Capstone Project 2
"Sentiment analysis of Tweets"

Milestone Report 1

Problem Statement

What is a Sentiment Analysis and why is it useful?
Sentiment Analysis represents the use of Natural Language Processing to determine the attitude, opinions and emotions of a speaker, writer or other subject within an online mention. In other words, it is the process of determining whether a piece of writing is positive or negative. A human is able to recognize and classify a text into positive or negative. However, a computer not but can learn to do so.

I would like to perform a sentiment analysis on Tweets containing *Trump* and *Ukraine*.
The Trump–Ukraine scandal is an ongoing political scandal in the United States. It revolves around efforts by U.S. President Donald Trump to coerce Ukraine and other foreign countries into providing damaging narratives about 2020 Democratic Party presidential candidate Joe Biden and about Russian interference in the 2016 United States elections.

Different types of people could be interested to know the proportion of positive vs negative tweets on this topic: politics, governments, individual people for or against Trump.

Methodology

I will be using the Twitter API to collect a Test set based on keywords. A function will return a list of tweets that contain our keywords selected. Each tweet's text will see itself attributed a label ('positive' or 'negative') to classify each tweet as positive or negative. The Training set will be downloaded because it has to be labelled into 'positive' or 'negative' on a big amount of tweets. The Training set is critical to the success of the model since our model will "learn" how to do create a sentiment analysis based on the Training set.

The steps that we will follow to perform the sentiment analysis are:

- Preparing the Test set
    - getting the authentication credentials
    - authenticating our python script
    - creating the function to build the Test set
- Preparing the Training set
- Pre-processing Tweets in the Data Sets (both Test and Training)
- Naive Bayes Classifier
    - build a vocabulary/list of words in our training data set
    - match tweet content against our vocabulary
    - build our word feature vector
    - training the classifier
- Testing the model

Preparing the Test set

Getting the authentication credentials

To do so, I visited the Twitter Developer website, applied for an API access and waited for Twitter to accept. After getting the approval email, I went back to the website and clicked on "Create an app". I filled in the information asked and finally got Consumer API keys, as well as Access token & access token secret.

Authenticating our python script

I then imported the twitter library, and created a twitter.Api object containing the credentials mentioned previously.

Creating the function to build the Test set

I build a function that takes a search keyword as in input, searches for tweets that include this keyword and returns them as twitter.Status objects that we can iterate through. The problem is that Twitter limits the number of requests we can make through the API for security purposes. This limit is 180 requests per 15-minute window. For the sake of simplicity, we will limit the search to 100 tweets for now, not exceeding the allowed number of requests. The function name is **buildTestSet** and will return a list of tweets that contain our search keyword.
Every tweet's text is associated with a label that is NULL for now. The reason for this is that we are going to classify each tweet as Positive or Negative later on, in order to determine whether the sentiment on the search term is positive or negative, based on the majority count.

Preparing the Training set

As suggested by examples of Sentiment Analysis, the training set is critical to the success of the model. Data needs to be labeled properly with no inconsistencies or incompleteness, as training will rely heavily on the accuracy of such data and the manner of acquisition.
Therefore, I will be using a downloadable training set, and the tweets contained in it were all labeled as positive or negative depending on the content. This is exactly what a training set is for.

The training set used comes from Niek Sanders' Corpus of over 5000 hand-classified tweets, which makes it quite reliable. The corpus includes a keywords (topic of the tweet), a label (pos/neg) and a tweet ID number of every tweet. The file was obtained here: https://github.com/karanluthra/twitter-sentiment-training/blob/master/corpus.csv

I will be using the API to get the actual tweet text through each tweet's ID number included in the Corpus we have. Since Twitter API has a limit mentioned above, I will make the code rest for 15 minute to download 5000 tweets.
The function to do so does the following:

- two inputs: "corpusFile" – a string path to the Niek Sanders' corpus file I downloaded (containing tweet's **topic**, **label** and **id**), and "tweetDataFile" – a string path to the file we would like to save the full tweets in (containing tweet's **text**, **topic**, **label** and **id**)
- started with empty list *corpus*
- then opened file "corpusFile" and appended every tweet from the file to the list *corpus*
- get the text of tweets based on the ID

- loop through the tweets in *corpus* calling the API on every tweet to get the Tweet.Status object of the particular tweet.
- use status object to get the text associated with it and push it into the "trainingDataSet"
- sleep for a couple of minutes to avoid the API limit mentioned

The tweets retrieved through the API are saved in a CSV file "tweetDataFile".

Pre-processing Tweets in the Data Sets

Before moving on to the classification (pos/neg) section, there is some cleaning up to do. For a Sentiment Analysis, words are the most important input used. However, punctuation, images, videos, URLs, usernames, emojis do not contribute to the analysis of tweets, and will have to be removed. This data cleaning step will be applied on both the Training set and the Test set. Creating the function to perform data cleaning includes the following steps:

- import necessary librairies:
  - *re*, to parse strings
  - *ntlk*, most commonly used Python library, Natural Processing Toolkit
- create a function, **processTweets**, that loops through all the tweets input into it, calling the function **processTweet** on every tweet in the list
- **processTweet** makes all text in lower-case, removes URLs and usernames, removes '#', gets rid of duplicate characters, and tokenizes text into words

Once the function is created, we can apply it on our Training set ("trainingData") and Test set ("testDataSet").

The data is now ready, we can proceed to building our model.