THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FINE TUNING TRANSFORMERS FOR POLITICAL MANIFESTO SUMMARIZATION AND QUANTIFICATION

PATRICK ELISII

SPRING 2023

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Amulya Yadav
Professor of Computer Science and Engineering
Thesis Supervisor

John Hannan
Professor of Computer Science Engineering
Honors Advisor

Abstract

This thesis presents a method to summarize political manifestos in an unbiased manor. Biased reporting of political candidates is prevalent in the media, which influences viewers to adopt that bias. To overcome this issue, the proposed method extracts specific policy standards from the manifestos to provide an impartial summary. The research involves various model experiments that fine-tune pre-trained language transformers to classify each sentence of political manifestos. The dataset used in the study is comprised of hand-annotated labelled provided by the Manifesto Project. The results of the experiments show that language models can identify political stances, even when there are many categories. However, the model's performance declines on categories with a small number of examples in the dataset. This study suggests and experiments a few ways to improve accuracy on these categories. The study makes significant contributions to the field of natural language processing and political science by providing a new approach for summarizing political manifestos. This approach could be helpful to political analysts, journalists, and policymakers in summarizing complex political documents. Furthermore, the proposed method can be extended to other languages, making it more widely applicable. In summary, this study suggests that language models are powerful tools in summarizing political manifestos and can contribute significantly to promoting unbiased reporting of political candidates.

Table of Contents

List of Figures

List of Tables

Acknowledgements

A special thank you to Amulya Yadav for his continuous mentorship throughout this project. Both his analytical and technical insights helped this project develop every step of the way.

Another thank you to John Hannan for his support throughout the writing process.

Chapter 1: Introduction

The goal of this research is to create unbiased summaries of political positions found in party electoral programs. In the United States federal elections, each party writes a manifesto called a party platform. The purpose of the platform is to describe a party's political positions and policy plans. Other countries around the world also follow this practice. The information contained in manifestos is verified and accurate, however, misinformation about their contents has become commonplace due to consistent biased and false reporting of political parties and their representatives. Major news networks, journals, newspapers, and social media influencers have been proven to skew the opinions of voters with their biased reporting of political candidates. This research project explores methods of extracting the positions of political parties from manifestos using a machine learning powered approach to deliver an unbiased summary of a party's manifesto. Our solution aggregates and summarizes the policy stances found in a manifesto to remove bias altogether. To accomplish this, the study tests deep learning language models' effectiveness at understanding complex political speech and summarizing it for readers. The approach furthers existing research by experimenting with different methods to solve current issues in political language understanding. Using labeled manifestos provided by The Manifesto Project [1], we trained transformers to classify statements as political positions. A summary is created by aggregating the results of all sentences of a political manifesto.

1.1. Motivation

Bias and misinformation in the news is a commonly accepted flaw of the United States media. According to a poll conducted by the Knight Foundation [4], 91% of Americans believe

bias exists in mainstream media sources and 56% believe there is a fair amount or a great deal of bias. Despite most Americans believing that news networks distribute biased information, most continue to watch their prefered news network. This has led to an increasing polarity in the United States. A survey by Pew Research [6] revealed that republicans and democrats agree that they disagree on basic facts, not just policies. In the poll, 81% of republicans expressed this view and 76% of democrats agreed. The obvious cause for these disagreements is the basic fact that the members of each side generally obtain their information from sources that support their side. Most notably, Fox News for republicans and MSNBC for democrats [7]. Another major contributor to bias and misinformation are the tech companies Meta, Twitter, and Google, or more specifically, their platforms.

Tech companies have control over the information distributed across their platforms and 75% of Americans say these companies have a responsibility to prevent misuse of their platform to influence the election [8]. However, intervening to stop the spread of information, true or not, can also be considered influencing an election. One such example is the Hunter Biden laptop Twitter case. The United States Committee of Oversight and Accountability hosted former Twitter employees to testify and reported in a press release “Former Twitter employees admitted the New York Post’s Hunter Biden laptop story didn’t violate any Twitter policies, but it was taken down anyway. They also admitted they made no attempt to verify the authenticity of the contents of Hunter Biden’s laptop.” [9] Despite no verification of the events, Twitter employees censored information that had the potential to influence an election. Each tech company has censored and has the power to continue censoring similar information, creating widespread confusion. This research introduces a method that extracts and summarizes policies directly from political candidates, to ensure the validity of information and reduce confusion amongst voters.

1.2. Project Overview

Our approach to summarizing manifestos is to classify each sentence of a manifesto into a political position using a transformer model fine-tuned for sequence classification. From the dataset of human labeled manifestos, there are 82 unique political position classifications. Examples of categories in the dataset include welfare state expansion and free market economy positive. To classify text, we trained a deep learning language model called BERT (section 2.2) on the dataset. The team conducted training experiments to increase the accuracy of the predictions and gain a deeper understanding of language models' capability to understand political text.

1.3. Existing Research

There are many statistical analysis papers that leverage the Manifesto Project but very few have tried a machine learning approach. We suspect this is due to the large number of categories to classify and an insufficient number of examples for many of the categories. One study [10] used the Manifesto Project Dataset to derive a score of a parties' level of populism using machine learning. In their approach, they ran binary classification on the MP dataset to detect if a sentence is about populism or not. A study by Kakia Chatsiou [11] compared the effectiveness of several types of embeddings to classify political speech. This paper furthers the research of classifying political speech through our experiments. In this paper, we tune parameters to obtain optimal performance for BERT, we test the impact of performance when isolating dialects of English, conduct experiments on subsets of the dataset, and test training using a custom loss function.

1.4. Manifesto Project

The Manifesto Project collects and analyzes party electoral programs. Their research team has compiled political manifestos from across the world and analyzed each document. They hired political scientists to read each manifesto and categorize each sentence into a policy stance. For example, a sentence in a party platform that states the party will increase funding for universal healthcare would classify as welfare state expansion. The dataset we used for this paper comprises of 51 English manifestos from the Manifesto Project database. The possible categorizations of sentences from the Manifesto Project Codebook are listed in table 1 below.

Domain 1: External Relations
101 Foreign Special Relationships: Positive
102 Foreign Special Relationships: Negative
103 Anti-Imperialism: Positive
104 Military: Positive
105 Military: Negative
106 Peace: Positive
107 Internationalism: Positive
108 European Integration: Positive
109 Internationalism: Negative
110 European Integration: Negative
Domain 2: Freedom and Democracy
201 Freedom and Human Rights: Positive
202 Democracy
203 Constitutionalism: Positive
204 Constitutionalism: Negative
Domain 3: Political System
301 Decentralization: Positive
302 Centralization: Positive
303 Governmental and Administrative Efficiency: Positive
304 Political Corruption: Negative
305 Political Authority: Positive
Domain 4: Economy
401 Free-Market Economy: Positive
402 Incentives: Positive
403 Market Regulation: Positive
404 Economic Planning: Positive
405 Corporatism: Positive
406 Protectionism: Positive
407 Protectionism: Negative
408 Economic Goals
409 Keynesian Demand Management: Positive
410 Economic Growth
411 Technology and Infrastructure: Positive
412 Controlled Economy: Positive
413 Nationalization: Positive
414 Economic Orthodoxy: Positive
415 Marxist Analysis: Positive
416 Anti-Growth Economy: Positive

Domain 5: Welfare and Quality of Life
501 Environmental Protection: Positive
502 Culture: Positive
503 Equality: Positive
504 Welfare State Expansion
505 Welfare State Limitation
506 Education Expansion
507 Education Limitation
Domain 6: Fabric of Society
601 National Way of Life: Positive
602 National Way of Life: Negative
603 Traditional Morality: Positive
604 Traditional Morality: Negative
605 Law and Order
606 Civic Mindedness: Positive
607 Multiculturalism: Positive
608 Multiculturalism: Negative
Domain 7: Social Groups
701 Labour Groups: Positive
702 Labour Groups: Negative
703 Agriculture and Farmers
704 Middle Class and Professional Groups: Positive
705 Minority Groups: Positive
706 Non-Economic Demographic Groups: Positive

000 No meaningful category applies

***Table 1**. Political classifications from Manifesto Project Codebook. Table from Bilbao-Jayo and Almeda [12]*

Standards

The Manifesto Project (MP) collects manifestos from Democratic countries across the world. The parties included in the MP are required to have at least 1 representative in the congress or parliament of the host country. Each document is written, verified, and published by a party before an election and describes in detail the policy plans for the party. The individuals who analyze the documents, called coders, are required to be political scientists or graduate political science students and be native speakers of the language. Coders must pass a screening exam to be qualified to code manifestos for their country.

Coding Process

The first step of coding a manifesto is to split sentences into what MP calls "quasi-sentences." Quasi-sentences are single statements. One sentence can have multiple quasi-sentences, but a quasi-sentence never contains more than one grammatical sentence. For example, take this excerpt from the 2012 Democratic party platform:

"President Obama has already signed into law $2 trillion in spending reductions as part of a balanced plan to reduce our deficits by over $4 trillion over the next decade while taking immediate steps to strengthen the economy now"

This sentence has 2 main ideas and thus the coder added 2 slashed between them to indicate they are separate quasi-sentences:

"President Obama has already signed into law $2 trillion in spending reductions as part of a balanced plan to reduce our deficits by over $4 trillion over the next decade // while taking immediate steps to strengthen the economy now"

Next, the coder assigns a category to each quasi sentence in a step the MP calls Code Allocation. First, reference the table below that shows the code, category, and description for a few policy categorizations. Code 414 represents the policy of economic orthodoxy. Economic orthodoxy is a broad set of policies around the belief that government should prefer economically healthy policies such as reducing budget deficits.

code	category	description
411	Technology and Infrastructure: Positive	Importance of modernisation of industry and updated methods of transport and communication. May include: - Importance of science and technological developments in industry; ...
412	Controlled Economy	Support for direct government control of economy. May include, for instance: - Control over prices; - Introduction of minimum wages.
413	Nationalisation	Favourable mentions of government ownership of industries, either partial or complete; calls for keeping nationalised industries in state hand or nationalising currently private...
414	Economic Orthodoxy	Need for economically healthy government policy making. May include calls for: - Reduction of budget deficits; - Retrenchment in crisis; - Thrift and savings in the fac...
415	Marxist Analysis	Positive references to Marxist-Leninist ideology and specific use of Marxist-Leninist terminology by the manifesto party (typically but not necessary by communist parties). *No...

Table 2. Example Manifesto Project political codes. From Manifesto Project [1]

The table below shows a few example statements from the 2012 Democratic party platform and their categorizations. The first sentence is classified as code 414, economic orthodoxy. The sentence states that the President of the Democratic party, Barack Obama, signed in a law that helps reduce the United States budget deficit. From the code descriptions above, we know trying to reduce the budget deficit is an example of economic orthodoxy. The other two sentences classified as 414 in the table also describe reducing spending and cutting the budget deficit.

quasi_sentence	category	description
President Obama has already signed into law $2 trillion in spending reductions as part of a balanced plan to reduce our deficits by over $4 trillion over the next decade	414	Economic Orthodoxy
while taking immediate steps to strengthen the economy now.	408	Economic Goals
This approach includes tough spending cuts that will bring annual domestic spending to its lowest level as a share of the economy in 50 years,	414	Economic Orthodoxy
while still allowing us to make investments that benefit the middle class now	704	Middle Class and Professional Groups
and reduce our deficit over a decade.	414	Economic Orthodoxy

Table 3. Example quasi sentences and their codes from Manifesto Project [1]

Usage

This project used the MP dataset to train a deep learning language model. The model was trained to classify a quasi-sentence into one of the policy standards from the MP Codebook. The language models we used were pre-trained on a large corpus of text and thus already has internal representations for language. We apply a process called fine-tuning which focusses the model on a specific type of language and task. In this case, we fine-tuned the model for text classification on the political speech provided from the MP.

Chapter 2: BERT

BERT is a transformer model well-regarded because of its strong performance and versatility. BERT stands for Bidirectional Encoded Representations from Transformers [3]. BERT is a pre-trained model, meaning it was trained on an expansive corpus of text and is mainly intended to be fine-tuned for specific tasks. This model was unique at the time of its release because of its bidirectional context training. Before BERT, language models were trained by inputting a sequence of tokens and training a model to predict the next token in the sequence. However, BERT was pre-trained using a method called masked language modeling (MLM). In MLM, a portion of tokens are masked, acting as fill-in-the-blank tokens for the model to predict during training. At its release, BERT was one of the best at learning the meaning and relationships between tokens because it was trained given the text to the left and to the right to predict masked tokens.

2.1. Tokenization and Embeddings

Tokenization is the process of breaking text into embeddings that can be used as inputs to NLP models such as BERT for language tasks. BERT tokenization converts text into a set of word embeddings, segment ids, and position embeddings. The first step of tokenization is to convert sentences into tokens. BERT uses the WordPiece algorithm to split text into sentences, then a sentence into word pieces, either whole words, pieces of words, or punctuation. Then, special tokens and padding tokens are added. The [CLS] token indicates the beginning of a sentence and the [SEP] token indicates the end of a sentence [14]. Then, to ensure all sentences are the same length, we manually set a token length for sequences and fill each sentence with

[Pad 0] tokens until the sequence matches the required length. Next, the tokens are converted to a word embedding vector paired with a segment id and position embedding. See in Figure 1 the inputs to the embedding process are the token outputs from the WordPiece algorithm.

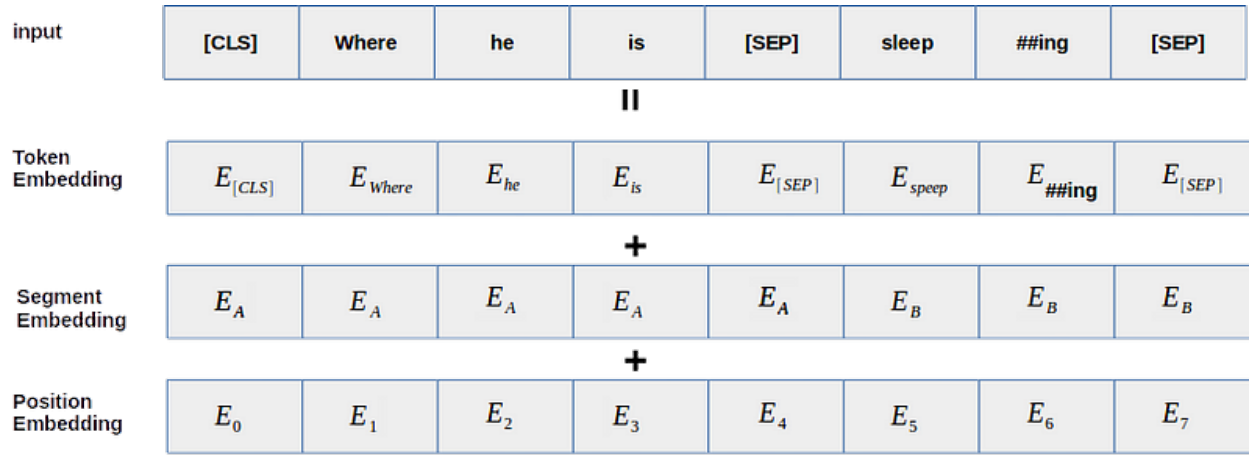

Figure 1. Visual example of converting tokens into embedding vectors. From Pritesh Prakash [15]

Token embeddings are a d-dimensional vector representation of the input token. The BERT tokenizer has an embedding layer that maps an input token to an embedding vector. It has been trained on an extensive vocabulary and has nearly every possible token mapped in its embedding layer. This embedding is paired with a segment embedding, which indicates the sentence of the token. Every sentence has a unique segment embedding. The token embedding is also paired with a position embedding, indicating the position of the token in the text. This enables BERT to process tokens in parallel which significantly speeds up training and inference time.

2.2. Transformers

Transformers are a type of neural network architecture that have revolutionized the field of natural language processing (NLP). The transformer model was introduced by the paper, Attention is All You Need in 2017 by a team at Google. Unlike traditional models that use recurrent or convolutional layers, transformers employ a mechanism known as self-attention. Transformers leverage the self-attention mechanism, enabling the model to weigh and combine embedding sequences based on their relevance within the input sequence. This powerful technique allows for more context-aware representations and long-range dependencies, which greatly improves performance on a wide range of NLP tasks. The architecture of a transformer consists of an encoder and a decoder, both of which are composed of multiple layers of self-attention and feedforward neural networks. The encoder takes a sequence of input tokens and maps them to a sequence of hidden representations. These representations are then used by the decoder to generate an output sequence, word by word. Reference figure 2 below for how attention in transformers works.

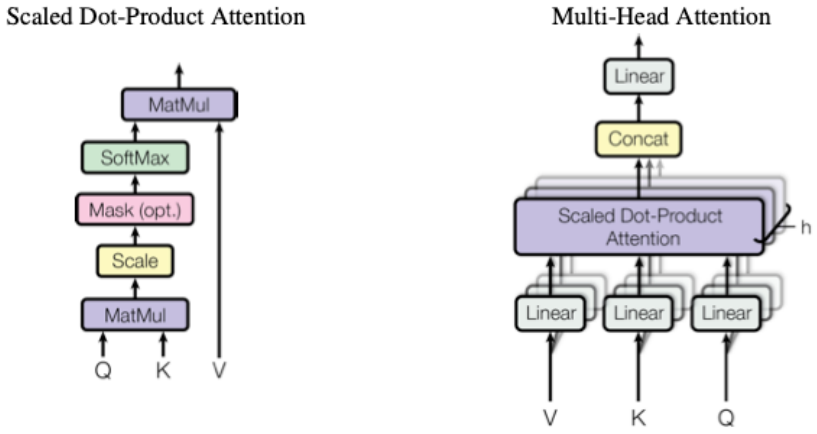

Figure 2. Attention and multi-head attention From [2]

Denote a sequence of tokens input as X. The input sequence is projected into 3 matrices:

- $Q = W^Q X$
- $K = W^K X$
- $V = W^V X$

The inputs consist of Q for query, K for key, and V for value. They are calculated by multiplying the input sequence X by a corresponding weight matrix. $W^Q, W^K,$ and W^V represent the weight matrices for each of the queries, keys, and values. They are initialized randomly and adjusted during the training process. To calculate one attention operation, we perform a matrix multiplication between Q and K, followed by scaling the result by dividing by the square root of the dimension of the K matrix, and optionally apply masking. The scaling factor reduces overfitting because large values of d sub k can result in extremely small gradients for the softmax function [2]. The softmax function converts a vector of real numbers into a probability distribution, turning the raw output scores into interpretable probabilities. In the last step of attention, the softmax of the product vector is computed and matrix multiplied with V. The output, Y, is given by:

$$Y = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

For the multi-head attention mechanism, the inputs Q, K, and V are linearly projected h times with different projections, which are computed in parallel. The resulting outputs are then concatenated and linearly transformed. This multi-head layer provides multiple sets of attention-weighted value embeddings, allowing multiple parts of the sequence to have significance in the final prediction.

2.3. Fine Tuning BERT for Sequence Classification

BERT has been pre-trained on a large corpus using masked language modeling through which it gained powerful representations of vocabulary and of relationships between words. Fine-tuning leverages transfer learning, using knowledge gained from pre-training to improve performance on a specific task. A pre-trained language model can be fine-tuned for various downstream tasks with smaller labeled datasets because it already has accurate representations of text. Popular downstream tasks include summarization, question answering, and sequence classification. Sequence classification, also called text classification or token classification, is the categorization of a sequence of tokens into a group. The most popular form of sequence classification is sentiment analysis which classifies tokens as having a positive or negative connotation. BERT can be fine-tuned to classify all kinds of texts and categories given a training dataset. A dataset containing text and corresponding labels is required to fine tune a language model for sequence classification.

The BERT sequence classification model builds upon the standard BERT architecture by adding a linear layer at the top of the final hidden state, which is then used to predict class probabilities. This layer takes the final hidden state of the special [CLS] token, marking the entire sequence of tokens, as input and outputs a probability distribution indicating the probability that the input sequence belongs to each label. In this paper, we used the Manifesto Project dataset and trained BERT to classify text into policy positions.

Chapter 3: Implementation

Our implementation consists of a data processing pipeline, tokenization, model training, and model evaluation. We used Python and Jupyter-Lab to implement the full pipeline and experiments. The project required extensive Python programming experience, wide knowledge of machine learning and model training, and expertise on the field of NLP transformers. This implementation was custom-tailored for the Manifesto Project dataset and the experiments conducted. The team created functions for calculating loss, created a custom dataset class for the data, created a custom trainer class, and wrote custom evaluation scripts.

3.1 Technology Survey

Hugging Face

To train the model, we used Hugging Face, a python library popular for its natural language processing (NLP) capabilities including summarization, translation, sentiment analysis, sequence classification, and more. Hugging Face has a collection of pre-trained models uploaded by the community and it gained popularity because many of these models achieve state-of-the-art performance on benchmark NLP datasets. One of the most popular and versatile models is BERT, which is the model we chose for this research. The NLP models from Hugging Face use a process called tokenization to break up text into numerical embeddings. This research also leveraged WandB for model surveillance. It is a software-plugin for hugging face that uploads model training information to the cloud, keeping a history of all model training and evaluation jobs.

3.2. Data processing

In the process of preparing data for machine learning experiments, several steps were undertaken to ensure efficient and accurate results. Initially, all labelled datasets were combined into a single dataset to facilitate ease of access and manipulation. The political category codes were then converted to floats to enable seamless integration within the machine learning framework. Subsequently, category variables ranging from 0 to n were created for classification labels, enabling a mapping system to map back to the true political stance code. A stratified train-test split was then implemented, with 20% of the data allocated for testing purposes. Finally, a custom dataset class was created by inheriting from torch.utils.Dataset, which included initializing encodings and labels attributes and overriding the getitem function to retrieve the encodings and label as needed. This comprehensive data processing approach laid the foundation for effective experimentation with the BERT model.

3.3. Training

To set up the training process for the BERT sequence classification model, several steps were taken, focusing on customizing the Huggingface Trainer and creating a tailored training pipeline for the dataset. Training was conducted on the Jupyter-Lab interface in a Google Cloud VM. BERT and other transformers typically require GPUs to train in a reasonable amount of time. With the Google Cloud GPU, the training took about 40 minutes to complete. For the training pipeline, we chose performance metrics that are standard for multiclass classification.

Performance Metrics

First, the chosen metrics, including accuracy, F1 micro, and F1 weighted were defined and included in the trainer initialization. Accuracy is the proportion of correct predictions and total predictions. F1 is a combination of two metrics called precision and recall. As an example, pretend we want to measure the performance of a category called positive. Precision measures how precise the model is. Meaning it is the proportion of correctly classified positives and all points predicted to be positive. Recall measures how well the model finds all positives. Meaning it is the proportion of true positives and all positives.

Accuracy	$\frac{\textbf{Correct Predictions}}{\boldsymbol{Total\ Predictions}}$
Precision (for one category)	$\frac{\textbf{True Positives}}{\boldsymbol{True\ Positives + False\ Positives}}$
Recall (for one category)	$\frac{\textbf{True Positives}}{\boldsymbol{True\ Positives + False\ Negatives}}$
F1 (for one category)	$\frac{\textbf{2} * \textbf{Precision} * \textbf{Recall}}{\boldsymbol{Precision + Recall}}$

Table 4. Performance Metrics. Created by Patrick Elisii

F1 combines precision and recall to obtain a balanced performance score. F1-micro metric calculates the total true positives, false negatives, and false positives globally, while F1-weighted determines the average metrics for each label, and combines them, weighting them by the number of instances for each label. All three metrics offer slightly different insights into model performance and were used to evaluate and adjust the model throughout the experiments.

Loss Function

A loss function is an equation that calculates the error of model predictions. Loss is used by the model to "correct" its mistakes by adjusting its weights according to the loss. The goal of training is to minimize loss. Cross Entropy loss is one of the most widely used loss functions for machine learning classification tasks. Loss is calculated by taking the negative log of the predicted probability outputted from the model [FOCAL LOSS PAPER]:

$$\mathrm{CE}(p_t) = -\log(p_t)$$

$$where\ p_t = \begin{cases} p & if\ prediction\ correct \\ 1-p & if\ prediction\ incorrect \end{cases}$$

Pytorch, a machine learning framework for Python, has a built-in function to calculate cross entropy which is what we used for this project. However, we test another loss function called Focal Loss in one of the experiments described below.

Trainer Settings

Next, we created the custom trainer, incorporating a custom loss function. Hugging face has a trainer class that fits a model to your specified dataset. We created a custom trainer by creating a new trainer object inheriting from the hugging face trainer. This was required to override the loss function. Depending on the experiment, the model used cross-entropy loss or balanced focal loss. Batch size was set to 20, the number of epochs was set to 10, and the learning rate varied depending on the experiment. The training pipeline utilized the Adam optimizer for learning optimization. We used the BERT tokenizer for tokenizing the training and

validation datasets. The BertForSequenceClassification model was initialized at the end of this process to be trained. It is important to note that multiple model checkpoints were saved during training to allow the selection of the best-performing model upon training completion.

Chapter 4: Findings

The purpose of this project is to test the capabilities of language models for understanding political literature. Using the model to extract information from the manifestos removes bias that news networks include in their reports because it provides information directly from the document. For the results to be unbiased, the model must be able to identify political stances accurately. The following experiments were conducted to improve model performance and understand what the model is capable of understanding and where it struggles.

We conducted two types of experiments to test BERT's ability to understand complex political speech. The first type of experiment tested the effect of hyperparameters on BERT's performance. Hyperparameters are model settings that change how the model is trained. We tuned the learning rate of the model and observed changes in performance. We then conducted an experiment to test another loss function called Focal Loss, which has proven more effective than cross-entropy loss on a large number of categories and small amount of data for many categories.

The second type of experiment observes changes in performance when training BERT on different subsets of the dataset. The first experiment was to measure BERT's test accuracy when trained on the entire English corpus. Next, we trained and evaluated BERT on only the manifestos from the United Kingdom. Then, we trained BERT on the top 5, 10, 15, and 20 most abundant categories in the dataset.

3.1. Learning Rate Selection

To achieve best model performance, an important hyperparameter to tune is the learning rate. The learning rate is a hyperparameter that determines how much the weights of the network are updated each training iteration. The weights are updated by adding them to the gradient of the loss multiplied by the learning rate. It is typically a small decimal number and the recommended learning rate for BERT is 0.00005. In this experiment, we evaluated performance when both increasing and decreasing the learning rate. Increasing the learning rate causes a model to learn in less iterations. It can be helpful in some cases but can potentially cause the model to overfit. Overfitting is a phenomenon in machine learning in which the model performs very well on the training data but performs poorly when tested on new data. Decreasing the learning rate can help by allowing the model to learn more gradually and typically generalize to new data more effectively. However, this can potentially harm model performance by underfitting the data.

Figure 3. Training loss after training on X number of batches. Created by Patrick Elisii using WandB.

The differences between learning rates are best illustrated in the training loss curve. The figure above shows the loss function over the course of training. The x axis is the number of batches trained on and the y axis is the loss calculation. We should expect a higher learning rate to converge in less training batches. Convergence refers to the point in model training in which loss stops decreasing or only minimal changes are observed. In the figure above, the 1e-5 learning rate curve converges at a loss of about 1 and is a case of underfitting. The curve for 5e-5 converges at about the same loss as 8e-5 very close to 0, indicating both models were able to learn the training data effectively.

Learning Rate	**Accuracy**	**F1**
8e-5	**57.98%**	**57.77**
5e-5	**57.59%**	**57.31**
1e-5	**49.52%**	**50.55**

Table 5. Performance of BERT with variable learning rate. Created by Patrick Elisii

The table above shows learning rate, accuracy, and f1 for each model trained. The experiments show that the learning rates of 5e-8 and 8e-5 yield very similar results. 8e-5 slightly outperformed 5e-5 in both accuracy and f1 and thus, we can conclude it is the optimal value for learning rate. The rest of the experiments use 8e-5 as the learning rate.

3.2. All English Corpus vs. United Kingdom

Experiment 3 tests if different dialects confuse the language model and decreases performance. American English and British English have many differences such as spelling. We suspected the slight differences in American, Canadian, Australian, and British English could be confusing BERT's internal representations of language. To test this, we ran this experiment isolating the United Kingdom's manifestos because they had the most data of any individual country in the data source. To conduct the experiment, we fine-tuned 2 BERT models. One model was trained using all 51 English speaking manifestos with 58,000 sentences in the dataset. This served as the mixed-dialect model. The second model was trained only using manifestos from the United Kingdom, totaling 29 manifestos and 27,104 sentences.

Model	**Accuracy**	**F1**
All English Manifestos	**57.98%**	**57.77**
United Kingdom Manifestos	**52.39%**	**51.25**

Table 6. Performance of BERT on British vs. Mixed English

In this experiment, we found that isolating the United Kingdom manifestos decreased accuracy and f1-score. We concluded that the differences in language did not impact the model and that more data is more important than a consistent dialect.

3.3. Performance on Abundant Categories

Experiment 4 tested the model's capability to accurately classify political stances, given an abundance of examples. If the model is able to perform well on these abundant categories, then the issue is not that the model is uncapable of learning this type of language, but rather that this dataset does not have sufficient examples. To run the experiment, we isolated the categories with the most examples in the dataset and exclusively trained the model on these subsets. The subsets include the top 5, 10, 15, and 20 categories ranked by support. Support is defined as the number of examples of a category in the dataset.

The dataset of sentences is highly skewed with a small number of categories making up the majority of the dataset. Reference the % of Dataset column in the table below. The top 5 categories make up 34.58% of the entire dataset, totaling 20,128 sentences. Their median examples per category is 3591 compared to the 325 median examples per category when all categories are included. Fine tuning transformers is known for being able to train effectively on small amounts of data, however, only a few hundred examples are likely not enough to understand complex political topics.

Model	% of Dataset	Median Examples per Category	Accuracy	F1
Top 5 Classes	**34.58%**	**3591**	**82.82%**	**82.79**
Top 10 Classes	**51.91%**	**2722**	**74.62%**	**74.55**
Top 15 Classes	**62.91%**	**2132**	**71.14%**	**71.20**
Top 20 Classes	**70.33%**	**1628**	**65.68%**	**65.73**
All Classes	**58,212**	**325**	**58.42%**	**57.89**

Table 7. Performance of BERT on subsets of the dataset. Created by Patrick Elisii

The experiment revealed that number of examples is highly correlated with accuracy and f1 for this dataset. In the table above, accuracy and f1 increase with an increase in median examples per category. This means that the more examples the model has for a specific political stance, the better the model is able to create internal representations and recognize that stance. An accuracy of 82.82% and f1 score of 82.79 are strong and indicates the model is very capable of learning to identify the policy stances in the manifesto given sufficient examples. More data is required to increase overall model performance.

3.4. Focal Loss

Focal loss is a loss function introduced by T Lin et all [15]. It was introduced to address class imbalance within datasets. The Manifesto Project dataset has a significant class imbalance. As shown in experiment 3, 20 classes make up 70% of the data points, leaving 30% of the dataset for the other 62 classes. Using focal loss can improve our model's performance by directly addressing this imbalance. Focal loss accounts for class imbalance by down-weighting easy examples and up-weighting harder examples.

$$FL(p_t) = -\alpha_t(1 - p_t)^{\gamma} - \log(p_t)$$

In the equation above, p_t is the predicted probability, α_t is the assigned weight of the class (described below), and γ is a tunable parameter to adjust how much to down-weight easy examples. Down-weighting easy examples means that focal loss results in a lower loss for correct predictions with high probability than cross-entropy loss. It accomplishes this by multiplying cross entropy loss, $\log(p_t)$, by 1 minus the predicted probability raised to the gamma power, $(1 - p_t)^{\gamma}$. Higher probability predictions will be multiplied by a small decimal,

effectively down weighting them. Then, it is multiplied by α_t, the balance factor. In the paper, the writers mention this is a tunable parameter, but they recommend assigning it to the inverse class frequency which is what we used for the experiment.

Gamma	Accuracy	F1
0.0	**57.95%**	**57.61**
0.5	**57.63%**	**57.27**
1.0	**57.79%**	**57.40**
2.0	**57.70%**	**57.29**
5.0	**57.69%**	**57.31**

Table 8. Performance of BERT trained with Focal Loss. Created by Patrick Elisii

As seen in the table above, training the model with focal loss had a near negligible effect on testing performance. We can see differences in performance, however, with lower gamma values yielding slightly higher accuracy and f1. The model with gamma=0.0 is functionally the same cross entropy with the balanced factor α_t. This model proved to be the best of the focal loss models, though the performance is slightly worse than with cross entropy loss. We can conclude that focal loss does not increase performance for this specific dataset.

3.5. Future Work

This research has shown the potential effectiveness and applications of large language models (LLMs) to summarize political manifestos and promote unbiased reporting of political candidates. In addition, the study also reinforced the idea that there are some real-world domains where the promise of few-shot learning by LLMs is unrealized. In the case of the imbalanced dataset used in this study, many categories had very few examples, and it was expected that LLMs should have been able to handle these categories effectively. Despite experimenting with alternate loss function formulations, the model was not able to achieve a reliable accuracy.

This observation suggests that in such domains, alternative ways of handling data imbalance need to be explored for training more effective and accurate LLMs. One potential direction for future research would be to leverage models such as GPT-4 to paraphrase text to generate additional data to compensate for skewed data distributions. In addition to exploring alternative ways of handling data imbalance, future research could also focus on using more advanced language models and extending the proposed method to other languages. By adapting the method for languages other than English, its applicability and utility could be broadened, benefitting political analysts, journalists, and policymakers worldwide in summarizing complex political documents.

Another possible avenue for expanding this research is creating a website that provides easy-to-read summaries of party policy stances is. By making these summaries readily available, citizens would be better informed about political candidates and their positions without having to rely on potentially biased news networks.

In conclusion, while this study has demonstrated the potential of large language models in summarizing political manifestos and contributing to unbiased reporting of political candidates, there is still work to be done in addressing data imbalance challenges. Exploring innovative approaches, such as GPT-enabled paraphrasing, and continuing to refine and expand the scope of the research will be crucial in realizing the full potential of LLMs in understanding and summarizing complex political documents.

Conclusion

The purpose of this paper was to test the capabilities of state-of-the-art NLP models to understand and summarize information from political manifestos. Rather than obtaining information about candidates from often biased news networks, citizens could learn directly from a manifesto without having to read the entire long and verbose document. From our experiments, we concluded that current state-of-the-art models are capable of understanding the political stances found in manifestos. However, their performance falters on topics with fewer examples in the dataset. Our research indicates that BERT would be able to perform well on all topics given sufficient examples. We also found that methods to increase performance, such as hyperparameter tuning and custom loss functions, only make small changes for language transformers. We learned that the amount of the data is much more important than setting the best hyperparameters for training large language models. In the future, this research can be improved using more advanced language models. Additionally, this research can be used to create a website that provides easy to read summaries of party policy stances. In conclusion, large language models are capable of understanding complex speech and these models need a sufficient amount of data in order to do so.

Bibliography

[1] Lehmann, Pola / Burst, Tobias / Matthieß, Theres / Regel, Sven / Volkens, Andrea / Weßels, Bernhard / Zehnter, Lisa (2022): The Manifesto Data Collection. Manifesto Project (MRG/CMP/MARPOR). Version 2022a. Berlin: Wissenschaftszentrum Berlin für Sozialforschung (WZB). https://doi.org/10.25522/manifesto.mpds.2022a

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, Aidan N, Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. ArXiv.org. https://arxiv.org/abs/1706.03762

[3] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. arXiv.org. Retrieved March 31, 2023, from https://arxiv.org/abs/1810.04805

[4] Chelba, C., Norouzi, M., & Bengio, S. (2017). N-gram Language Modeling using Recurrent Neural Network Estimation. *ArXiv:1703.10724 [Cs]*. https://arxiv.org/abs/1703.10724

[5] Watson, A. (2020, October 23). *News sources bias U.S. 2020*. Statista. https://www.statista.com/statistics/874821/news-media-bias-perceptions/

[6] Knight Foundation. (2020, August 4). *American Views 2020: Trust, Media and Democracy*. Knight Foundation. https://knightfoundation.org/reports/american-views-2020-trust-media-and-democracy/

[7] NW, 1615 L. S., Suite 800Washington, & Inquiries, D. 20036USA202-419-4300 | M.-8.-8. | F.-4.-4. | M. (n.d.). *Republicans, Democrats agree: They can't agree on basic facts*.

Pew Research Center. https://www.pewresearch.org/fact-tank/2018/08/23/republicans-and-democrats-agree-they-cant-agree-on-basic-facts/

[8] Grieco, E. (2020, April 1). *Americans' main sources for political news vary by party and age*. Pew Research Center. https://www.pewresearch.org/fact-tank/2020/04/01/americans-main-sources-for-political-news-vary-by-party-and-age/

[9] Van Green, T. (2020, September 9). *Few Americans are confident in tech companies to prevent misuse of their platforms in the 2020 election*. Pew Research Center. https://www.pewresearch.org/fact-tank/2020/09/09/few-americans-are-confident-in-tech-companies-to-prevent-misuse-of-their-platforms-in-the-2020-election/

[10] *The Cover Up: Big Tech, the Swamp, and Mainstream Media Coordinated to Censor Americans' Free Speech - United States House Committee on Oversight and Accountability*. (2023, February 9). United States House Committee on Oversight and Accountability. https://oversight.house.gov/release/the-cover-up-big-tech-the-swamp-and-mainstream-media-coordinated-to-censor-americans-free-speech-%EF%BF%BC/

[11] Di Cocco, J., & Monechi, B. (2021). How Populist are Parties? Measuring Degrees of Populism in Party Manifestos Using Supervised Machine Learning. *Political Analysis*, 1–17. https://doi.org/10.1017/pan.2021.29

[12] Chatsiou, K. (2020). Text Classification of Manifestos and COVID-19 Press Briefings using BERT and Convolutional Neural Networks. *ArXiv:2010.10267 [Cs]*. https://arxiv.org/abs/2010.10267

[13] Bilbao-Jayo, A., & Almeida, A. (2018). Automatic political discourse analysis with multi-scale convolutional neural networks and contextual data. *International*

Journal of Distributed Sensor Networks, *14*(11), 155014771881182. https://doi.org/10.1177/1550147718811827

[14] *Libraries*. (n.d.). Huggingface.co. Retrieved March 31, 2023, from https://huggingface.co/docs/hub/models-libraries

[15] Pritesh Prakash. An Explanatory G*uide to BERT Tokenizer*. (2021, September 9). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/09/an-explanatory-guide-to-bert-tokenizer/

[16] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2018). Focal Loss for Dense Object Detection. *ArXiv:1708.02002 [Cs]*. https://arxiv.org/abs/1708.02002

Academic Vita

Education: The Pennsylvania State University

Bachelor of Science in Computer Science Anticipated Graduation: May 2023

- ◊ **Schreyer Honors College**
- ◊ **Relevant Coursework:** Data Structures, Algorithms, Systems Programming, Operating Systems, AI, Regression Analysis, Machine Learning, Application Development in SwiftUI

Work Experience

Machine Learning Research Assistant at Penn State College of IST August 2022 - Present

- ◊ Researching the summarization and information extraction of political candidate manifestos
- ◊ Experimenting with transformer machine learning models to summarize and extract information from manifestos

College to Corporate Applications Developer Intern at Vanguard May 2022 - August 2022

- ◊ Researched and experimented with similarity search algorithms then developed an API endpoint in python
- ◊ Conducted data analysis and designed a solution to predict client future value. Led data engineering and machine learning modeling to generate predictions
- ◊ Collaborated with my team to make decisions affecting the solution architecture

Software Engineering Intern at ICR Inc. May 2021–August 2021

- ◊ Developed an end-to-end natural language processing analytic in python into deployment
- ◊ Created solutions implementing optical character recognition (OCR) and named entity recognition (NER)

Data Science Intern at Nittany AI Alliance (Avantor Sciences) June 2020 – August 2020

- ◊ Conducted data analysis and created time series forecasts as a consultant for the Avantor Sciences supply chain division. Worked under Innovation team (including the VP of supply chain) at Avantor
- ◊ Led data analysis and time series forecasting on team of three interns. Delivered a user interface displaying inventory stocking forecasts. Presented our findings and recommendations to the Avantor executives

Skills

- ◊ **Programming Languages**: *Advanced*: Python | Swift; *Intermediate*: Java | C | SQL | HTML | JavaScript
- ◊ **Technical Skills**: Machine Learning | IOS Development | Computer Vision | Natural Language Processing | Time series forecasting | Image classification | Optical Character Recognition | Web Development - React
- ◊ **Libraries:** Pytorch | TensorFlow | Pandas | OpenCV | Huggingface | Scikit-learn | Gym | Faiss | Flask

Leadership and Awards

Nittany AI for Good Challenge Team Lead: AIPR Jan 2021 – September 2021

- ◊ AIPR is a startup automating the sorting of recyclables using a machine learning powered robot and connecting recyclers together with a mobile application (see LinkedIn for articles and demo video)
- ◊ Served as team leader and applications developer, using skills including marketing research, project management, and mobile app development using Swift

President of Nittany AI Student Society May 2022 – Present

- ◊ Leading team in developing workshop content including technical lessons and entrepreneurial workshops
- ◊ Promote and speak on behalf of the Nittany AI Alliance as their student representative

2nd Place National CodePath IOS App Competition Jan 2021 – June 2021

- ◊ Developed mobile app for smarter recycling, placed 2nd in the national IOS competition (see LinkedIn for demo