# Headline Text Mining Project

Gregory Sylvester, *Undergraduate*, CSI - 5810

*Abstract*

*This project tasks us to perform a form of data analysis to collect some useful information for the data. In my case my project involves collection of CNN headlines[1] and predicting which categories they belong to, perform sentiment analysis, and clustering. To perform all these operations the headlines, need to be cleaned and vectorized. After all this has been done, we can look at the information extracted to help us with which articles we may want to read.*

## I. Introduction & Domain

Understanding which articles, you may or may not want to read helps to critically reduce time and resources. CNN provides news articles on the most popular and prevalent information. They host a variety of categories for news articles. Depending on the users wants they may be a category of news stories that is not listed. Clustering helps to show subcategories or types of news stories that are prevalent in the news cycle.

The sentiment of the stories is also important depending on if you want to read about something that may be negative, positive, or neutral[2].

All of this processing should help expedite the process of receiving news and the types of news stories you would receive.

## II. Problem Knowledge & Overview

To perform these three major tasks there are a few things that need to be done before we can start these tasks (sentiment analysis, classification, and clustering).

The first major step is data collection, which will require data scraping. We will be pulling the headlines from the CNN webpage and categories.

The next step is preprocessing[3], which will entail data cleaning, lemmatization, vectorization, and understating the spread of information with some visuals.

We can now perform classification once all the busy work has been handled. For classification we use 4 different models with some tuning of the parameters.

We can also perform clustering on the data to help us identify patterns in the headlines. This will help to reveal prominent events/news that is receiving heavy amounts of reporting.

Finally, we can also perform sentiment analysis. To do this all we need to do is look at the polarity of words in headlines.

## III. Data Collection

For the project we need to collect a good number of headlines from a popular news distributor. CNN provides an API to gather their news, however you need to obtain permission. Therefore, I just used Beautiful Soup[4], which is a free to use data scaping library for python which is relatively easy to use for beginners.

To get the headlines from the website you need to feed BeautifulSoup the URL's and do a little bit of html searching to find your target data. Luckily the tags of classes I used for my classification tasks were also easy to find. Since each of the 10 categories has a separate but nearly identical webpage no

processing was required to find the true tags for class attribute.

## IV. Preprocess

### A. Overview

Looking over the data we have 10 categories and 658 headlines in total.

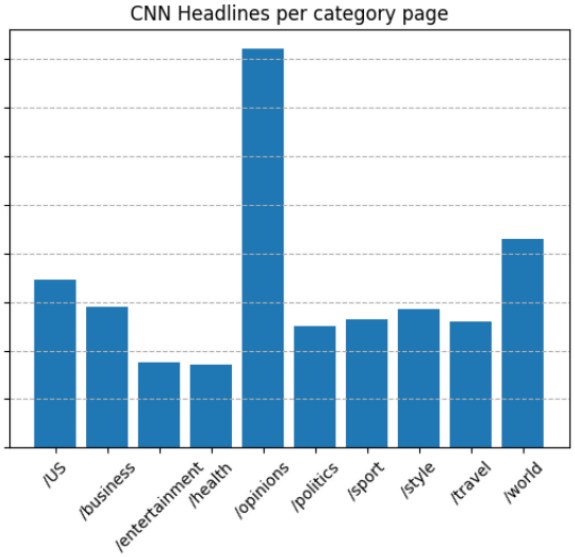


*Fig. 1 Categories of CNN headlines*

To process this data, we need to remove unwanted characters, tokenized, lowercase, lemmatize, stop words, and vectorize. After this is performed, we will have the necessary data to move forward. This is where an error was made, subject matter stop words (words like news) were removed after the fact for clustering later in the project. The primary library used was spacy[5].

### B. Characters, Lowercase, & Tokenization

To properly clean the headlines, they need to be in the form of readable word tokens. For this step removing characters not in the alphabet, such as percentage and dollar sign symbols, will reduce confusion.

The words now need to be separated to be processed individually and if at any point can be remerged to pull the processed headline out. To do this we simply just perform tokenization. We also quickly lowercase all characters.

### C. Lemmatize

Lemmatization is the action of taking a word and reducing it to the more basic form of word. An example would be ‘woods’ becoming ‘wood’. This helps any processing down to be able to allow similar tokens to be processed as the same but different instance of the token.

### D. Stop Words

It is important to remove words that are extremely frequent and don’t usually provide any relevant information. The best example would be the word ‘the’. This is where a clear mistake was made. Stop words can also be prevalent to the topic not just to general text.

### E. Vectorization

The last important step is to transform these words into numbers for these series of tasks which will be performed. TF-IDF is the method of vectorization which was undertaken to represent the tokens.

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad \mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

*Equation. 1 TF-IDF pulled from Wikipedia[6]*

TF-IDF means term frequency and inverse document frequency. This value is measured in the following equation.

## V. Classification

### A. Overview

Given the data and labels we want to build a model that can predict which of the ten categories a news article will belong to. The data was split .7 for the training set, and .3 went to the test set. The true labels were found by which webpage the article was pulled from.

### B. Models

The models built were Multinomial Naïve bayes, Decision Tree classifier, Random Forest, Bernoulli Naïve Bayes, and Ensemble. The metrics used to measure the output were macro averages of the precision and recall. We want to find a model that strikes a good amount of precision and recall for all classes. We also use the vectors produced during the preprocessing stage (TF-IDF vectors).

### C. Result

None of the models were able to achieve a high overall precision while retaining recall.

| *Model* | *Precision* | *Recall* |
|---|---|---|
| *Multi* | .27 | .25 |
| *Bernoulli* | .34 | .23 |
| *Forest* | .29 | .21 |
| *Tree* | .15 | .14 |
| *Ensemble* | .30 | .18 |

*Fig. 2 Macro Averages of Precision and Recall of models.*

The main issue that would be the fault is the low amount of training data. Another fault is the number of classes each classifier is trying to sort into 10 categories. It should be noted that if this project is continued then a multilevel classifier will be used to help resolve this problem.

## VI. Clustering

### A. Overview

The purpose of clustering the data is to find patterns (information) about the number of articles and types published. Seeing which types of clusters emerge allows us to see the importance of different topics in the news cycle.

To perform clustering, I will be using the same vectors from earlier and use K-means with 5 clusters and a L2 (Euclidean distance) metric to find the difference between vectors.

### B. Results

Once the clusters have been made, we need to be able to pull useful information from them. To do this we can look at the most frequent words in each category.

This is where being able to remove subject matter stop words became important. At first all the clusters appeared to be the same with prevalent words like “new” and “kill” appearing as the top two words in two separate clusters. The quick fix was to remove the top five words which solved the issue.

### C. Information insight

The new top words in each category as well as other words we know allows us to build a idea of the prevalent topics.

| *Cluster number* | *First* | *Second* | *Third* |
|---|---|---|---|
| *1* | Israel | Hama | Know |
| *2* | Trump | Putin | Reveal |
| *3* | Kill | New | World |
| *4* | Life | Save | Change |
| *5* | Year | Record | World |

*Fig. 3 Top 3 words of each cluster.*

Looking at the cluster's top words, cluster one you can tell is about the Israel-Hamas crisis. Cluster five developed around end of the year events (since at this moment it is December of 2023). Cluster 2 is a little more difficult but appears to be about general politics. Cluster 4 and 3 don't have any good generalizable information that can be pulled. Viewing these clusters allows us to develop which news topics dominate the headlines.

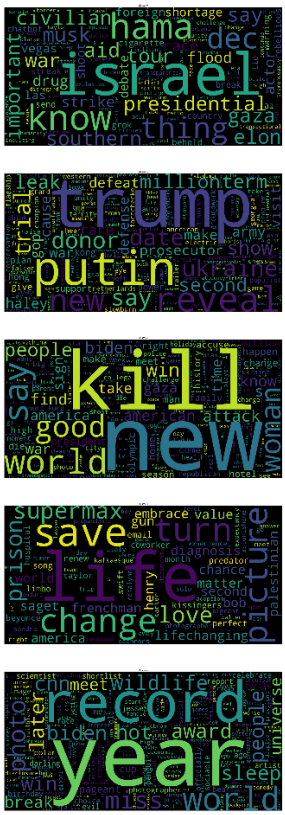


*Fig. 4 Word clouds for cluster 1 – 5.*

The generated word clouds[7] also help us understand the information we can pull from the generated clusters of information.

## VII. Sentiment Analysis

### A. Overview

Performing Sentiment Analysis will allow us to gage the general attitude in the news right now. Allowing us to tell the quantities of 'good' and 'bad' news articles.

To view the overall quality of news we will be using TextBlob[8] which comes with a prebuilt polarity function. This function can gage the polarity of text like "good" or "bad" giving a scale from -1 to 1 for the polarity of the text. Feeding the headlines, we can gage the general polarity of articles.

### B. Results

Using this we can lookup whether any induvial article headline is classified as positive, negative, or neutral.

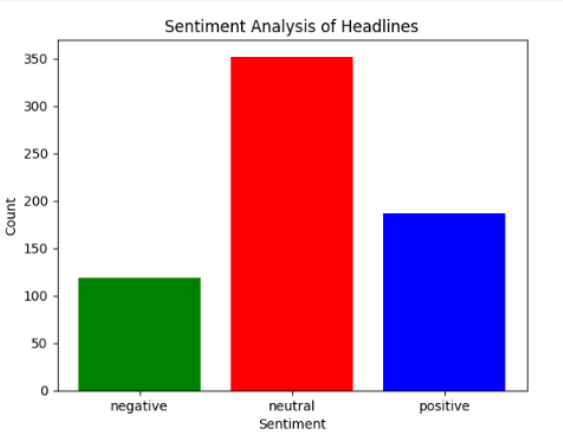


*Fig. 5 Bar Graph of Sentiments of headlines.*

## VIII. Errors

The biggest issue that I was faced with was my classification models not being able to produce good results. If this project is

continued at any point the solution is to build a multilevel classifier to reduce the amount of work each level of the classifier need to do.

The next big issue which faced me was when building the clustering, subject matter stop words weren't removed in the preprocessing phase. The stopgap for this issue was to remove the most frequently used words in all the clusters.

## IX. References

Google colab is where all the source code: https://colab.research.google.com/drive/1-WvVg0TXgf6jSgdP_z3p1P-_ygbn2nXS#scrollTo=LeJPprkNv1Q9

[1]https://www.cnn.com/

[2]https://en.wikipedia.org/wiki/Sentiment_analysi

[3]https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/

[4]https://www.youtube.com/watch?v=bargNl2WeN4

[5]https://www.youtube.com/watch?v=dIUTsFT2MeQ

[6]https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[7]https://www.youtube.com/watch?v=HcKUU5nNmrs

[8] https://textblob.readthedocs.io/en/dev/