Article Classifier

Miguel Gutierrez | Yue Shen | Ana Ysasi

Agenda

- — Overview
- — Data Collection
- — Data Cleaning and Preparation
- — Matching Articles
- — Article Classifier
- — Example Output
- — Results

Overview

How can we classify articles based on their political leaning?

We wanted to create an algorithm that could compare the way that two different news sites report political news and quantify their bias.

Data Collection

Gathering Articles

We used RSS feeds to obtain URLs of different news articles in near realtime as they are published. A script was run every 10 minutes to acquire these URLs.

Scraping Articles

Initially, we used the newspaper3k package in Python to obtain the title and text of articles from their URL. The package is versatile and allowed us to easily scrape multiple news sources.

Storing Articles

We set up a MongoDB database that was integrated into the scraper to automatically upload the article and its properties. The current database stands at over 15000 articles.

Data Cleaning and Preparation

Ad removal

The web scraped articles had advertisements or specific formatting from the original websites, to not alter the meaning we had to remove them.

Stopwords

Stopwords are commonly used words in the English language. In order to stop them from creating extraneous features in our model, they needed to be removed.

Lemmatizing and Tokenizing Articles

In order to make a sensitive analysis, we need to lemmatize the text, i.e. get the root forms of the words (verbs in present, nouns in singular...). We also need to tokenize each article, i.e. break it by words.

Specific words

After lemmatizing and extracting all the symbols, some words lost their meaning, we had to make specific transformations for this not to happen.

Article Matching

We used Gensim to apply *Latent Semantic Indexing (LSI)* on all of the CNN and Fox News articles in our database.

LSI implements Singular Value Decomposition to group together documents in vector space by topics. Our model uses 300 as the number of topics for each dataset.

The collections of articles are then transformed to LSI space and indexed. The query article can then be vectorized and compared to the index using cosine similarity.

Article Classifier

1 Article Selection and Test Train Split

We used only political articles based mainly on the US for the model. Two news sites were selected to train – Fox News and CNN.
To validate our model and calculate accuracy, a 70/30 train-test split was used.

2 Tagging the Articles

We tagged Fox News as right-leaning and CNN as left-leaning to train the model. Ideally, our next steps moving forward would be to train the model on additional news sites.

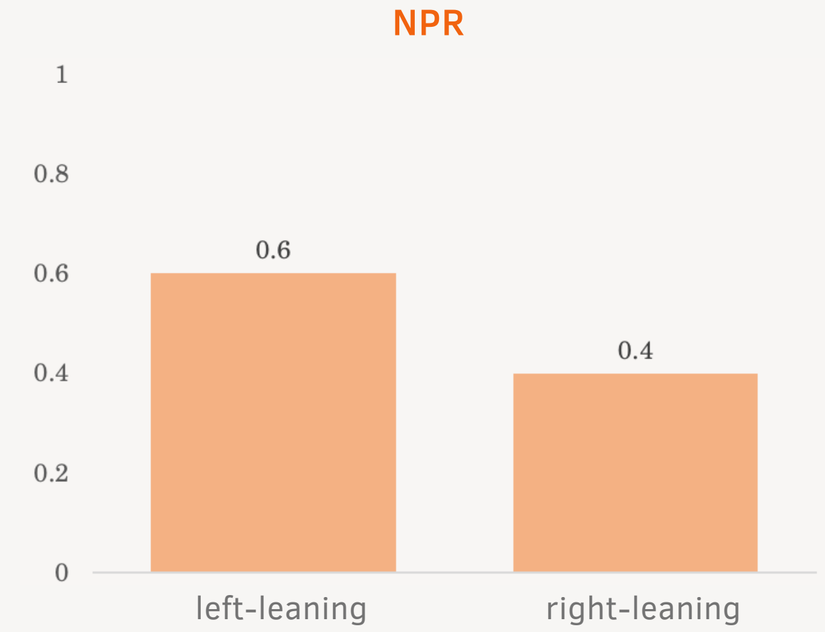
3 The Model

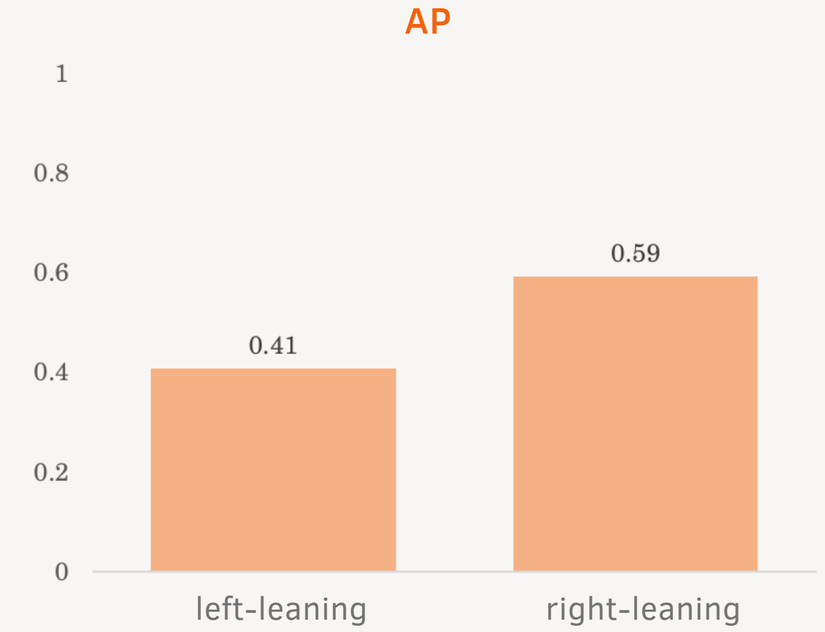
A Naive Bayes Classifier was used to classify the articles. As this is a probabilistic model, our output contained the probabilities of the article belonging to each class.

Results

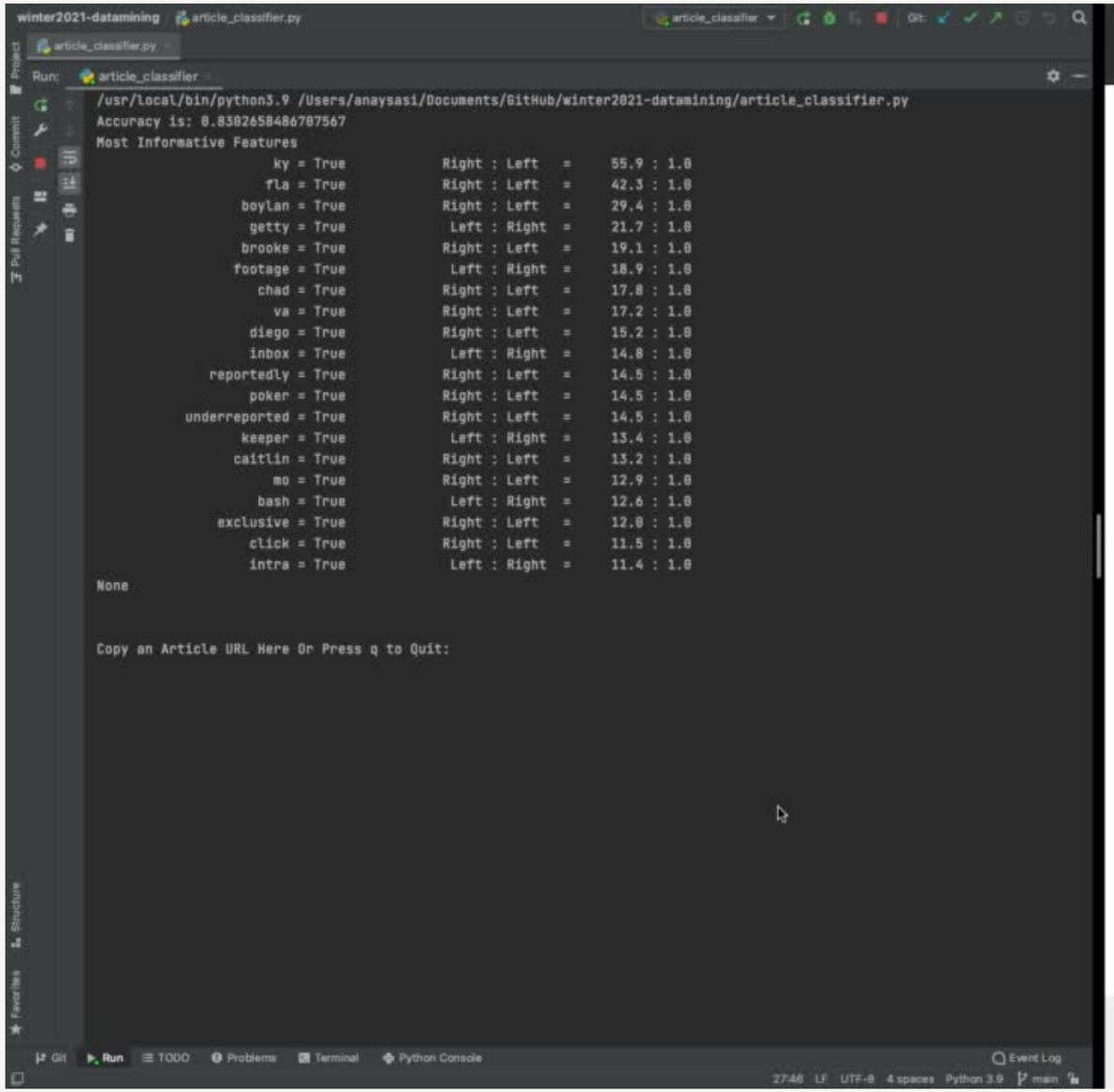
Our Article Classifier had an 83% accuracy in predicting samples in the holdout set with the current articles in the database.

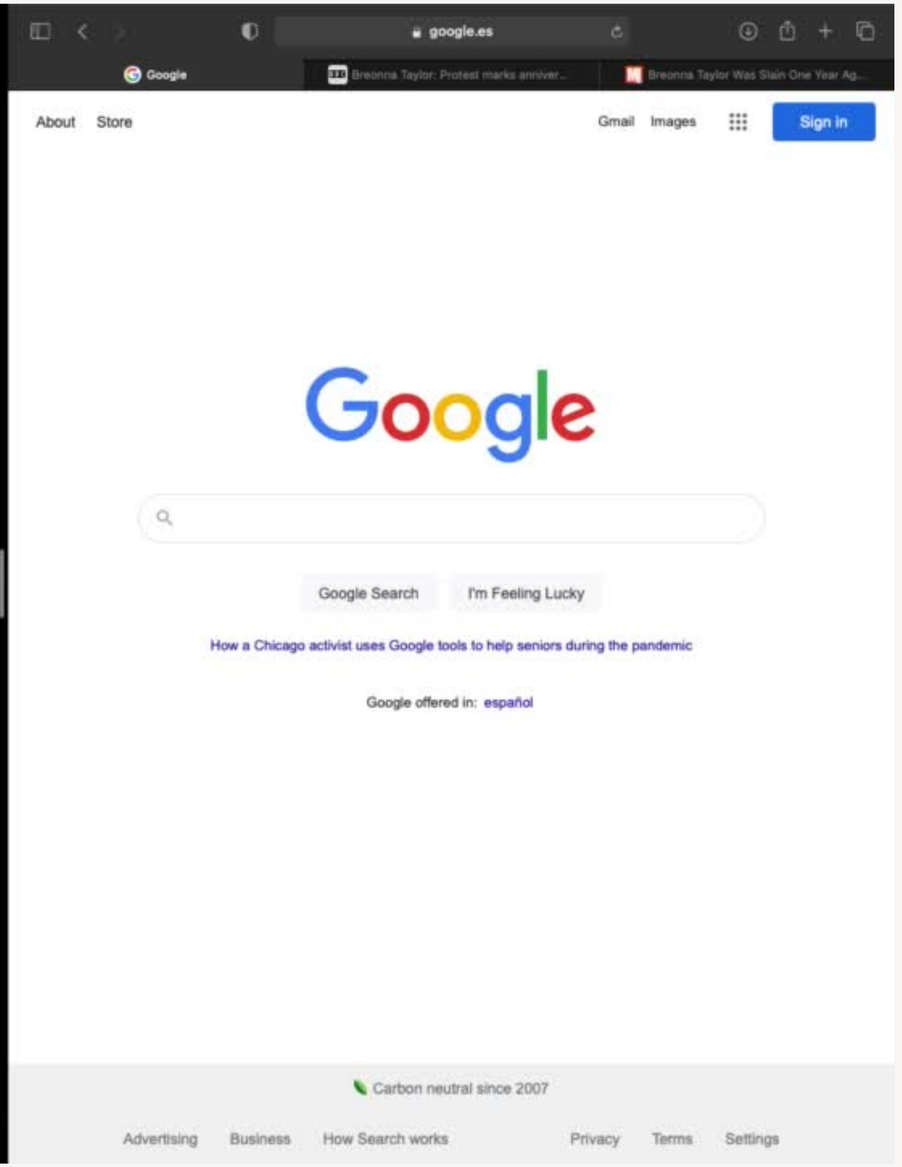
Extremity score for NPR: 0.929
Extremity score for AP: 0.931





Example Output





Example Output

```
Copy an Article URL Here Or Press q to Quit: https://www.bbc.com/news/world-us-canada-56387446
The article is predicted as:  Left
Probability of being a left-leaning article: 0.5173
Probability of being a right-leaning article: 0.4827
Extremity score:  0.0346

Matched CNN Article:
'Go back to where you came from': Cricket fan details allegations of racial abuse by staff and supporters
Cosine Similarity: 0.5379

Matched FOX Article:
South Carolina adds firing squad to list of execution methods
Cosine Similarity: 0.5182
```

Thanks for listening!

If you would like to take a look at the code to try it yourself, give us feedback, or improve upon it, check out our GitHub!