Detecting Media Bias

Introduction and Objectives:

Media has always played a significant role in the public opinion and therefore the way the media mentions or portrays certain events or figures is important. Using machine learning to classify the sentiment of a particular text can be used to measure the media's support or opposition for political parties, candidates, bills or legislations. In this project we will be using NLP and machine learning techniques to create a sentiment classifier using Google Sentiment analyzer and Text Blob as benchmarks. We will also use sentiment Entity analyzer to compare the media sentiment towards the last five runners in the US 2020 democratic party elections.

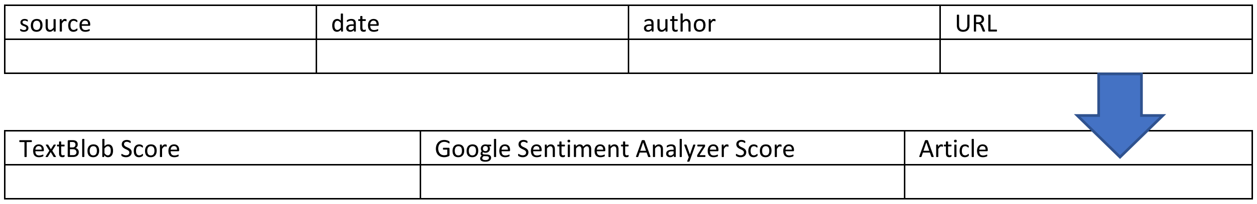
Background:

One of the major distinctions between different chat bots, search engines and recommendation systems is their level of understanding of natural language. Due to its importance and complexity, NLP is one the most researched topics in for AI and machine learning. Analyzing language and sentiment in the context of media and politics has drawn a lot of attention as it can confirm obvious trends or reveal unnoticed ones. Several researchers have tackled similar problems and many of them have used neural networks like Victor Saenger in his article 'Media Bias Detection using deep learning libraries in Python' published on towardsdatascienece.com and Martha Bellows in 'Exploration of classifying sentence bias in news articles with machine learning models'. Other less common approaches for this kind of problem are random forest and logistic regression.

Data Collection and Preprocessing:

For this capstone, the dataset was scraped using newsapi.org from twenty news sources : ('abcnews.go.com', 'apnews.com', 'bloomberg.com', 'breitbart.com', 'cbsnews.com', 'cnbc.com', 'cnn.com', 'foxnews.com', 'msnbc.com','nbcnews.com','politico.com', 'theamericanconservative.com', 'thehill.com', 'huffingtonpost.com', 'washingtonpost.com', 'time.com', 'usatoday.com/news', 'wsj.com','nymag.com').

Newsapi.org returns a json object which includes; news source (string), date (datetime), author(string), title(string), content snippet (string) and URL(string). Using the returned URL another request was initiated from the code to scrape the full article using beautiful soup.



source	date	author	URL

TextBlob Score	Google Sentiment Analyzer Score	Article

Figure 1: Data Wireframe

The scraped data in many times included some html elements and URLs referencing other websites and advertisements, all this had to be removed, also we made sure there was no duplicate rows or rows where the scraped text was null.

In order to be able to have a training dataset for any sentiment analyzing model the data has to be labelled or classified as positive sentiment, negative or neutral. Ideally to obtain the best results we should have asked multiple people to read all scraped articles and assign a sentiment score for every one of them based on a pre identified scoring strategy. However, since the dataset was large, we looked for automated alternatives. We ended up finding two feasible methods, one was using Python TextBlob library and the other one was using Google sentiment analyzer through its API. So, for each article we had two sentiment scores.

Before modelling the data we still needed to transform the article text into a numeric format, one of the most commonly used techniques to do this is TF-IDF. TF-IDF vectorizes the text after tokenizing it into unique words and giving each word a weight based on its appearance in the corpus. Having said that, we also lemmatized the words after excluding stop words, and words that appeared less than 5 times in the corpus using min_df argument. We ended up with more than 26 thousand unique words.

<u>Modelling:</u>

The sentiment score we get from TextBlob and google sentiment analyzer ranges between -1 for extremely negative sentiment to +1 for very positive sentiment passing by zero when the sentiment is perfectly neutral. These numbers might make it sound like linear regression can be a good candidate to model the data but since we were dealing with more than 26 thousand words (dimensions), calculating the accuracy using R squared or AUC would still be very hard to interpret as a small error in half of the dimensions would still accumulate to a large error. Hence we decided to turn this problem into a classification problem, so when the score is between -1 and -0.25 then the sentiment is negative, and when the score is between -0.25 and +0.25 the sentiment is neutral and if the score is above 0.25 the sentiment is positive. We decided on these ranges so that we can have more balanced distribution among the three classifications.

For modelling the classified data, neural networks, random forest and even KNN could have worked but they would have been more complex than logistic regression and will most likely consume more computational resources and time. To keep things simple and efficient logistic regression was selected. After splitting the dataset into training and test, optimizing the logistic regression hyperparameters was done using grid search.

Furthermore, we wanted to look at the media sentiment for the last five runners in the US democratic party elections. For this reason we had to use Google entity sentiment analyzer which for each article will return the entities mentioned and the sentiment for them. We filtered the response so we will only look at the entity if it was referring to Joe Biden, Bernie Sanders, Elizabeth Warren, Pete Buttigieg or Michael Bloomberg.

<u>Findings:</u>

The optimized logistic regression model, performed well, see table (1). The accuracy was high over both TextBlob and Google sentiment analyzer scores. Nonetheless the precision and recall for google were a bit low mainly because most of the data lies between -0.25 and 0.25 (data imbalance)

Target	**Regularization**	**Accuracy**	**Precision**	**Recall**
TextBlob	L2, C=10	0.87	0.86	0.70
Google Sentiment Analyzer	L2, C=10	0.90	0.61	0.56

Table 1:Model Evaluation Scores

Also in the code we were able to plot the words that contributed the most to the articles low or high score using WordCloud library and the model's coefficients. Figure (2) is for the words with the strongest impact on the TextBlob score as per our logistic regression model.

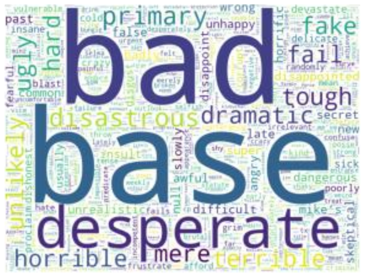


Figure 2: Words With The Highest Impact On the TextBlob Score

With respect to the entity sentiment analyzer, we can see in figure (1) below which was generated using Tableau, that most news outlets mentioned Joe Biden more than the other candidates. Moreover, the average sentiment score for the last three the candidates was negative but the lowest average score was for Bernie Sanders and his average score is significantly below all his rivals.

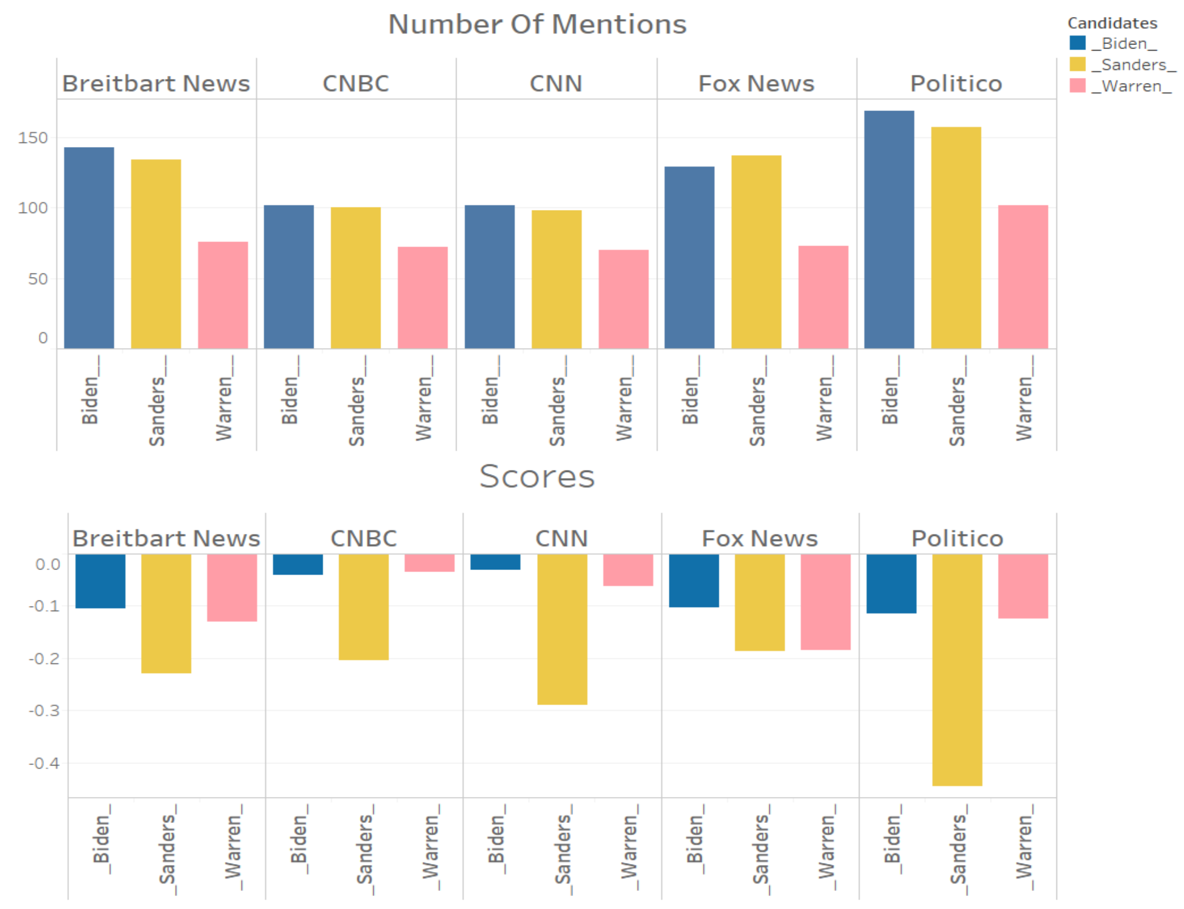


Figure 3:Some Channels Scores Results Filtered On The Last 3 Runners

<u>Future plan:</u>

This project can be ameliorated in many ways. But what we are aiming to do in the short term is to try word2vec for the text vectorization and see how much this can enhance the results. The other thing is trying neural networks for the predictions as neural networks can predict two outputs (TextBlob and google analyzer scores) both at the same time.

We think that this project can have some real applications built on it as it can be used by search engines when people are searching for how news were reported by 'biased' or 'objective' news outlet (i.e : search string=the democratic debate analysis by Warren's supporters). Additionally, with continuous improvements we can create a bias index for the different news outlets.