Tokenizing Political Language to Determine Bias

ECG564 Fall 2022 Project Submission

Keenan Smith

12/13/22

Abstract

This project analyzes the corpus of 4 political think-tanks/new sources to classify them into either left-wing or right-wing categories. Four candidate models were chosen and analyzed for accuracy. The data was tokenized and lemmatized using the NLTK library in Python. Each model was optimized using a randomized grid search with cross validated for computational efficiency. Each model was built and trained of 80% of the total corpus. Each model was initiated using the Scikit-Learn library in Python. Each model was optimized using hyperparameters pertinent to their construction and theory. The model which shown the greatest accuracy, was an optimized Random Forest model which had an accuracy of 87%. However, the other 3 models (LASSO Logistic Regression, Linear SVM, and k-Nearest Neighbors) all had an accuracy of 86%. This is a promising result for providing evidence that political speech tokens can show bias.

1 Introduction

We currently reside in a time period that is politically charged (Kteily and Finkel 2022). It is a time where political norms that have been established since at least the 1990s if not earlier are currently in flux. This leads to questions within society of why this moment has arisen and what can be done about it to ease tensions or to incite further change. This is also happening at a time where we are all increasingly connected and data is being created in record numbers. According to the site TechJury, in 2021, "people created 2.5 Quintilian bytes of data every day" (Bulao 2022).

So where does it leave this project? My goal is not to solve the politically charged moment with data, but instead to add another tool to enhance the debate. The way we use natural language is and has been an incredibly interesting area of study for most of the modern world. It is all around us in our modern world. If you use a virtual assistant, such as Alexa or Google, you are benefiting from speech recognition software and natural language processing (NLP). If you have ever written an email or wrote a text message and used predictive text to help compose your message, you have benefited from NLP systems and data.

This project is the first step in the analysis of written political language and whether the words that author's use when writing political speech is indicative of their political bias. This project itself seeks to answer the question of whether written language and the words used within that language can accurately classify text correctly based on their political leanings.

1.1 Background

This question came to me over the summer of 2022 when the Supreme Court was set to overturn the 50+ year precedent of Roe V. Wade. I had known that the Supreme Court was anything but a political entity (Zeitz 2022) (Breiner 2021) , but I had not really looked too much into the court in my past research. I was always interested in the Legislative or Executive branch. However, with the recent news that Roe v Wade was overturned (Delaney 2022) I wanted to understand a

bit more about this third branch of government.

The latest legal theory that has some considerable buy-in from the current court is a legal theory called textualism, which "is a mode of legal interpretation that focuses on the plain meaning of the text of a legal document." (Annotated 2022) and another legal theory called originalism which is "the idea that the meaning of each provision of the United States Constitution becomes fixed at the time of its enactment." (Litman 2021). I will not give an in-depth history and background here of these two theories, but they do set up my hypothesis that I wanted to test.

1.2 The Research Question

Now, this is a complicated question, but my overall question and overall aim is to eventually determine, using NLP techniques, if Supreme Court opinions can be classified by using a political sentiment dictionary. For this project, I narrowed the scope to asking the question: **Given 4 sources, 2 Right-wing and 2 Left-wing, and a corpus of articles written on their websites, can you use the language to accurately classify each document as right-wing or left-wing.**

I believe that this is an important question for the time we are currently living. I think it is something that many people have an opinion on if they are interested or even not interested in politics. I think that most people would have some opinion that the words people use give some insight into what political leaning that person may be. I also think that this is shown in political punditry as well as political comedy. There are numerous "memes" on the internet making fun of "insert stereotypical political stance here." What I wanted to do was to test this hypothesis. To actually look at the data and see if there is a pattern to which we use speech politically and if there was a corpus that could be defined as right-wing words or left-wing words.

2 Methods

2.1 Data Collection

For this project, I will be using a web-scraped data set of article text from four sources.

- Brookings Institute: A nonprofit think-tank based out of DC. They are considered to be a leading progressive think-tank.
- Jacobin Mag: A for profit left-wing news source.
- Heritage Foundation: A nonprofit think-tank based out of DC. They are considered to be a leading conservative think-tank
- The American Mind publication: A magazine distributed and funded by the Claremont Institute, which is a nonprofit think-tank based out of Upland, CA. They are considered to be a leading conservative think-tank.

These sources were selected using a ranking of influential political think-tanks (Barham and Barham 2021) as well as personal interpretation. For instance, Jacobin was included since they represent a truly left-wing stance on American politics compared to the standard news sources.

These data were already collected prior to this project over the course of the summer. It was collected using the programming language R, and primarily using the `rvest` package, which is based off the popular `beautifulsoup` library in Python.

The full corpus constitutes a collection of 21,756 articles spanning these sources. There are 3,110 Brookings Institute articles. There are 7,205 Jacobin articles. There are 10,582 Heritage Foundation articles. There are 860 American Mind articles. There are a total of 10,315 left-wing and 11442 right-wing articles. You will notice that if you add these articles up, there is one missing. It was written in a foreign language.

2.1.1 Data Limitations

These data were collected first using the LinkChecker api to collect the sitemap link data. They were then cleaned and distilled down to articles with text to be scraped. In addition to the text of the article being collected, the title, date, author(s), and, categories (as defined by the publication) were collected. These do not factor into the modeling analysis.

It is important to note, that though the date of the article was collected, the articles were not filtered based on date. It is known that language and it's use does change over time. Since the scope of the question being answered in this project is not how political language changes over time, all data were incorporated into this project.

2.2 Modelling

2.2.1 Programming Language Choice

For this project, I choose to utilize Python as the language of data cleaning, data processing, and data modelling. I am normally an R programmer, but decided that this would be an opportunity to use the Python ecosystem to grow my ability and knowledge in this platform. The drawbacks to this approach are denoted below:

- Scikit-learn has a focus on Machine Learning but not Statistical Output. This means that it uses metrics to determine that the model is a good fit or a poor fit, but it does not have the same statistical summary ability that R has where the statistical models can really be interrogated.(Hvitfeldt and Silge 2022)
- My ability with Python's visualization libraries are inherently absent. This means that this report will lack in depth visualization, but as this is not the focus of the project, I think this is acceptable. I will grow this skill, but where R has statistical plotting based in the base language, Python takes much more effort for the same results.

I would like it to be known that these are my personal issues with the Python machine learning interface and it was a choice that I made to grow my experience in the Python Data Science space. I do not think this affected my ability to do this project well for the scope that I have set for it.

2.2.2 Modelling Methodology

For this question, I utilized guidance from Supervised Machine Learning for Text Analysis in R (Hvitfeldt and Silge 2022) and the Natural Language Toolkit book (Bird, Klein, and Loper 2019) as well as the Scikit-Learn's documentation, particularly "Working with Text Data" (developers 2022b) and "Pipelining: chaining a PCA and a Logistic Regression" (developers 2022a).

The main objective when working with text data is to get it tokenized. A token can be multiple things in NLP, but for this project, I focused on word tokenization. This is a non-trivial objective with how the text data was collected. The text data had to be combined so that it could be: One Row = One Article. The next challenge was tokenizing these articles where the data could still be extracted out of the Scikit-learn object attributes. This meant that the data needed to be tokenized within the DataFrame object.

2.2.2.1 Model Pipeline

- Tokenization
- Lemmatization
- Vectorization
- Hyperparameter Tuning
- Final parameter selection
- Modeling

2.2.2.2 Tokenization

In this modelling execise, I choose to tokenize the words in each article and then tokenize them into their Lemmas. Lemmatization is a linguistics-based stemming of words. This essentially uses a rules-based approach to tokenizing the word down to its meaning in context and based on linguistics rules. (Hvitfeldt and Silge 2022) For this, I utilized the tokenizer and the lemmatization processes within the NLTK package in Python. (Bird, Klein, and Loper 2019). I choose this methodology because in testing out my models originally, I found that there were a lot of words that would contribute to the modelling score that were just different spellings of the same word. I wanted to have as many unique words be a part of the selection corpus as I reasonably could and Lemmatization was the right course of action with this corpus. I also choose to remove common English stop-words using the NLTK's stopword list.

2.2.2.3 Vectorization

The last thing that I had to decided before moving on to the modelling portion of the project was how to vectorize the word tokens. There are a few different ways to vectorize natural language including term frequency (TF) or one hot encoding, but since the core question is whether or not the model can classify based on the text to use the Term Frequency-Inverse Document Frequency vectorization method that has long been used in token vectorization.

Term Frequency is defined as how frequently a word appears in a document. The inverse document frequency measures how frequent a word is in a unique to the document context. (Silge and Robinson 2017)

$$tfidf = f_{i,j} / \sum_{i' \in j} f_{i',j} \cdot log(\frac{N}{df_i})$$

where $f_{i,j}$ is number of occurences of i in j, df_i is the number of documents containing i, and N is the total number of documents.

This is a useful vectorization technique since it ensures that words that are rare but heavily used

in one document get higher weights than words that are used through all documents. Once the words are vectorized in this manner, they become a part of a document term matrix which is a sparse matrix. In this case, the matrix was quite large considering how large the corpus are. Since the model is tokenized as part of the pipeline in Scikit-learn, I am not sure how large the sparse matrix is.

2.2.2.4 Feature Selection

Prior to modeling, since the feature space is incredibly rich. I wanted to narrow it down to a certain number of features. I did this in two ways, first in each of the models, I ran a randomized grid search that measured 200, 300, and 400 tokens and 1 token or a 2 token ngram (which is two words next to each other instead of just a single word). I found this quite often when inspecting scikit-learns documentation as parameters to be optimized. (Pedregosa et al. 2011)

Secondly, I utilized scikit-learn's `TruncatedSVD()` utility in the pipeline to do further dimensionality reduction. I also ran parameter optimization with this technique of selecting (5, 15, 30, 45, 60) components to run each model. I chose this technique at the suggestion of scikit-learn and learned that this is called "Latent Semantic Analysis." This is an unsupervised learning technique that is often used in text data to perform feature reduction. It also is a technique that works on the types of sparse matrices that you get with text data. The overall concept is that it performs a Singular Value Decomposition on the text data which will hopefully decompose the text data into vectors whose data encapsulates words that are closely aligned. (Ioana 2020)

2.2.2.5 Models

The models that were chosen to model this question were:

- LASSO Logistic Regression
- Linear Support Vector Machine
- k-Nearest Neighbors

- Random Forest

2.2.2.5.1 LASSO Logistic Regression

The LASSO logistic regression model was used because of its simplicity and its ability to do feature reduction. Since this text model has a lot of features, the ability to determine which features were most important in classifying the document was an important question that I wanted to have an answer for. I also chose it for its simplicity and that I am only classifying between two classes. I used the `LogisticRegression()` function in scikit-learn.

The only metric that I chose to tune in this model was lambda or C as it is called in scikit-learns nomenclature.

2.2.2.5.2 Linear Support Vector Machine

Support Vector Machines are one of the standard text classification models and almost every source I use in this project makes references to them. Since the data are in a large feature space, it appears to be a good space to perform the hyperplane linear classification that comes with SVMs. (Li et al. 2021) I used the scikit-learn `LinearSVC()` function and used their regularization methods used within the function. I tried to optimize the regularization technique based on l1 or l2, and then the lambda penalty.

2.2.2.5.3 k-Nearest Neighbors

k-Nearest Neighbors is good in large spaces since they are a relatively simple classifier that goes based on distance between features to determine the class. I wanted to use this algorithm since it selects features based on similarity since that addresses the main research question. (Li et al. 2021) I used the `KNeighborsClassifier` in scikit-learn and optimized based on number of neighbors (2, 3, 4, 5, 6).

2.2.2.5.4 Random Forest

I used Random Forests because of their improvement over regular decision trees. I also wanted to ensure as much as possible that the tree didn't overfit the training data so that it had good results on the test data. Random Forests are used often in text classification techniques. Random Forest also utilize the "divide and conquer" that are easier to understand. (Li et al. 2021) I used the `RandomForestClassifier()` from scikit-learn and optimized based on max features being (sqrt, log2), criterion (gini, entropy) and max depth (5-50 as determined by numpy).

3 Results

The following section will go over the results of the 4 models and what their metrics were against the test set. The test set is 20% of the total corpus and is shuffled and stratified on the bias classifier.

Models were optimized using a Randomized Grid Search script in Python for each model. This was done to save on computational resources. Each optimization run took 60-120 minutes and output 40 results. Each result was derived using 5-fold cross validation that is baked into the `RandomizedGridSearch()` class.

The section has the modelling code as a part of the model to show the code utilized. The overall flow will be standard model using the default parameters of each function in the Pipeline of scikit-learn. Then the optimized Pipeline using the parameters found in the grid search.

Pipeline objects were used for their reproducibility across all of the models while only having to change the modeling object.

3.1 Overall Summary

The optimized models performed nearly identically on their Accuracy scores with the highest accuracy score being 87% for the Random Forest model. Each of the other models achieved an 86% accuracy. Accuracy was used as the chief metric in the analysis since it is more important to

get the classification right rather than objectives of the metrics of Precision or Sensitivity. The two classes are of equal importance in their classification. It was not important that one was classified correctly more often than the other.

As for hyper-parameter selection, every model choose the model with 400 max features. Every model besides knn chose having a 2 ngram token rather than a single word token. This aligns with the fact that two words will give more information than a single word, but it appears to have more impact as well with political speech. Number of components was selected to be 60 in every situation other than the Linear SVM. I think this allows for the same conclusion as the 2 ngram token, which is that more components will capture more of the variance and capture the most information.

The overall conclusion is that traditional machine learning techniques using this method reaches their peak at about 86-87% accuracy which means that there is good evidence that the words chosen from this corpus do a good job in correctly classifying political speech from these sources.

3.2 Structure of the Results

For ease of use, I have attached the results as an appendix in html. This is the best output for the jupyter notebook that I have used to get the results.

Preceeding the results is a table of the grid search and the top 5 results.

First there will be the standard model without any hyperparameter optimization and then the optimized result.

The output from scikit-learn will be the classification metrics based on test set being measured. The second output is an array of the confusion matrix. I know there are prettier ways of presenting the confusion matrix, but this is functional for this project.

Lastly, a print-out of the 400 tokens follows the result for further inspection of the tokens prior to SVD. Then an array of variance explained by the SVD is shown.

4 Further Work

This project was done with the intention of further work being accomplished after its completion. The intent of this project was to see if the words of an article could be used to predict what political bias they come from and each model could accurately classify their bias with an accuracy of 86%.

The work I would like to do in the future is to interrogate the pipeline a bit further. I may do this in R rather than Python since it has easier access to statistical properties without having to calculate them each through code abstraction in Python. I would like the piecemeal the pipeline to really see what each vector of the SVD is really encompassing from a word perspective. I could not get a chance to examine these attributes with my current knowledge of Python and scikit-learn. I would be interesting in the unsupervised aspect of this and what words were grouped together.

I would also like to further refine my list of stopwords, I noticed upon completing this report that "Heritage Foundation" is one of the top tokens and had I known that or thought to put that in the stopword list, it would not have appeared in the data set. Considering that the Heritage Foundation is a large portion of the right-wing data, this would be problematic in a finished product.

I would like the grow the corpus to incorporate other think-tanks and news sources. I would keep them to being ideological. I think that bias is too hard to assign to normal news websites, so I would want to ensure than any further data would have significant and explicit bias baked into the assumptions.

I would also like to grow this model into the deep learning space. I think that moving this model into a political word embedding model may have future utility in the world of political speech.

Lastly, I would like to productionize this model to analyze Supreme Court opinions. That is the end goal of this work. I would like to be able to build a political sentiment dictionary from which to analyze Supreme Court opinions to see if the data will show the merits of textualism or if it

will provide evidence that it is just a word and not an actual legal theory.

5 References

Annotated, Congress. 2022. "Intro.8.2 Textualism and Constitutional Interpretation." US Congress. 2022. https://constitution.congress.gov/browse/essay/intro-8-2/ALDE_00001303/.

Barham, J., and J. Barham. 2021. "Top Influential Think Tanks Ranked for 2022." Academic Influence. June 2021. https://academicinfluence.com/inflection/study-guides/influential-think-tanks.

Bird, S., E. Klein, and E. Loper. 2019. *Natural Language Processing with Python*. O'Reilly.

Breiner, A. 2021. "How Did the Courts Become so Politicized?" Library of Congress. September 2021. https://blogs.loc.gov/kluge/2021/09/how-did-the-courts-become-so-politicized/.

Bulao, J. 2022. "How Much Data Is Created Every Day in 2022." November 2022. https://techjury.net/blog/how-much-data-is-created-every-day/.

Delaney, N. 2022. "Roe v. Wade Has Been Overturned. What Does That Mean for America?" June 2022. https://www.hks.harvard.edu/faculty-research/policy-topics/fairness-justice/roe-v-wade-has-been-overturned-what-does-mean.

developers, Scikitlearn. 2022a. "Pipelining: Chaining a PCA and a Logistic Regression." 2022. https://scikit-learn.org/stable/auto_examples/compose/plot_digits_pipe.html#.

———. 2022b. "Working with Text Data." 2022. https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html#.

Hvitfeldt, E., and J. Silge. 2022. *Supervised Machine Learning for Text Analysis in r*. 1st ed. Data Science Series. CRC Press.

Ioana. 2020. "Latent Semantic Analysis: Intuition, Math, Implementation." May 2020. https://towardsdatascience.com/latent-semantic-analysis-intuition-math-implementation-a194aff870f8.

Kteily, N., and E. Finkel. 2022. "Leadership in a Politically Charged Age." August 2022. https:

//hbr.org/2022/07/leadership-in-a-politically-charged-age.

Li, Q, Peng H., J Li, Xia C, R Yang, L Sun, P Yu, and L He. 2021. "A Survey on Text Classification." *ACM Trans. Intell. Syst. Technol* 37 (111). https://arxiv.org/pdf/2008.00364.pdf.

Litman, H. 2021. "Originalism, Divided: The Theory Has Not Provided the Clarity Some of Its Early Proponents Had Hoped It Would." The Atlantic. May 2021. https://www.theatlantic.com/ideas/archive/2021/05/originalism-meaning/618953/.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Silge, J, and David Robinson. 2017. *Text Mining with r.* O'Reilly. https://www.tidytextmining.com/.

Zeitz, J. 2022. "The Supreme Court Has Never Been Apolitical." Politico. April 2022. •.