Classifying Democratic/Republican Political Speeches

Jose Miguel Montoro

Motivation

- The goal is to work on an NLP problem (text classification) from the beginning to end of the pipeline.
- It will cover all the NLP steps:
  - Obtaining the data
  - Data cleaning and text pre-processing
  - Vectorization of text
  - Text classification with ML models
  - Text classification with deep learning models

Obtaining the Data

- The data needed for training and testing the classification models are political speeches classified by affiliation (Republican/Democratic)
- I’ve obtained the current data from the following website: millercenter.org/the-presidency/presidential-speeches
- The main tool used for obtaining the data was **Selenium.** It allows for dynamically headless browser interaction. BeautifulSoup wasn’t sufficient in my case because the website has some dynamic JavaScript object the user has to interact with in order to display the speeches (see example in next slide).

Obtaining the Data

Text Pre-Processing

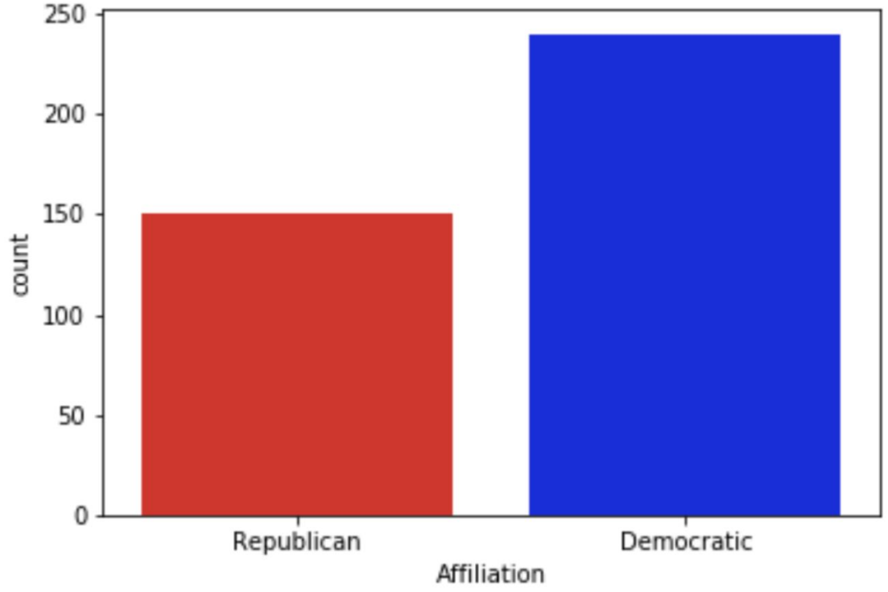
- One of the most important steps in NLP is text pre-processing. The following steps were performed, through a combination of custom functions and the text cleaning methods in SpaCy:
  - Expand contractions (“don’t” > “do not”) - custom function
  - Remove non-word characters - custom function
  - Lemmatize the text - SpaCy
  - Remove stop words - SpaCy
  - Remove non-ascii characters - SpaCy
  - Consolidate spaces - custom function

Exploratory Data Analysis

- The EDA section of the project is not as extensive as other projects, because the data is text. However, some summary statistics are presented in charts here.

Exploratory Data Analysis

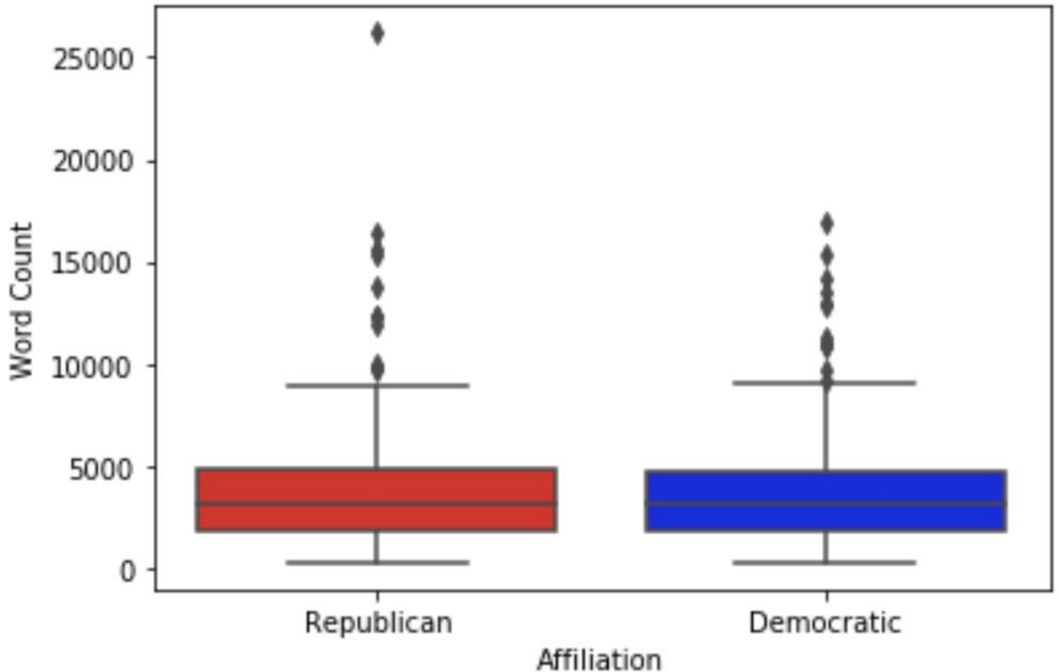
- There are a more Democratic than Republican speeches



Count of speeches by Affiliation

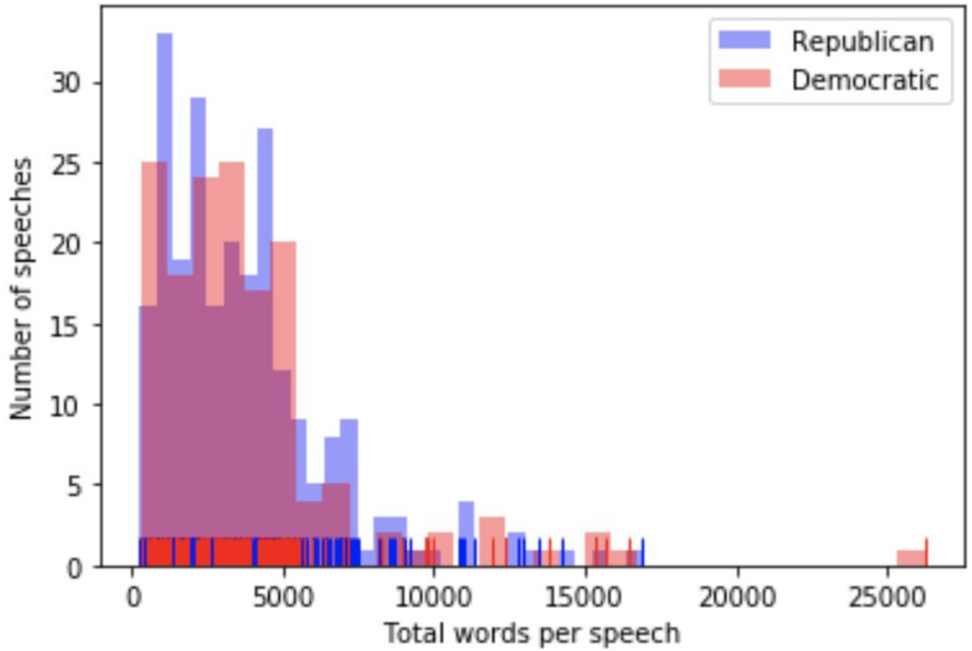
Exploratory Data Analysis

- Very similar distribution (except for one Republican outlier - one speech by Trump on Coronavirus)



Exploratory Data Analysis

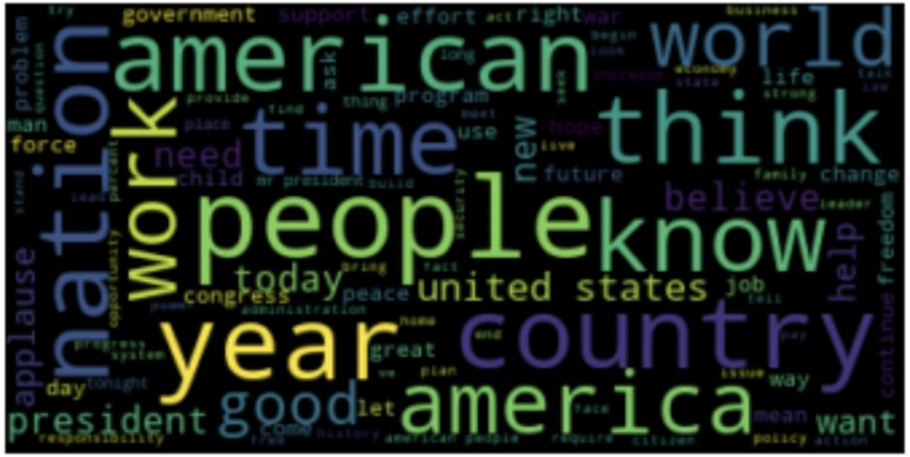
- Simiar distribution of the Number of words per speech



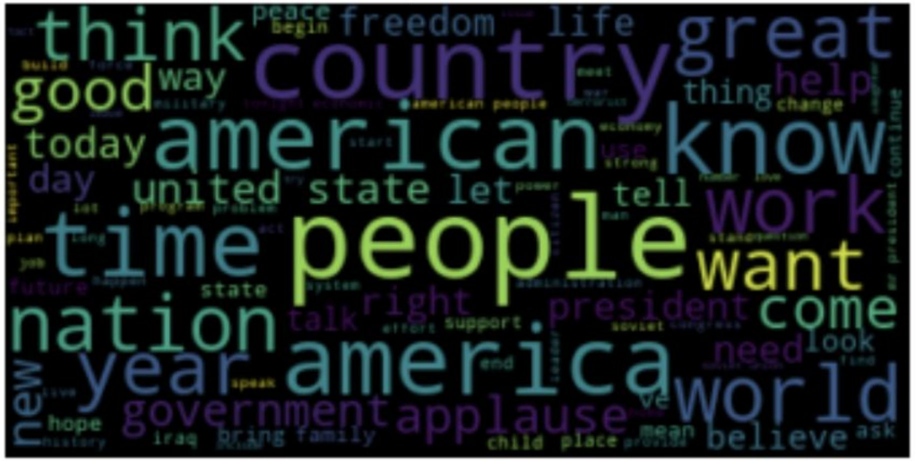
Exploratory Data Analysis

- Word Clouds. The msot common words are very similar in both cases.

Democratic



Republican



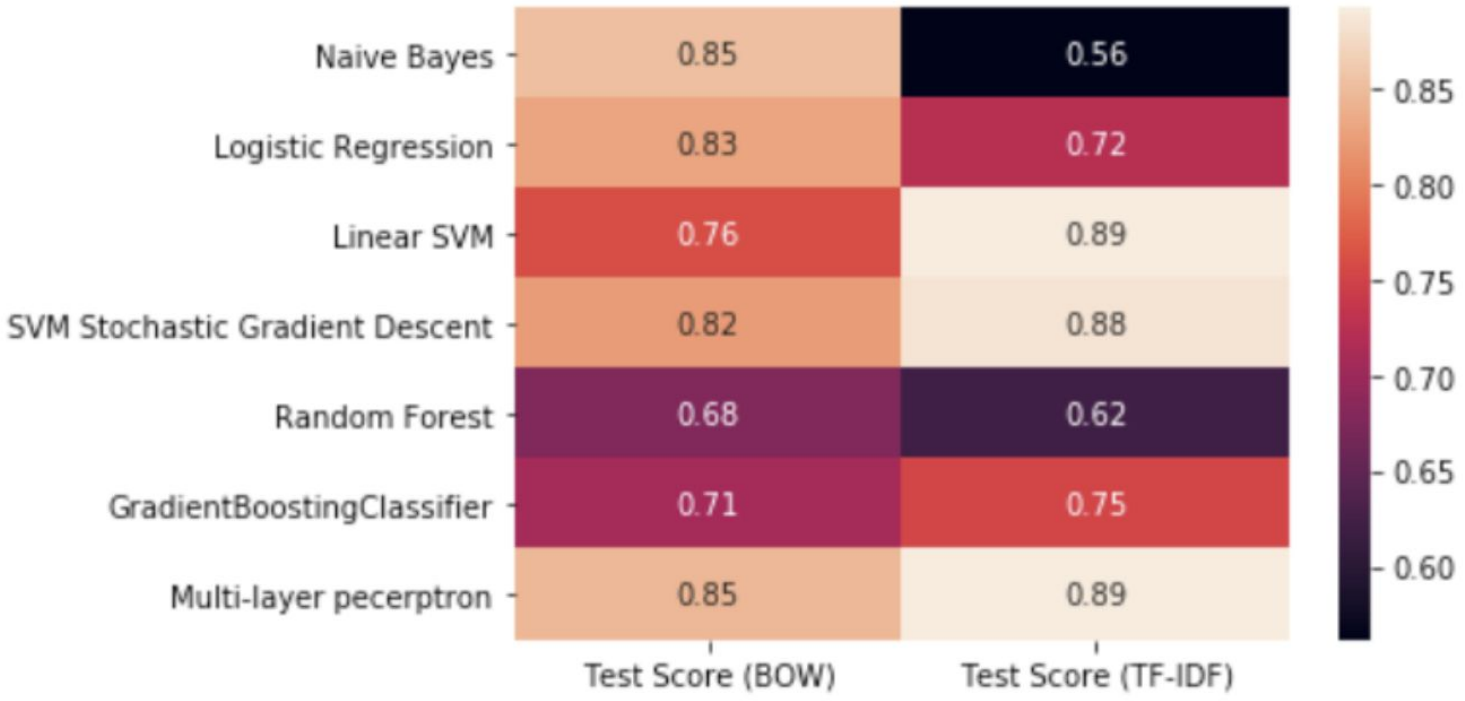
Vector Representation of Text

- The next step was converting text to numeric values that can be used by machine learning algorithms to predict the text's class. This process is called "vectorization".
- Two ML approaches were tested:
  - Bag Of Words: It represents each text document as a numeric vector where each dimension is a specific word from the corpus and the value is its frequency in the document.
  - TF-IDF: The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.
- In general, TF-IDF offered better performance when applied to ML classification models.

ML Classification Algorithms

- The following 6 standard Machine Learning classification algorithms were applied:
  - Naive Bayes
  - Logistic Regression
  - Linear Support Vector Machines
  - SVM Stochastic Gradient Descent
  - Random Forest
  - Gradient Boosted Machines
- Additionally, a Multi-layer Perceptron neural network was trained with the same vectors as well.
- See results in the next slide

ML Classification Algorithms

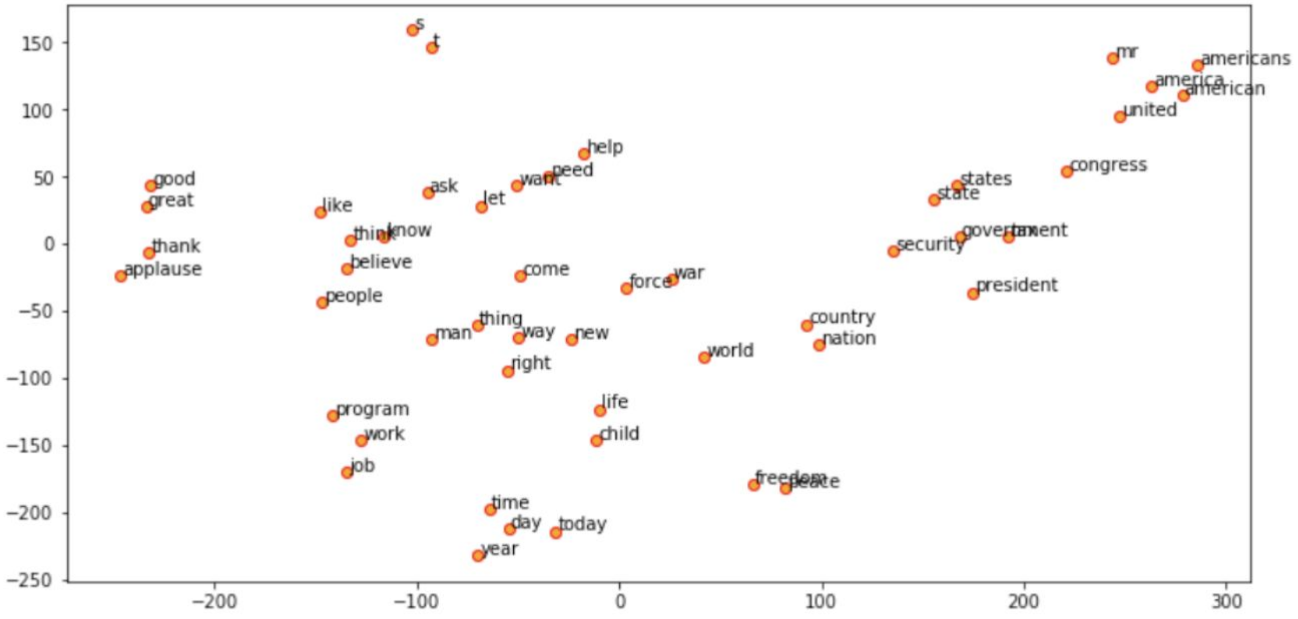


ML Classification Algorithms

- In general, the highest scores were obtained with the TF-IDF approach, but BOW offers a pretty good, stable baseline.
- The best performing models were *Linear Support Vector Machines,* and the *MLP neural network.* Both achieved an accuracy of 89% with the TF-IDF vectorization.
- Applying GridSearch to the SVM model didn’t improve its performance.

Deep Learning Approaches (Word Embeddings)

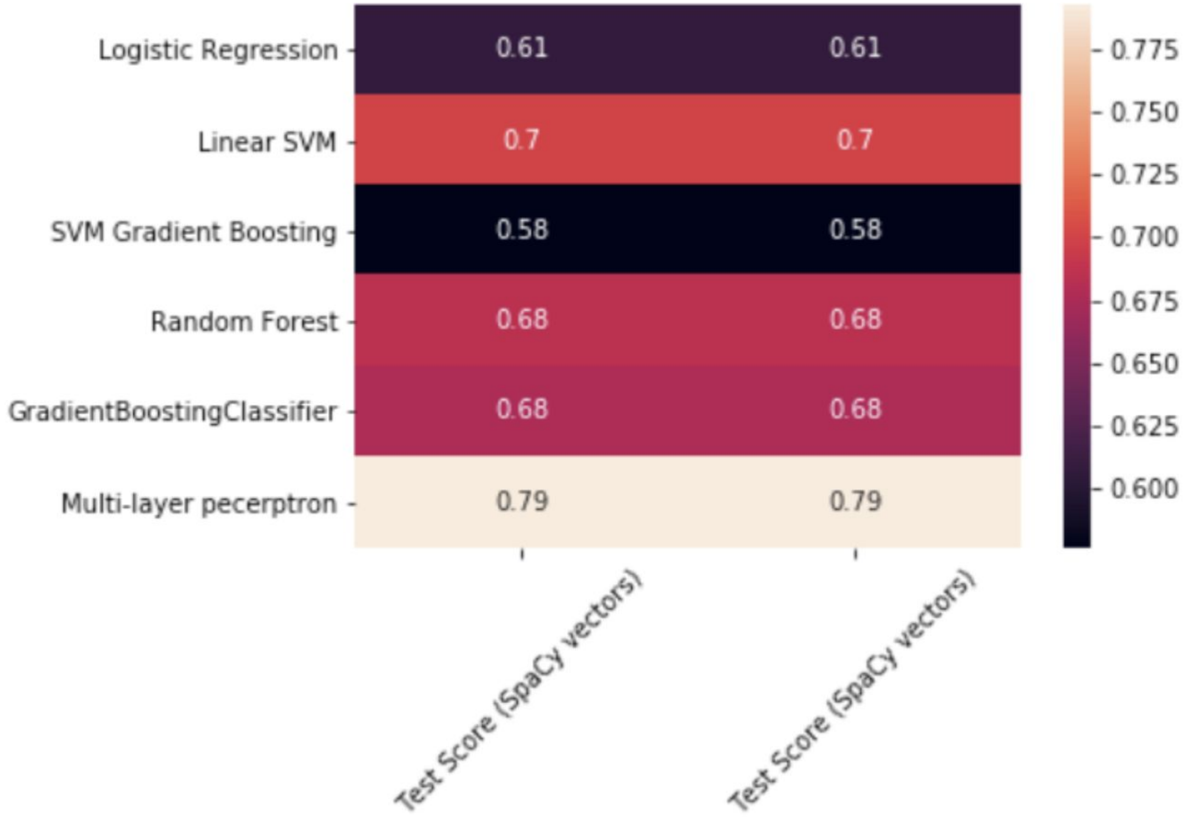
- In this project, I used SpaCy's pre-trained word embeddings. Here is a TSNE representation of the most common words in the speeches.



TSNE representation of the most common words in the corpus of political speeches. It's clear that similar words are found in an approximately similar vector space.

Deep Learning Approaches (Word Embeddings)

- The same 7 classification methods were applied to the pre-trained vectors.



Word Embeddings + Neural Networks

- Finally, a combination of pre-trained vectors from FastText (Facebook) and a Convolutional Neural Network was implemented.
- The results were not great. This is most likely due to the fact that, again, the amount of speeches we have is not too big, and deep learning approaches, in general, need much more data than traditional Machine Learning approaches.
- With no validation dataset, the accuracy was around 75%, not great compared to the other models and considering the requirementes needed for training.

Conclusion

- It was fascinating to discover how many approaches can be taken to classify text and how many techniques are available.
- When working with text, the aspects that make the most difference in the final result (other than choosing the right classification model) are:
  - Having the right data and the right amount of it
  - Pre-process your data in the most efficient way possible
  - Choose the right vectorization approach (feature extraction)

Next Steps

- **Ingest more data**. This is the one addition that would definitely impact the model the most.
- **Train our own word vectors.** Training our own word vectors could potentially improve the accuracy of the end classification models.
- **Fine-tune the neural networks' hyperparameters**. Both the MLP and CNN networks can be tweaked to try to improve their results.
- **Host the best performing model as a web application and continue ingesting user-generated data**. A reinforcing learning loop can be set up to improve the model.
- **Topic modeling.** Additionally, doing topic modeling on the data can provide additional insights on the speeches themselves and offer insights on how each category (Democratic/Republican) is different from each other.

🙏 Thank you!

Jose Miguel Montoro Costela
linkedin.com/in/josemontoro/
@JMontoro3
@JoseMMontoro
jose@josemiguelmontoro.com