Predicting the Political Compass Orientation of US Presidential Speeches

Alma Linder
Linköping University
alma24@ru.is

Maximilian Pfeil
Technische Universität Wien
maximilian24@ru.is

Abstract

Political classification is not a novel task. In this work, we create an application that classifies and maps text entered via a GUI onto Wayne Brittenden's Political Compass. Several classification and regression methods were implemented and compared. We trained our model on the Miller Centers's collection of United States president speeches.

1 Introduction

Predicting the political orientation of text data is not a new concept. In this work, we train various classifiers and regressors on a selection of speeches by US presidents. The speeches, collected by the Miller Center of Public Affairs, are in the public domain. Our goal is to accurately predict the orientation of unseen speech data, focusing on the binary classification of *left-leaning* and *right-leaning* positions, as well as regression to determine a specific position on the Political Compass.

This report provides an introduction to the Political Compass (PC) model and documents the implementation process of our prediction methods. It was created as part of the final project for the course "Natural Language Processing" at Reykjavik University during the winter semester of 2024.

1.1 The Political Compass

The Political Compass, developed by New Zealand journalist Wayne Brittenden (O'Connell, 2003), argues that a simple left-right scale is insufficient for accurately capturing the complexity of political views. To address this limitation, it introduces a second axis: authoritarian–libertarian, alongside the economic left-right dimension. The official Political Compass website also features a test comprising 62 propositions. After users indicate their level of agreement with each proposition, they receive their "PC position". In a brief introduction video on YouTube (The Political Compass Organization, 2017), the compass is described as being "free of any agenda," stating it is "simply about describing politics in more meaningful terms."

Both the test's propositions and the model have faced criticism. Journalist Tom Utley critiques the phrasing of the propositions, while economist Daniel J. Mitchell questions the positioning of historical figures on the compass. Academic William Clifton van der Linden and the Encyclopedia Britannica both agree that the scientific basis for the model is questionable (Wikipedia). However, the official Political Compass website includes several media reviews and endorsements from subject matter experts, supporting the credibility of the compass and its underlying concepts (The Political Compass Organization). Therefore, we conclude that the compass is a sensible tool for our experiments within the scope of a university course.

1.2 Related Work

Political classification has been subject to NLP-based approaches in several previous works. For example, (Falck et al., 2020) investigates political bias in newspapers using entity sentiment analysis. Furthermore, the works in (AlDahoul et al., 2024) and (Hey) both deduce transformer-based approaches such as BERT yield better classification results than vector embeddings like word2Vec or GloVe. The former classifies the leaning of YouTube videos based on titles, and the latter uses various news corpora to evaluate different classification methods. These previous woks have one factor in common; less focus on feature extraction and more focus on training methodology. In addition, political classification of speeches occur less frequently in the literature. Considering the unsta-

ble political circumstances in many parts of the world - including the upcoming American president election - we thought a classifier of politicians' speeches would be an interesting dimension to the political classification problem.

2 Implementation

The corpus of speech data we used is a collection made public by the Miller Center of Public Affairs. It consists of more than 1000 speeches by US presidents, from George Washington's first annual message to congress in 1790 to Joe Biden's 2024 State of Union Address. We chose Python as our programming language, as several useful libraries were introduced to us during the NLP course.

2.1 Preprocessing

We first created `preprocessing.py` which converts text to lower case, removes punctuation and stop words, tokenizes and lemmatizes the speeches. Named entities are also extracted. We continued by assigning all presidents a position on the Political Compass, mapped as an (x, y) coordinate pair in the range $[-10, 10]$ (see table 3 in appendix A). The x-axis represents the economic left-right dimension, the y-axis the authoritarian-libertarian scale (more info in the Discussion section).

2.2 Vectorization, Classification & Regression

Running our preprocessing module results in a JSON file containing all preprocessed speeches and their corresponding political orientation. This file serves as input to our `main.py` module, which trains and evaluates five different feature extraction methods combined with three classification and three regression methods. Our feature extraction methods include TF-IDF, Bag-of-Words, n-grams, Word2Vec, and BERT. For classification, we use Random Forest, Support Vector Classifier (SVC), and Logistic Regression. For regression tasks, we apply Random Forest, Support Vector Regressor (SVR), and Linear Regression. The performance of all these methods is shown in section 3.

Implementation of the TF-IDF feature extractor was done by using the `TfidfVectorizer` from SciKit-learn on the preprocessed speeches. The number of features extracted was limited to 5000 by the `max_features` argument.

The `CountVectorizer` from SciKit-learn's `feature_extraction` was used for both the bag-of-words and n-grams feature extraction. The n-grams extraction used an additional parameter to specify that features should be generated from all possible bigrams in the speeches. Similar to the TF-IDF extractor, the maximum number of features extracted was set to 5000 for both bag-of-words and n-grams extraction.

Word2vec embeddings is a model that comes from the Gensim library. Our word2vec feature extractor used the `word2vec-google-news-300` corpus, where vector dimensions are 300. The word2vec feature extraction also computes a single vector for each speech by averaging the word embeddings of each word. It is these vectors that are used for training and testing.

The `BertModel` was retrieved from the transformers library, and built on the `bert-base-uncased` data set (`bert-base-uncased` was trained on a corpus of raw text in English (Face)). The `BertTokenizer` had to be used as well (rather than the tokenization implemented in the preprocessing), so that the tokenization of the speeches would be compatible with the BERT model. Pooling was applied to the output embeddings generated by the BERT model to improve both accuracy and efficiency. The specific pooling method used was mean pooling, which produces a vector for each speech by averaging the information from all token embeddings, similarly to the averaging done for the word2vec extractor (Jurafsky and Martin, 2024). Finally, the dimensionalities of the feature vectors from the BERT model were reduced to 50 by using Principled Component Analysis (PCA). The PCA function was taken from the SciKit-learn library.

Finally, all the feature extractors use the `train_test_split` function of SciKit-learn's `model_selection` to split the vectorized speech data into a training set (80%) and a test set (20%). To maintain modularity and organization, the feature extractors were implemented in respective files, namely `tfidf.py`, `ngrams.py`, `bow.py`, `word2vec` and `bert.py`.

Regarding the classifiers and the regressors, these were imported from SciKit-learn's `svm` (Support vector classification and -regression), `ensemble` (Random forest classification and -regression) and `linear_model` (Logistic regression for classifica-

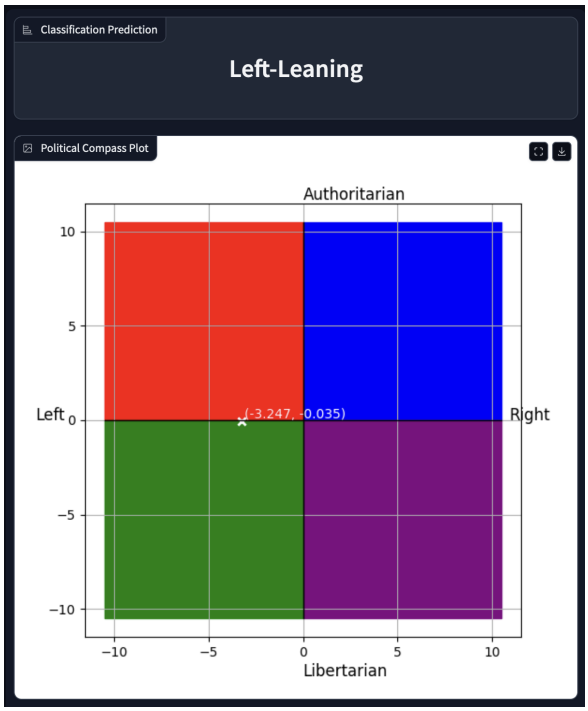

Figure 1: Prediction of Kamala Harris speech.

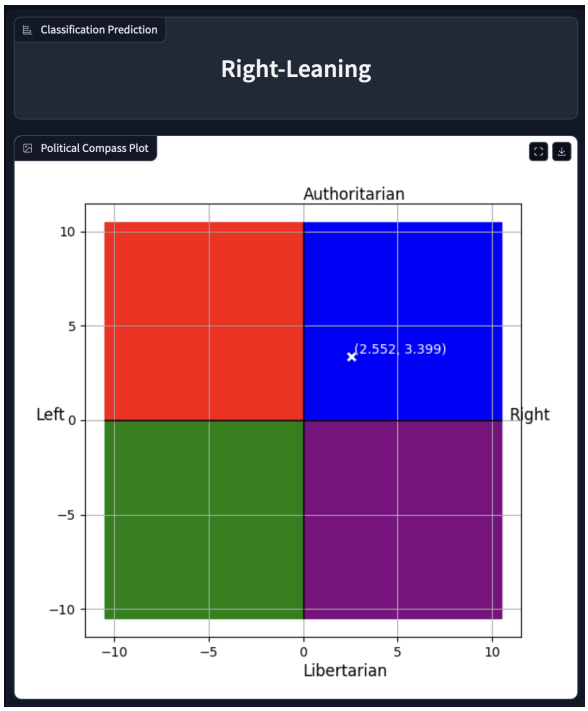

Figure 2: Prediction of Donald Trump speech.

tion and Linear regression for the regression task). The number of iterations of the Logistic regression classifier was limited to 1000.

Whenever `main.py` is run, all vectorization, classification and regression methods are stored as serialized .pkl files. These files are then deserialized and used for the next component we implemented.

The `main.py` file also generates the SHAP plot described in section 3. This is a beeswarm plot from the SHAP library, with detail over the 100 most important words.

2.3 Gradio UI

Based on what we saw in Lab 7, we implemented a basic user interface using Gradio in `gradio_ui.py`. The UI consists of an input text field where users can paste speech data or experiment with shorter word spans. There are separate output fields for classification and regression. The classification result is calculated by a majority vote over all available classifiers, and either "Left-leaning" or "Right-leaning". The regression result is presented as a position of the predicted x and y coordinates on a Political Compass plot we drew using matplotlib. Our current implementation uses only our most accurate regressor for prediction, as this allowed us to intuitively demonstrate the results of the SHAP plot (see Section 3). However, this behaviour can easily be changed in the `gradio_ui.py` module. We experimented with calculating a mean and median prediction similar to the majority vote in classification. A median prediction worked better as there was occasionally one poor regressor which would distort the overall prediction.

The Gradio UI can be seen in Figures 1 and 2. The figures show the predicted positions of US presidential candidates Kamala Harris and Donald Trump, based on speeches from 2020 and 2024, respectively.

3 Results

Running `main.py` both trains and evaluates classification and regression models. Evaluation results are printed out for each feature extraction method. A full log of one such run is contained in `eval.txt`. This section presents our results in Tables 1 (Classification) and 2 (Regression). Running `main.py` takes approximately 10 minutes on a MacBook Pro with an Apple Silicon M2 CPU, with the BERT method accounting for most of that time (around 7 minutes).

3.1 Classification

Table 1 shows the performance of the classifiers we trained. Interestingly, the simple TF-IDF combined with the Support Vector Classifier performs best. Bag-of-words and n-gram vectorizers achieved comparable results to the word2vec and BERT methods. We also noticed how choosing a different classification method has a significant impact on performance (e.g. word2vec with SVC or Random Forest). We conclude that we would be able to improve these results further with hyperparameter tuning through grid search.

Vector.	Class.	P	R	F1	Acc.
TF-IDF	LogR	0.84	0.83	0.84	0.83
TF-IDF	**SVC**	**0.85**	**0.85**	**0.85**	**0.85**
TF-IDF	RF	0.75	0.75	0.75	0.75
BoW	LogR	0.82	0.82	0.82	0.82
BoW	SVC	0.77	0.75	0.75	0.75
BoW	RF	0.70	0.70	0.70	0.70
n-gram	LogR	0.78	0.77	0.77	0.77
n-gram	SVC	0.75	0.73	0.72	0.73
n-gram	RF	0.74	0.74	0.74	0.74
word2vec	LogR	0.66	0.67	0.66	0.67
word2vec	SVC	0.69	0.69	0.69	0.69
word2vec	RF	0.77	0.76	0.76	0.76
BERT	LogR	0.72	0.71	0.71	0.71
BERT	SVC	0.72	0.72	0.72	0.72
BERT	RF	0.76	0.76	0.76	0.76

Table 1: Classification Results (Feature extraction method, Classifier type, Precision, Recall, F1-score, Accuracy)

3.2 Regression

Table 2 displays the regression results of each feature extractor, evaluated using Mean Absolute Error (MAE, which measures the absolute distance between predictions and true observations (Wikipedia.)) and Coefficient of Determination (R^2-score, evaluating how well the regression model fits the data (wikipedia.)). The results are shown for computed x and y coordinates. It is evident that the TF-IDF feature extractor, combined with linear regression, yielded the best results across both evalutation metrics and coordinate values. In contrast, the n-grams and bag-of-words extractors — especially when paired with linear regression — perform poorly, with average MAE-values of 6.32 and 6.76, respectively, from the true observation. No combination of feature extractor and regressor achieves R^2-scores as high as TF-IDF with linear regression. The generally powerful BERT model is also outperformed by the TF-IDF performance. The BERT model is likely to require additional fine-tuning. The Random Forest and Support Vector regressors are the most stable, showing minimal variation in MAE and R^2-scores across different feature extractors, even though they are not in par with TF-IDF and linear regression.

		x		y	
Vector.	**Regr.**	**MAE**	**R2**	**MAE**	**R2**
TF-IDF	**LinR**	**3.28**	**0.52**	**2.23**	**0.54**
TF-IDF	SVR	4.46	0.19	2.76	0.13
TF-IDF	RF	4.50	0.27	2.77	0.31
BoW	LinR	6.76	-1.82	4.49	-1.26
BoW	SVR	4.55	0.16	2.89	0.05
BoW	RF	4.44	0.31	2.76	0.31
n-gram	LinR	6.32	-1.07	4.10	-0.85
n-gram	SVR	4.81	0.02	3.00	0.00
n-gram	RF	4.43	0.29	3.10	0.13
word2vec	LinR	4.47	0.00	3.17	0.07
word2vec	SVR	4.74	0.03	3.00	-0.03
word2vec	RF	4.44	0.28	2.90	0.23
BERT	LinR	4.61	0.19	3.21	0.11
BERT	SVR	4.51	0.18	2.86	0.10
BERT	RF	4.91	0.18	3.02	0.17

Table 2: Regression Results (Feature extraction method, Regressor type, Mean Absolute Error and R2-score for x and y)

3.3 SHAP plot

An excerpt of the resulting beeswarm SHAP plot described in section 2.2 is shown in Figure 3 over the 32 most influential words for the model prediction. The more weight a word has to left- or right leaning, the higher the SHAP value (i.e. red) in the plot is in that direction. For example, the word "*state*" is strongly associated with right-leaning, and the word "*men*" is strongly related to left-leaning.

A few non-political words the model detects and uses during classification and regression are "*that's*" or "*said*", for instance. Interestingly, the model also captures certain anticipated neutral words - such as the countries "*Vietnam*" and "*Iraq*". These are on the contrary quite left- and right oriented, respectively. This is due to the historical

context of presidential leaning during the time of the Vietnam war (John F. Kennedy, who was a democrat) and the wars in Iraq (George W. Bush, who was a republican).

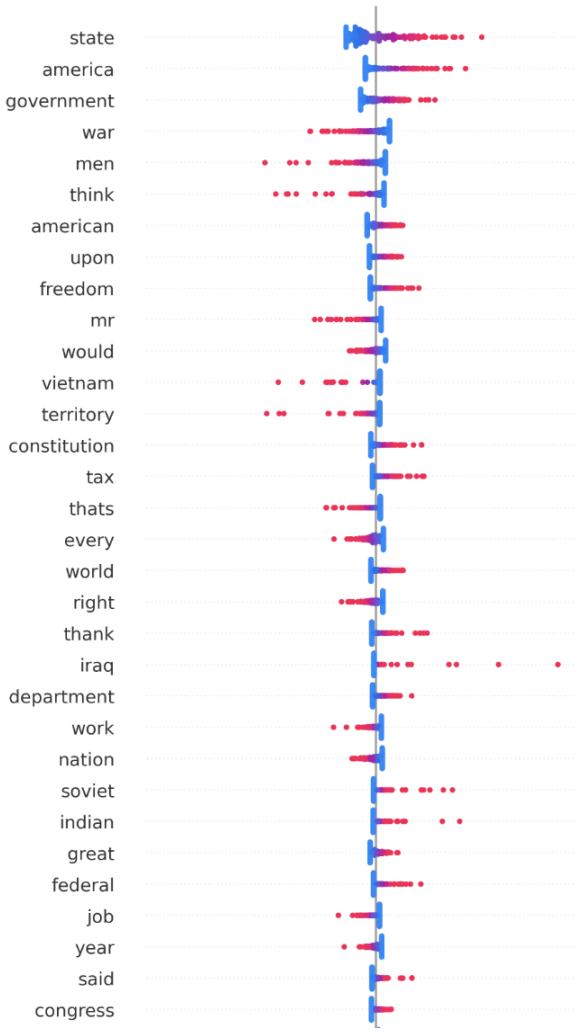

Figure 3: Shap plot of the 32 most important words

4 Discussion

As explained in section 2.1, the presidents positions were mapped onto the political compass by our hand. This methodology obviously comes with flaws - although we tried to be as objective as possible - since some unconscious subjectivity remain. This lack of control can result in bias amplification and in extension lead to skewed classification and regression.

Furthermore, since the speech corpus from Miller Center of Public Affairs contains speeches from 1790 to the present day, the used language has evolved significantly - for example choice of words, rhetorical style and phrasing. This historical variation is problematic for our model, which is trained on data where extreme political standpoints (left-libertarianism for instance) are reflected by speech styles not found in contemporary discourse. Consequently, modern speeches generally get mapped to the center portion of the compass, undermining ideological variety.

An attempt to improve the feature extractors included named-entity recognition tagging (NER tags) in the preprocessed speeches. However, all attempts proved futile, as conflicts arose between the dimension differences in our vectorizers and the NER tags information. To resolve the issue, zero-padding on either NER tags or features from the feature extractors was experimented with, as well as testing different methods of distributing NER tags (such as `en_core_web_sm` from the spaCy library, or the `ne_chunker` from nltk). Future work could explore more sophisticated NER tagging methods that allow for more dynamic dimensionality handling; transformers pipelines for instance.

Referring back to the results in section 3, the transformer-based feature extractor, BERT — anticipated to be the strongest due to its large contextual windows enabled by the transformer architecture — was outperformed by simpler methods, particularly TF-IDF and bag-of-words. BERT's underperformance is likely due to the use of a general-purpose model, `bert-base-uncased`, which may not capture domain-specific nuances. To improve BERT's performance, fine-tuning on domain-specific data could be beneficial. Additionally, using pretrained models that are more domain-tuned, such as the `bert-political-leaning-it` model on Hugging Face (Face.), might better capture the intricacies of political discourse and spoken language.

For demonstration purposes, the mapping of an empty user input in the gradio interface plot was shifted to the origin with mean centering (subtract the average of all x- and y coordinates from all initial president coordinates in table 3 (University)) - otherwise empty prompts would be mapped into the second quadrant, since the average of all the politicians is located there. This solution obviously fails to do the same for nonsense-input or input unrelated to politics, such as "*stairs*" for instance. The desire to have typical non-political content mapped to origin is because we want it to be labelled/depicted as neutral. To tackle the problem at its root we would require an extensive de-biasing

algorithm of the speeches data. This could involve for example adversarial debiasing, which involves training an additional so called adversarial model alongside the task-specific model that detects and minimizes certain biases z according to a loss function $L(z', z)$ (Zhang et al., 2018).

References

Nouar AlDahoul, Talal Rahwan, and Yasir Zaki. 2024. Polytc: a novel bert-based classifier to detect political leaning of youtube videos based on their titles. *Journal of Big Data*, 11(1):80.

Hugging Face. Bert base model. https://huggingface.co/google-bert/bert-base-uncased. Accessed: 2024-10-23.

Hugging Face. Bert political leaning. https://huggingface.co/MattiaSangermano/bert-political-leaning-it. Accessed: 2024-11-14.

Fabian Falck, Julian Marstaller, Niklas Stoehr, Sören Maucher, Jeana Ren, Andreas Thalhammer, Achim Rettinger, and Rudi Studer. 2020. Measuring proximity between newspapers and political parties: the sentiment political compass. *Policy & internet*, 12(3):367–399.

Alexander Martin Hey. Using nlp analysis to categorise statements to values on the political compass.

Dan Jurafsky and James H. Martin. 2024. Speech and language processing (3rd ed. draft). pages 143–144. Prentice Hall. Draft edition.

Miller Center of Public Affairs. Presidential speeches: Downloadable data. data.millercenter.org. Accessed: 2024-10-17.

Pamela Licalzi O'Connell. 2003. Online diary. https://www.nytimes.com/2003/12/04/technology/online-diary.html. Accessed: 2024-10-17.

The Political Compass Organization. The political compass. politicalcompass.org. Accessed: 2024-10-17.

The Political Compass Organization. 2017. The political compass - a brief intro. youtube.com/watch?v=5u3UCz0TM5Q. Accessed: 2024-10-17.

Kent State University. Spss tutorials: Computing variables: Mean centering. https://libguides.library.kent.edu/SPSS/MeanCentering#:~:text=Introduction-,Mean%20centering%20refers%20to%20a%20type%20of%20variable%20transformation%20wherein,whose%20mean%20value%20is%200. Accessed: 2024-11-01.

wikipedia. Coefficient of determination. https://en.wikipedia.org/wiki/Coefficient_of_determination. Accessed: 2024-10-14.

Wikipedia. Mean absolute error. https://en.wikipedia.org/wiki/Mean_absolute_error. Accessed: 2024-11-14.

Wikipedia. The political compass. https://en.wikipedia.org/wiki/The_Political_Compass. Accessed: 2024-10-17.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340.

A President mappings

President	x	y	President	x	y
Andrew Jackson	6	6	Martin Van Buren	-5	-3
James K. Polk	-6	3	Franklin Pierce	-4	1
James Buchanan	-3	2	Grover Cleveland	-4	-1
Woodrow Wilson	-6	8	Franklin D. Roosevelt	-9	6
Harry S. Truman	-7	4	John F. Kennedy	-4	-2
Lyndon B. Johnson	-5	5	Jimmy Carter	-4	-3
Bill Clinton	-3	-3	Barack Obama	-4	-2
Joe Biden	-5	2	Thomas Jefferson	-7	-8
James Madison	-6	-7	James Monroe	-5	-5
Andrew Johnson	5	4	Abraham Lincoln	-6	7
Ulysses S. Grant	4	4	Rutherford B. Hayes	3	2
James A. Garfield	2	1	Chester A. Arthur	4	1
Benjamin Harrison	5	3	William McKinley	6	4
Theodore Roosevelt	-5	5	William Taft	4	3
Calvin Coolidge	8	4	Herbert Hoover	7	5
Dwight D. Eisenhower	3	2	Richard M. Nixon	7	8
Gerald Ford	4	3	Ronald Reagan	9	7
George H. W. Bush	6	5	George W. Bush	6	5
Donald Trump	8	7	John Adams	3	5
John Quincy Adams	2	2	William Harrison	5	3
John Tyler	6	4	Zachary Taylor	5	4
Millard Fillmore	4	3	Warren G. Harding	7	6

Table 3: Hand-labelled president coordinates.