rethink how I was going to process the tweets. My initial plan of using streaming listener was not going to work for this project.

A screenshot of a cell phone

Description automatically generated

Eventually, after much fine turning and research, I realized that for this project I must approach the data extraction part a little differently than how people in the past did. As I conducted my research, I realized that I needed to use a slightly different tweepy tool called tweepy cursor. This automatically read my tweets into a list, which greatly improved processing time since I didn’t have to convert from CSV to data frame or list.

After I obtained my tweets, I cleaned the tweets in various forms. First, I converted all the words in the list into lower case. This was because if there were two of the same words and one was in upper case and the other was in lower case, the algorithm would interpret those words as different. Additionally, I had to remove the stop words from the tweet. Stop words are unimportant words that need to be eliminated, so that model is more accurate. The Natural Language Toolkit, otherwise known was NLTK, has a built-in library of stop words that I downloaded. Once I removed the stop words from the tweet, my tweets were clean. After that I used text blob, a natural language processing library that is built upon NLTK, to assign sentiment to the tweets. It was imperative that I cleaned the tweets before I assigned the sentiment value as it noises from the uncleaned tweets will an false sentiment value. The sentiment value was a value ranging from -1 to 1, and I assigned it as Positive if the value was above 0 and negative if the value was below 0.

*A close up of text on a wooden surface

Description automatically generated*

***Figure 1:*** *Displays a word cloud displaying the most popular words in the negative sentiment tweets*

A screenshot of a cell phone

Description automatically generated

***Figure 2:*** *Displays a word cloud with most popular words in the positive sentiment dataset (below).*

Upon initial analysis of my dataset, I was not surprised to see words such as COVID and action and science on the word clouds. Both sides of the spectrum are talking about the same things, such as the effects that the pandemic has on climate change etc. I was surprised to see that many of the words on the positive sentiment word cloud overlapped with the negative sentiment word cloud.

A screenshot of a cell phone

Description automatically generated

***Figure 2****: More positive classified tweets than negative.*

I was also surprised to see that there were more positive tweets than negative tweets. In my Deliverable 1, I hypothesized that there were going to be more negative tweets relating to climate change than positive as I was under the impression that people tend to voice their negative opinion on social media more than a positive one. Therefore, in my sample dataset to see the overwhelming positive tweets was shocking.

After this, I tokenized my tweets which means that in order to accurately process the tweet in the machine learning algorithm the tweets needs to split into words rather than stay a cohesive sentence. After this I had to lemmatize my tweets via another package from NLTK, a normalization technique that “analyzes the structure of the word and its context to concert it to a normalized form” [5].

A screenshot of a cell phone

Description automatically generated

***Figure 3****: This shows the tweets after all the cleaning and transforming (far right)*

**Model Construction:**

In order to construct my model, I first need to convert the tweets into a vector that my machine learning algorithm will be able to understand. I used two different approaches to do this: Bag of Words and TF-IDF.

Bag of words is a model that counts the frequency in which the word occurs in the tweet, weights that count and normalizes it per the tweet length. Luckily enough, SciKit Learn has a Count Vectorization function that allowed me to obtain the vector I needed. I plugged in the lemmatized tweet into the matrix and got my vector.

Another way to retrieve weights for my tweets was to use the TF-IDF (Term Frequency, Invert Document) approach which is equally popular tool in text mining. In TF-IDF the “importance increases proportionally to the number of times the word appears in the document” [5].

After processing the data via bag of words and TF-IDF, it was time to pass it through a machine learning algorithm. I decided to use a logistical regression algorithm instead of Naïve Bayes due complexities surrounding my dataset. I wanted to compare which type of vectorization process performs better, and I thought that bag of words would work better.

In order to see which performed better, I used the F1 score to evaluate how my classification algorithm performed. The higher the F1 value, the more accurate the classification algorithm is.

**Results and Conclusion:**

A picture containing knife

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Although the F1 score was very similar for both the types of models, the Precision and Recall score was higher for Bag of Words, therefore it was the more accurate in determining the sentiment of the tweet. Although I would have liked for my recall score to be higher, the model was as accurate as it could be with the dataset I had.

**Setbacks and Future Endeavors:**

There were various different setbacks in my project, all mainly surrounding my dataset. At first I was unable to obtain the amount of data I wanted to, due to the massive amount of tweets being made every day. I wanted to analyze the popularity of words across a span of five years and which words were more popular at what times and correlate it to word events that were happening at that time. Unfortunately, I simply lack the machine power and the time to scrape that amount of data and store it. Additionally, I had problems with how I wanted to extract my data, and I did not know it at the time but the process in which I extract my data greatly affected how my project was conducted. In the future, I plan to use a premade dataset, so I can focus more of my efforts into creating various different machine learning algorithms such as Naïve Bayes or possible neural network. A much smaller endeavor that I wish I had time for was creating a user interface in which the user enters a tweet and the program classifies it as positive or negative using my machine learning algorithm. Furthermore, I also would like to focus more on creating more complex visualizations such as data maps showing the density of where tweets are originating from.

References:

[1] Gupta, S. (2018, January 19). Sentiment Analysis: Concept, Analysis and Applications. Retrieved May 11, 2020, from https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17

[2] <https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/>

[3] <https://towardsdatascience.com/twitter-sentiment-analysis-classification-using-nltk-python-fa912578614c>

[4]: <https://realpython.com/twitter-sentiment-python-docker-elasticsearch-kibana/>

[5] DigitalOcean. (2019, October 30). How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK). Retrieved May 11, 2020, from https://www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk