Sentiment Analysis Research and Implementation

Data Visualisation and Analysis CA 2 Data Wrangling Assessment

Implementing Sentiment Analysis wasn’t required for this assignment but during research on the topic I found some articles that discussed, and showed in-part, how to implement it with a library that I could use to include some analysis in my python program at a basic level without much additional work and I wanted to do it.

The main article that I followed was one created by *Rodolfo Ferro*, a Python developer who works at FutureLab which is a European Customer Centricity and Customer Experience (CX) consultancy company. In his article he describes how he went about implementing “Sentiment analysis on Trump's tweets using Python“. This article included good descriptions and breakdowns of code snippets that are provided throughout. This article can be found here: <https://dev.to/rodolfoferro/sentiment-analysis-on-trumpss-tweets-using-python->. What differs from the implementation in the article and my implementation is that the article analyses the sentiment of a *person’s own tweets*, while my implementation analyses the sentiment of *tweets that reference a topic*.

The library used is called TextBlob, which is a NLP (Natural Language Processing) that can perform “tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.” <https://textblob.readthedocs.io/en/dev/>

My use of this library is code taken and modified from Rodolfo Ferro’s article. The first function, clean\_tweet\_text(tweet\_text), uses regular expression joining, substitution and splitting to clean a string of links and special characters in preparation for sentiment analysis. I used this right after fetching each tweet and before putting into my list of dictionaries structure.

The second function analyseSentiment(tweet\_text) takes in a string and calls on TextBlob to return an analysis object. This object contains the sentiment polarity, which is a float value between 1f to -1f where >0 can be considered a positive perception, 0 can be considered a neutral perception and <0 can be considered a negative perception.

The second function is used with every tweet text in the generated data frame in a for loop in the third function, displaySentimentPercentages(data\_frame). This function compiles a list containing all the polarities of the tweets and then breaks that list down into 3 more lists, positive\_tweets, neutral tweets and negative tweets. The % of each perception is the size of each sub list in reference to the size of the whole data frame.

The first search term that I tested this with was “Fallout76”, a newly released game that has received a good bit of negative reception amongst long-time fans of the intellectual property (myself included) as it has taken a direction that strips it of what made the previous games so enjoyable.

The second search term that I tested was “Donald Trump”, who is an extremely polarizing figure that has gained broad negative perception amongst many people throughout the world.

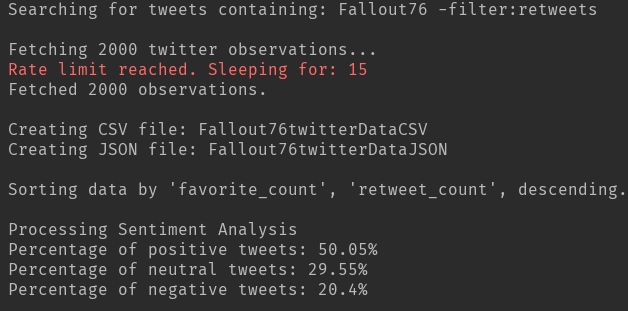
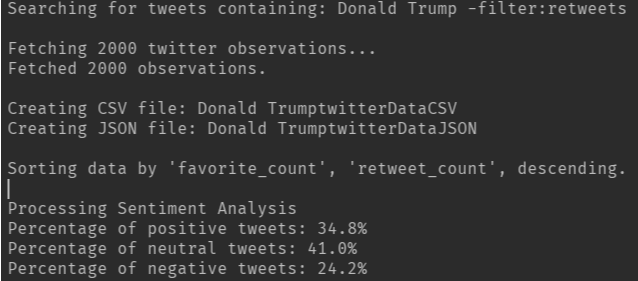
Here are the results, both use 2000 tweet text observations, which is an admittedly small sample six relative to the sheer number of tweets that occur.

Figure 2: Topic "Donald Trump" sentiment analysis.

Figure 1: Topic: “Fallout76” sentiment analysis.

Due to the limitations pertaining the number of tweets that are analyzed and my lack of understanding pertaining to the accuracy or performance of the library that was used, I can’t know if these metrics have much integrity. Regardless, this task in combination with the assignment was genuinely fun for me and has provided me with a much more solid python programming foundation going forward. I would absolutely continue improving this program and exploring other features if time permitted.