Does the news have impact on the stock market?

An NLP project that relates to Stock Market Prediction

Charita Ramnathsing

JUNE 27TH, 2020
MILESTONE REPORT CAPSTONE II

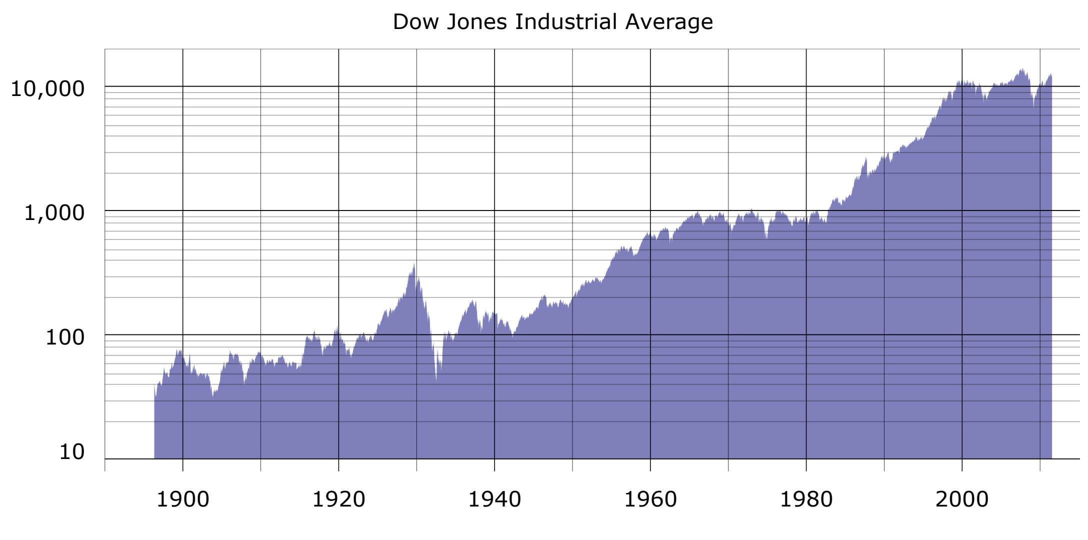
Overall Problem Statement - Summary

Problem statement

It’s always interesting to know that an age-old institute like The Dow Jones Index (DJIA), is sensitive to impacts that are happening in and around the world.

The index is a stock market index that measures the stock performance of 30 large companies listed on stock exchanges in the United States and was first calculated on May 26th 1896. It is today one of the most watched stock-indices in the world. It was created by the editors of the Wall Street Journal and the owner of Dow Jones & Co. The Dow Jones Industrial Average (DJIA) is named after Dow and one of his business associates, a statistician named Edward Jones. Dow was known for his ability to ethically explain complicated financial news to the public. He believed that investors needed a simple benchmark to indicate whether the stock market was on the rise or the decline. And so did Dow choose several industrial-based stocks for the first index, which was reported as an average of 40.94.

Although the Dow index is compiled to gauge the performance of the industrial sector within the American economy, the index's performance continues to be influenced by not only corporate and economic reports, but also by domestic and foreign political events such as war and terrorism, as well as by natural disasters that could potentially lead to economic harm.



Since the Dow is mostly a reflection of our economy, it is nonetheless to say that it would be interesting if we could make a model that could eventually predict if the index would rise or fall by looking at the impacts in the news. People in decision making roles, market leaders, banks, and many more would benefit if there eventually could be a good algorithm in predicting the index.

Dataset

For this Capstone i have used the dataset available on Kaggle.com and Yahoo Financial. These websites make data available in several formats, without paying for subscriptions :

1. Dow Jones Industrial Average (DJIA) (Range: 2018-01-01 to 2018-12-31); DJIA_table.csv

2. 'articles_for_prediction.csv'

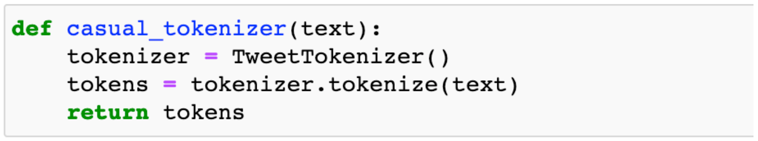
The range of data is chosen for the year 2018, in this way I could make use of the most available data in at Yahoo Financial in combination with the data at Kaggle.com

Both datasets have different amounts of rows, the DJIA-file has only 250 rows while the articles_for_prediction-file has 40744 rows. Eventually I am going to add each result and day of the DJIA to the rows and days of the articles_for_prediction-file. The DJIA-index is a daily index and has also no results in the weekends hence the few rows in this dataset.

Collecting Data and Cleaning Data

This part of collecting and cleaning data will be different for the planned 'LSTM' model. When using this model, I won't be cleaning much of the data and feed the model the raw data as is. In this paragraph I have mostly used the cleaning of the data for visualization purposes.

Having the data in a csv format it is so much easier to go through the information and eventually make a good visualization of it. There are different ways of collecting and importing data, but my datasets were already available in csv format. Therefor the step towards cleaning the data became easier as well. In general, I would start with a few steps in general using RegEx. But for cleaning the textual data, I have considered every step. Meaning, with too much cleaning you can lose information for the context of the text and the same would happen with too little of cleaning of the data:



```
def casual_tokenizer(text):
    tokenizer = TweetTokenizer()
    tokens = tokenizer.tokenize(text)
    return tokens
```

```python
def process_text(text):
    text = text.replace("U.S.", "usa")
    text = casual_tokenizer(text)
    text = [each.lower() for each in text]
    text = [re.sub('[0-9]+-', '', each) for each in text]
    text = [w for w in text if w not in punc]
    text = [w for w in text if w not in ENGLISH_STOP_WORDS]
    text = [each for each in text if len(each) > 1]
    text = [lemma.lemmatize(each) for each in text if not each.isnumeric()]
    return text
```

There are different tokenizers but each with its flavors. For this data I have chosen 'TweetTokenizer' from nltk because of the comparable character of short and impactful headlines in articles and text of Twitter. What you would like to do with tokenization is to split the data into 'tokens' for eventually making it more understandable for the algorithm. When tokenizing in the incorrect way, it can slightly influence the outcome of the accuracy of the model.

The same for stemming and lemmatization. When lemmatizing a word, it is reduced to a word that is related to its lemma (= where the word comes from). E.g running, runs, ran are from the same lexeme with 'run' as lemma. When stemming group of characters (aka a word), the word is also reduced to its stem only the stem doesn't need to be a normal dictionary word: argue, argued, arguing is being stemmed to 'argu' with PorterStemmer[1].

Explore Data with visualization using Bokeh

From Vectorization

Every algorithm working with text can understand the semantics better when using a vector. Word Embeddings or Word vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics. The process of converting words into numbers are called 'Vectorization'**.**

With vectorization we can work now with the data in a numerical way, rather than in just strings of characters. The text of the dataset, and then specifically the headers of the articles, have been cleaned in the previous step and it is now time to transform this combination of characters into matrices and vectors. One way to do that is by making use of TfIdf (Term frequency , Inverse document frequency). The Scikitlearn library has the package TfidfVectorizer that transforms text to feature vectors, but you have to feed it with the documents which you would like to use for the algorithm. It basically converts a collection of raw documents to a matrix of TfIdf features and it can become very useful for scoring words in algorithms with text processing.

[1] https://en.wikipedia.org/wiki/Stemming

To put it in more formal mathematical terms, the TF-IDF score for the word i in the document j from the document set N is calculated as follows[2]:

Tf-idf formula

$$w_{i,j} = tf_{i,j} * \log(\frac{N}{df_i})$$

$$w_{i,j} = \text{tf-idf weight for token } i \text{ in document } j$$

$$tf_{i,} = \text{number of occurences of token } i \text{ in document } j$$

$$df_i = \text{number of documents that contain token } i$$

$$N = \text{total number of documents}$$

TF-IDF enables us to give us a way to associate each word in a document with a number that represents how relevant each word is in that document. Then, documents with similar, relevant words will have similar vectors. Per document there will be a weigh of the word and the total score will be able to eliminate or keep the word 'important' related to the full set of documents.

Headlines in online news sites are increasingly packed with just the search-friendly keywords that identify the story and tfidf is a good way of filtering content quality from content randomness meaning getting to give more value to the words which can be more meaningful per sentence instead of more common words. On the other hand, my next step for text mining will not use tfidf but its own interpretation of the text based on deep learning of the text. The reason for using the tfidf in this context of the matter, is to use it for visualization and to use a non-heavy-computational example of understanding data in more depth.

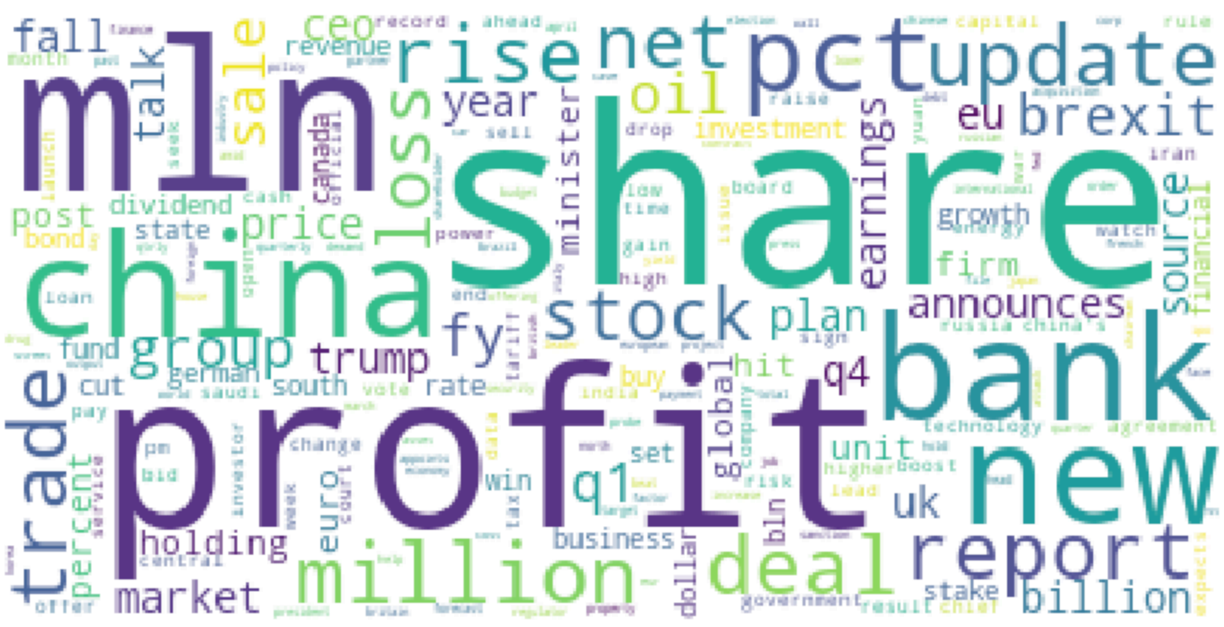
After having the Tfidf-vector we can now make use of it with different visualization packages.

To Visualization

With WordCloud

The tfidf calculation is used to exemplify the difficulty of working with text but also on the other hand the intelligent way of still giving some semantics to text as numbers. I have further used Tfidf for creating a WordCloud. This visualization technique brings forth the importance of word use in the full context of the dataset 'articles_for_prediction.csv' where the size of a term is dictated by its TF-IDF weight in a given document.

[2] https://en.wikipedia.org/wiki/Tf-idf



From the WordCloud you can see that the newspaper gives a solid amount of information about the topics ‘China’, ‘profit, ‘share’ but also about foreign politics. The extracted text is from the headlines, so we might conclude that ‘Reuters’ informs about the (many) profits gained and the losses made. The year 2018 was a good year for the economy in America and relationship with China was evidently there, and maybe even could be speak of solidarity given the word ‘share’.

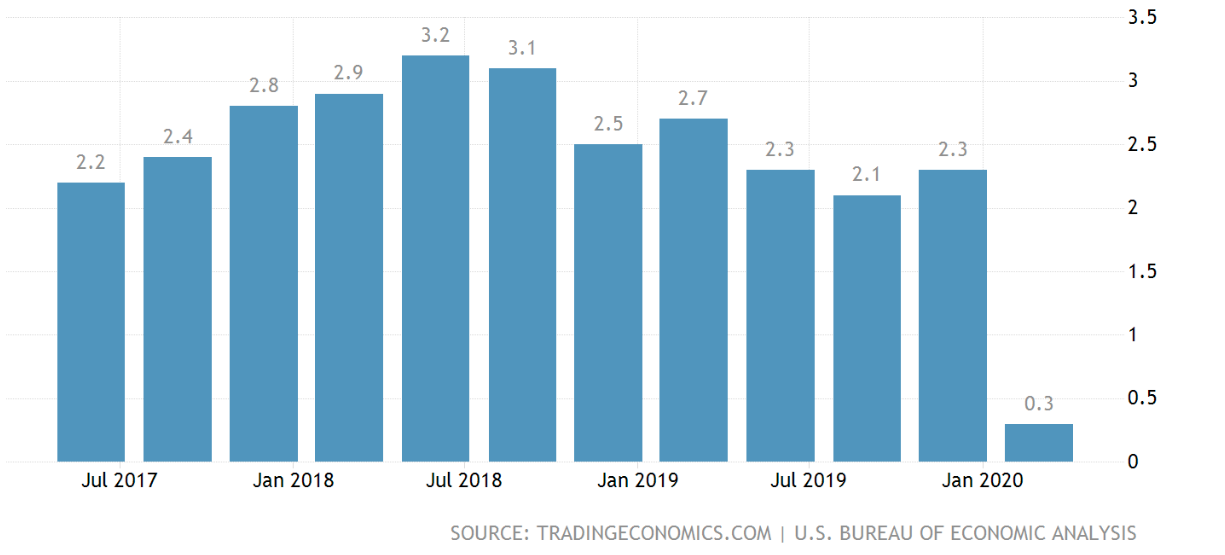


Figure 1 Annual Growth Rate in the USA[3]

[3] https://tradingeconomics.com/united-states/gdp-growth-annual

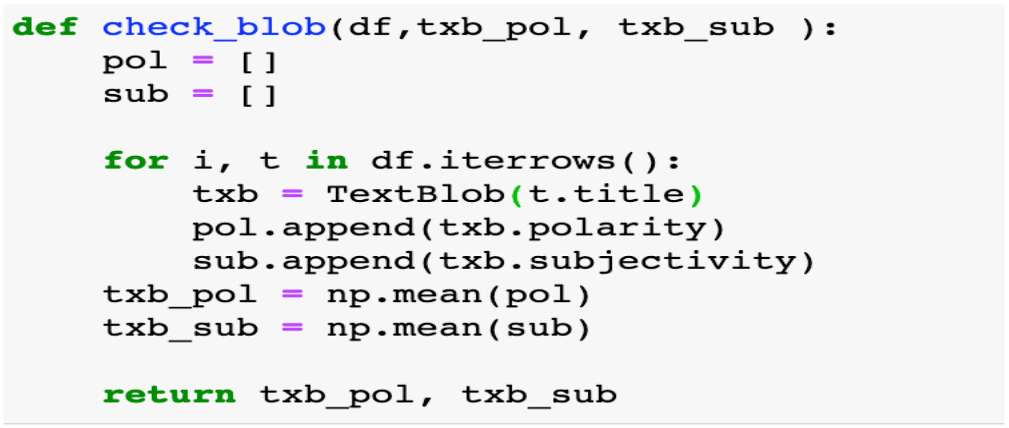
With Bokeh

Bokeh API is a Python library for interactive visualization that targets web browsers for representation. One can use the JSON format for further interaction with Bokeh and websites. It is a quick way of using data and having a clear representation of this. Through Bokeh I have used a few graphs to find out how overall the sentiment was in 2018 but also how the stock prices changed in that year.

TextBlob

For the sentiment information displayed in Bokeh, I have used TextBlob to generate a polarity and a subjectivity in the headlines. TextBlob is another library for processing text and it is built on top of 'NLTK'[4] and the 'pattern'-library. The 'polarity' attribute will be giving a float number calculated between -1 and 1, where -1 is negative and 1 is positive. In this way you can test the text on the neutrality. Eg text like 'I dislike carrots', will have a negative polarity whereas 'I love Paris' will have a more positive polarity. On the other hand, to find out more about the subjectivity of the text there is 'subjectivity' that generally refers to personal opinion, emotion or judgment (whereas objectivity refers to factual information). 'Subjectivity' in TextBlob will giving back a float which lies in the range of [0,1].

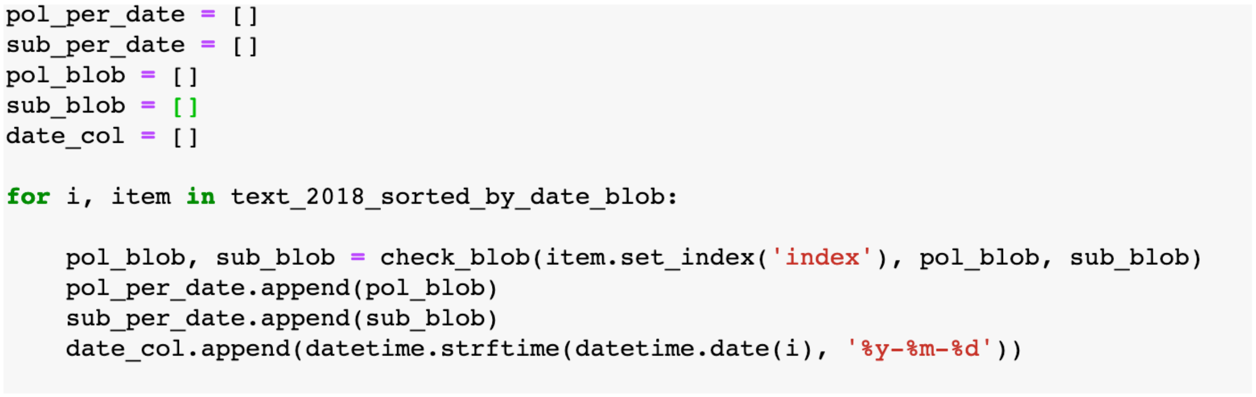
The code I have used for getting the polarity and subjectivity:



```
def check_blob(df,txb_pol, txb_sub ):
    pol = []
    sub = []

    for i, t in df.iterrows():
        txb = TextBlob(t.title)
        pol.append(txb.polarity)
        sub.append(txb.subjectivity)
    txb_pol = np.mean(pol)
    txb_sub = np.mean(sub)

    return txb_pol, txb_sub
```



```
pol_per_date = []
sub_per_date = []
pol_blob = []
sub_blob = []
date_col = []

for i, item in text_2018_sorted_by_date_blob:

    pol_blob, sub_blob = check_blob(item.set_index('index'), pol_blob, sub_blob)
    pol_per_date.append(pol_blob)
    sub_per_date.append(sub_blob)
    date_col.append(datetime.strftime(datetime.date(i), '%y-%m-%d'))
```

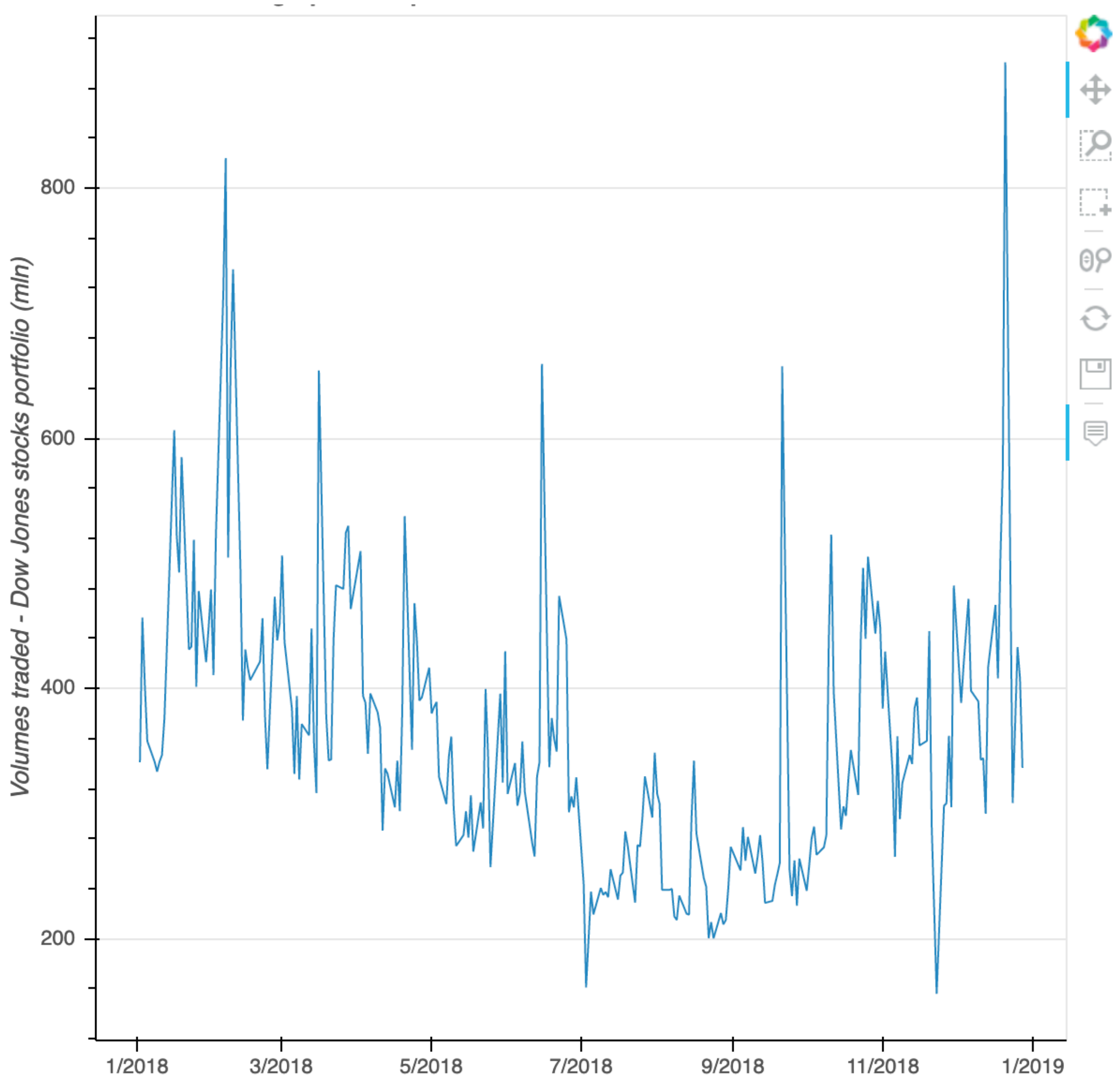
[4] http://www.nltk.org/book/

ColumnDatasource

Most of the plotting methods in Bokeh API are able to receive data source parameters through ColumnDatasource object. It makes sharing data between plots and 'DataTables' easier.
A ColumnDatasource can be considered as a mapping between column name and list of data.

One thing to know is that you cannot buy shares in the Dow Jones Industrial Average, but you can buy an exchange-traded fund that tracks the index and holds all 30 of the stocks in proportion to their weights in the DJIA.

The graph shows the volatility of the volumes traded on different dates. Especially at the end of the year the trades are in big volumes but took a big dive.

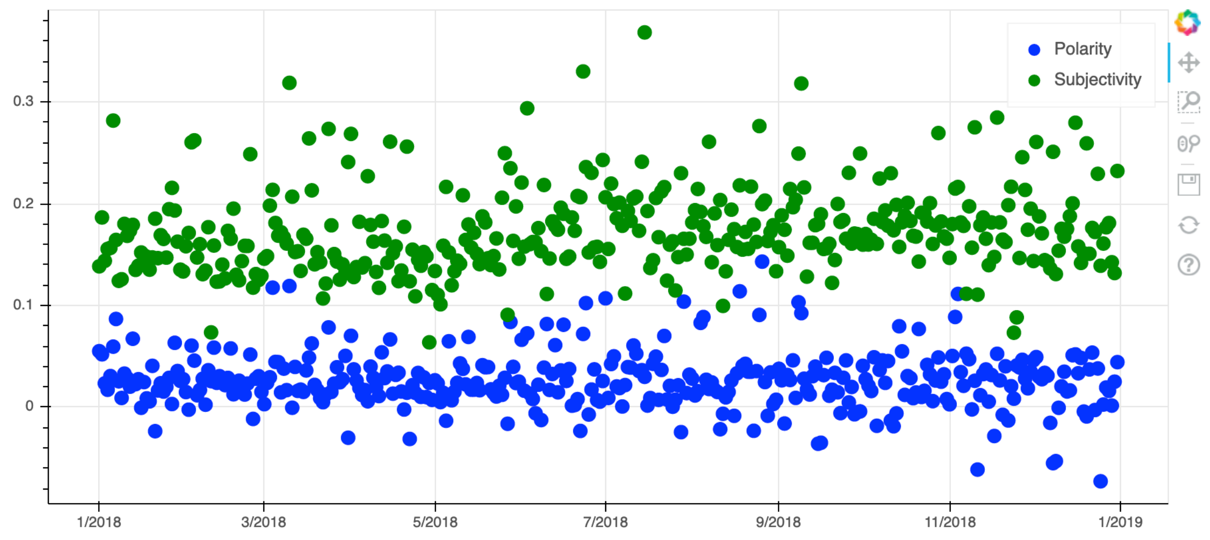


The news articles confirm it[5]:

‘The Dow Jones Industrial Average fell 414.23 points to finish at 22,445.37 in turbulent trading that sent the blue-chip index up as much as 300 points earlier in the day, only to trade back in negative territory less than one hour later. The initial rally upward on Friday came as Federal Reserve Bank of New York President John Williams told CNBC that the central bank could reassess its interest rate policy and balance sheet reduction in the new year if the economy slows.

But those gains slowly disappeared as investors used that short-term pop as a chance to sell more. The broader fell 2.1 percent on Friday to close at 2,416.58, while the tech-heavy Nasdaq Composite shed 2.99 percent to 6,332.99 with big losses in technology stocks including Facebook, Amazon and Apple.’

In the last part I wanted to look at the actual text which has been spread in the news articles. From the graph we can see that especially at the end of the year, the polarity became very negative but overall the drop in the index didn’t really affected the ‘subjectivity’ as much in a relative sense. The month ‘December’ overall seems to be an interesting month.



Next Stage

In the next stage of this Capstone, I will be able to dive deeper into the text with Neural Networks and try to make more out of the text. Neural Networks make it easier to ‘calculate’ the meaning of the text and hence make the text better understandable for the algorithms needed to predict.

[5] https://www.cnbc.com/2018/12/21/us-stocks-set-for-lower-week-after-fed-decision-government-shutdown-fears.html