Abstract

The rise in political polarization and growing distrust in the media have made it increasingly difficult for readers to discern unbiased news. Here, I explored the development of a tool designed to detect political bias in news articles, addressing the need for greater transparency in media consumption. Various models, including traditional machine learning algorithms and advanced NLP models like BERT and LLaMA3, were built and evaluated. Despite initial expectations, the BERT model outperformed the newer LLaMA3 model, likely due to its superior ability to capture nuanced language context. The fine-tuned BERT model proved to be the most effective, and an additional filtering mechanism based on sentiment analysis was implemented to further reduce false negatives. This adjustment provided only marginal improvements, underscoring the already high performance of the fine-tuned BERT model.

Introduction

Since the 2016 presidential election cycle, there has been a significant shift in the political landscape in the United States as political parties have increasingly moved away from the center and towards more extreme positions. As a result, the lines between objective reporting and opinionated commentary have become increasingly blurred, making it difficult for readers to discern unbiased information from partisan perspectives. Approximately two-thirds of U.S. adults surveyed in 2022 express minimal or no trust in the mass media's ability to report the news accurately and fairly. Moreover, over 30% of respondents believe that national news outlets actively seek to mislead the public or sway them towards a particular political viewpoint.[1]

In response to these challenges, there is a growing need for tools that can accurately assess and predict the political bias of news articles. Developing and optimizing a model that can perform this task is a crucial first step towards informing readers about the skew and bias that is present in the media they consume. Transparency in media reporting would allow readers to evaluate the sources, writers, and articles they encounter, leading to a more informed and discerning public. This report details the development of a political bias prediction tool, including the underlying models, optimization techniques, and the potential impact on media consumption and public discourse in politics.

[1] https://www.statista.com/topics/3251/fake-news/

Related Work

In reviewing research on bias detection using Natural Language Processing (NLP), several different approaches have been explored. Some studies go into the various forms of media bias, such as the work by Rodrigo-Ginés (2022)[2], which identifies 17 distinct types. These studies focus on how bias can influence the classification of misinformation, highlighting the complexity of bias in media content. Other research takes a different approach by focusing on the language itself—specifically, how certain words and phrases might lead to biased interpretations. For example, Raza (2022) discusses strategies for spotting biased language and suggests ways to replace it with more neutral terms. The goal here is to enhance the fairness of news articles, making them less prone to the influence of biased language and more balanced overall.[3]

Another approach focused on quantifying political bias in news articles, particularly by analyzing the bias of search engines based on the political slant of the top articles they return for specific queries (Gezici, 2021)[4]. In this study, the researchers crowdsourced a set of controversial search engine queries and developed three key metrics to measure political bias, specifically looking at the distribution of political perspectives within the search engine results pages (SERP). These metrics provided a structured way to evaluate how search engines may contribute to political bias by favoring certain viewpoints over others.

These various research efforts demonstrate that while significant work is being done across a broad spectrum—from the specific phrasing within articles to broader concepts like the overall bias of entire search engines like Google or Bing—there remains a clear gap. There is an opportunity to focus on optimizing the recognition of bias in individual articles as a whole. By doing so, we can provide readers with a tool or filter that helps them navigate political news without needing to dive into overly specific details or resort to drastic measures, such as avoiding entire search engines altogether.

Project Details

Data Gathering and Pre-Processing

The initial challenge was identifying a suitable dataset. I explored various datasets, but none fully met the specific constraints I was aiming for. Ideally, the dataset would include articles from a diverse range of authors, representing different genders, races, and news sources. Additionally, I wanted these sources to be spread evenly across the political bias spectrum, as

[2] https://www.sciencedirect.com/science/article/pii/S0957417423021437
[3] https://arxiv.org/pdf/2208.05777
[4] https://ar5iv.labs.arxiv.org/html/2210.03404

rated by the AllSides[5] website, which categorizes news sources based on their political leanings. Unfortunately, I couldn't find a dataset that met all these criteria. To address this, I began manually scraping news articles to create a dataset that would better represent the diverse perspectives I was looking for. However, due to time constraints, I had to compromise and ultimately decided to use an unlabeled dataset of approximately 2,500 political news articles from CNN, published from 2011-2022.[6] While this dataset had its drawbacks—being solely from CNN, potentially covering political topics beyond U.S. politics, and featuring an uneven distribution across authors—it was a solid starting point given the 8-week time frame for this project and was deemed usable for building the first iteration of the bias detection tool. I then labeled the data by feeding the dataset into the GPT 4 model built by Open AI through their online ChatGPT tool using the following prompt:

"I will give you a csv file with a bunch of cnn articles and you will add two columns to them: 1 with a label for their sentiment (positive or negative) and one for a label of their political lean (liberal or conservative)"

This took a few tries to find the correct prompt. But eventually, GPT4 took my unlabeled dataset, did a sentiment analysis and political bias analysis, and filled two new columns in my dataset with these new labels. Since GPT 4 is the most advanced model available to the public when this paper is written, I accepted these labels as the benchmark for my models - after doing a manual check of 10 articles at random to confirm GPT4's validity. I then pre-processed the data by removing extraneous data features in file and cleaned the articles' content by removing special characters that would increase the difficulty in the eventual tokenization and classification process using the python `re` (regular expressions) library (`pre_process.ipynb`).

Models and Metrics

With the data ready for use, the next step was to experiment with various models. First I used four standard machine learning (ML) models: Logistic Regression (LR) , Random Forest (RF), Support Vector Machine (SVM), and a Neural Network (NN). Under the assumption that none of these models would perform as well at semantic analysis as an NLP model, I did not spend too much time fine tuning the hyperparameters as would have been done in standard cases. Parameters such as C value in the LR, tree stub depth in the RF, and number of hidden layers and max_iterations in the NN were all set to default values. Based on the surprisingly high effectiveness of these models in their base forms (as seen later in the results section), I decided to stick with these hyperparameter values as these models were supposed to serve as benchmarks before moving on to the NLP models. All four of these models were used to classify both sentiment and political bias relative to the GPT4 labels

[5] https://www.allsides.com/media-bias/media-bias-chart
[6] https://www.kaggle.com/datasets/hadasu92/cnn-articles-after-basic-cleaning

(`base_ml_models.ipynb`). Since these models did not require a lot of computing power and my dataset was relatively small (2461), I was able to run these on just a single computer's cpu.

After this, I moved on to NLP models, specifically the BERT model. I researched different models in this family of models (RoBERTa, ALBERT, DistilBERT) but decided to go with the `bert-base-uncased` model. This model was selected because it is optimized for capturing nuanced meanings in text,[7] making it well-suited for identifying biases in political news articles. I was also not limited to working on a cpu and so I was able to refrain from using DistilBERT which while performing comparably, is a smaller model and might drop in performance slightly due to this.

To implement this model, I created a pipeline that first pre-processed the data, as I had done with the base ML models. Next, I used the `BertTokenizer` to prepare the text for the model. The tokenizer converted the text into a format that BERT could work with by mapping each word to an integer ID. It also automatically created attention masks, which were used to distinguish between actual words and padding that I added, in order to ensure all inputs were the same length - max of 512 tokens in this case, but I would like to experiment with different values for this hyperparameter given more time. Then I created a custom `NewsDataset` class. This class took care of the tokenized text and the labels, making them easy to feed into the model during training. I then loaded a pre-trained `BertForSequenceClassification` model, which is specifically designed for classification tasks like this. I fine-tuned this model using part of the dataset (80-20 train-test split), experimenting with a few key aspects such as the number of training epochs (see results) and the batch size. As the model trained, I tracked its performance using the metrics described in the results section and the best version of the model was then evaluated for comparison to the other models (`bert_model.ipynb`).

I wanted to compare the results of this model to a model that was more recently created in the transformer space. So initially I planned to use the LLaMA2 model created by Meta. However during the project, Meta released an even newer model LLaMA3, which I was able to access. Specifically, I chose to use the `Prompt-Guard-86M` model from the LLaMA3 model family. With 86 million parameters, it is not as large as some other models, which made it easier and faster to fine-tune while still being capable of capturing important language patterns. The "Prompt-Guard" name also caught my attention since it suggests that the model is designed to follow specific prompts or guidelines closely—something that's really important in bias detection, where subtle language cues can make a big difference.

The process began with setting up the tokenizer and model. I used the `AutoTokenizer` and `AutoModelForSequenceClassification` from the Hugging Face library to load the model and then tokenize the text data. The tokenization involved converting the text into a format that

[7] https://arxiv.org/abs/1810.04805

the model could process, including handling padding and truncation to maintain a consistent input length across all samples. Similar to the BERT pipeline, I created a custom dataset class that converted the labels for political bias into a numeric format, with "liberal" and "conservative" assigned values of 0 and 1, respectively. Finally, I fine-tuned the model with a training split of the data across different numbers of epochs and batch sizes to find the optimal model, which was then evaluated on the test split of the data. Note that while the split was random, it was seeded allowing each model to train and test on the same randomly selected subsets of the data (`llama_model.ipynb`). Both the BERT and LLaMA3 models could run on a single laptop. But to optimize performance and give me time to fine-tune the epoch hyperparameter, I used an L4 GPU on Google Colab for the training aspect of these pipelines.

This LLaMA3 model ended up performing slightly worse than the BERT model after both were fine tuned. While it is designed for efficiency and specific prompt adherence, I believe that the combination of having less parameters than BERT (86 million vs 110 million) and the fact that prompt-guard might not capture the same level of nuanced context that BERT does is why it performed in this way.

Finally, based on the results (below) of these models, I implemented an additional filtering function on top of the BERT model's predictions to improve its accuracy in the edge cases of false negatives (ex. articles that were labeled as negative sentiment and liberal when they were in fact conservative articles). The filtering function involved compiling a list of terms and phrases that are commonly associated with either a liberal or conservative viewpoint. I then analyzed the frequency of these politically charged terms within each article, relative to the article's length. If an article with a predicted negative sentiment contained a high frequency of terms typically associated with the viewpoint of the political bias, the initial prediction was re-evaluated. Specifically, if the sentiment was negative but the article contained a high frequency of liberal terms, and the initial prediction was liberal, the adjusted prediction was changed to conservative and vice versa. This adjustment aimed to better reflect the inherent bias in political writing, where criticism of one party often serves as implicit support for the other. To optimize this extra layer of filtering, I identified an optimal threshold for how often these politically charged terms appeared (se results section). By applying this adjustment, I aimed to fine-tune the model's accuracy, especially in those tricky cases where sentiment analysis and bias detection didn’t quite align in the traditional sense.

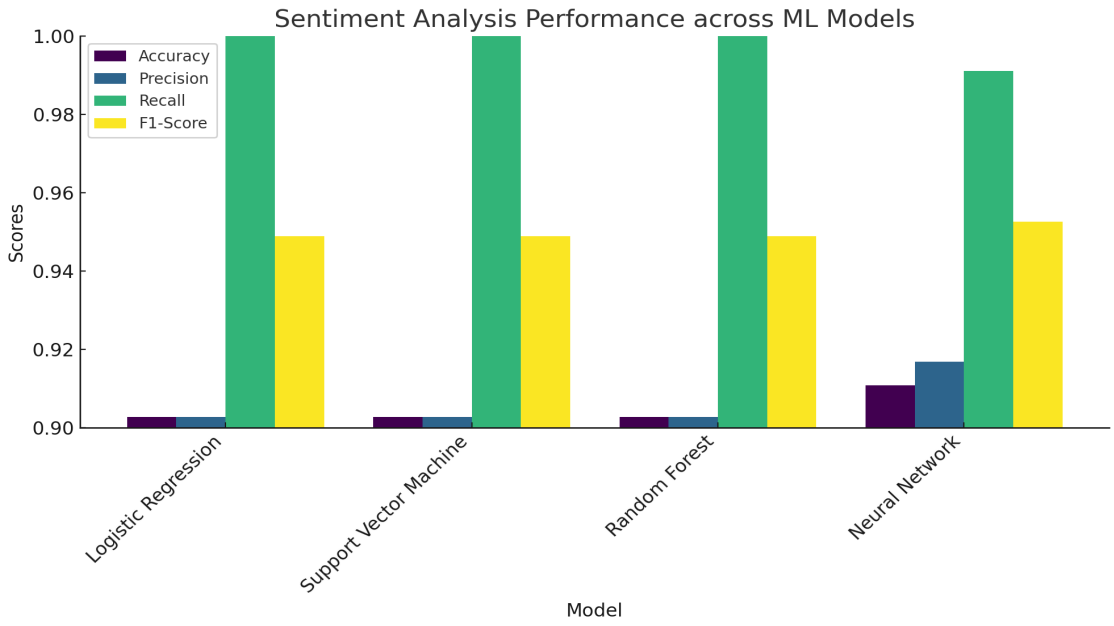
Results

To measure the effectiveness of the models, four metrics were used (accuracy, precision, recall, and F1 score). Accuracy is the standard metric for measuring a linear/binary model’s performance, such as these. However, due to the fact that based on the GPT4 labels, 2213 out of 2461 articles had a positive sentiment and 1690 had liberal bias vs 771 having conservative

bias, I concluded that other metrics should be used when working with base ML models such as these. As a result, the precision and recall scores reflected the model's handling of false positives and false negatives. The F1 score, which balances precision and recall, provided a weighted average that took into account the misclassified articles, considering the proportions of liberal versus conservative content and positive vs. negative sentiment within the dataset. Originally, I was only focused on the accuracy and F1 metrics. However, after running my custom filter on the data and finding that it did not improve my top model, I added the other two metrics as the high recall explained why my filter wasn't needed.

First, I compared the four standard ML models in their sentiment analysis (Fig 1) ability as this was my intermediate goal for the project in case data gathering took too long. The graph below shows that the neural network performed slightly better, correctly classifying approx 20-25 more articles than the other models. However, it should be noted that all the models were significantly more accurate than I would have expected, having a minimum accuracy of 90% in all cases.

Figure 1



For the BERT (fig 2) and LLaMA3 (fig 3) models, I focused only on the ability to predict political bias. In both cases, I first experimented with different numbers of training epochs to find the one that optimized the model without overfitting it to the training data. As you can see in the graphs below, I found that 3 training epochs was best for the BERT model while the LLaMA3 model was optimized at 5 training epochs.

Figure 2

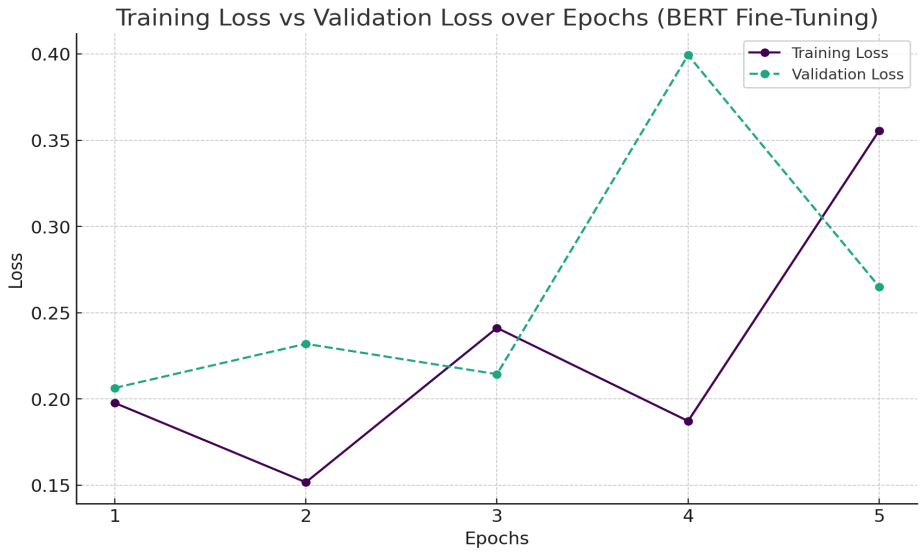


Figure 3

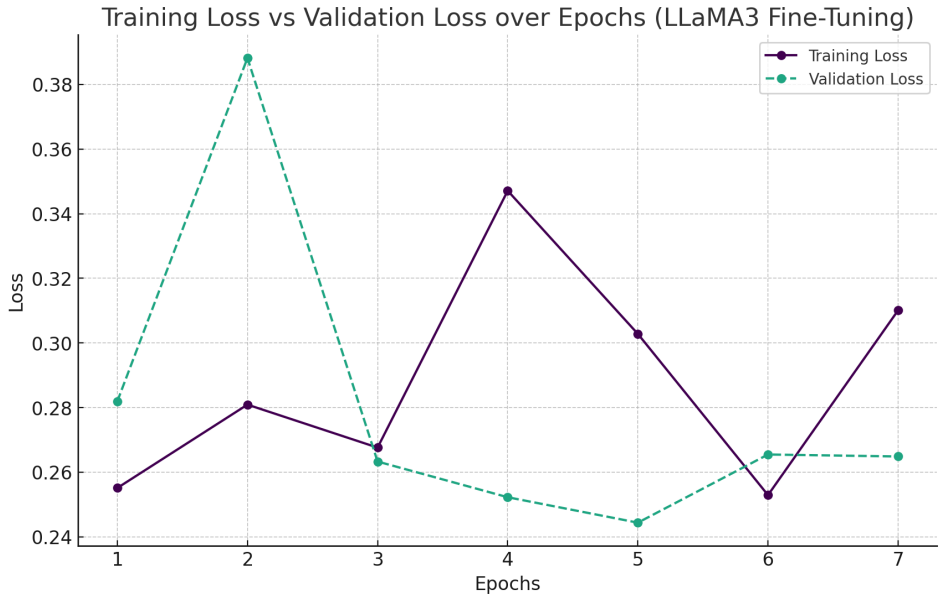
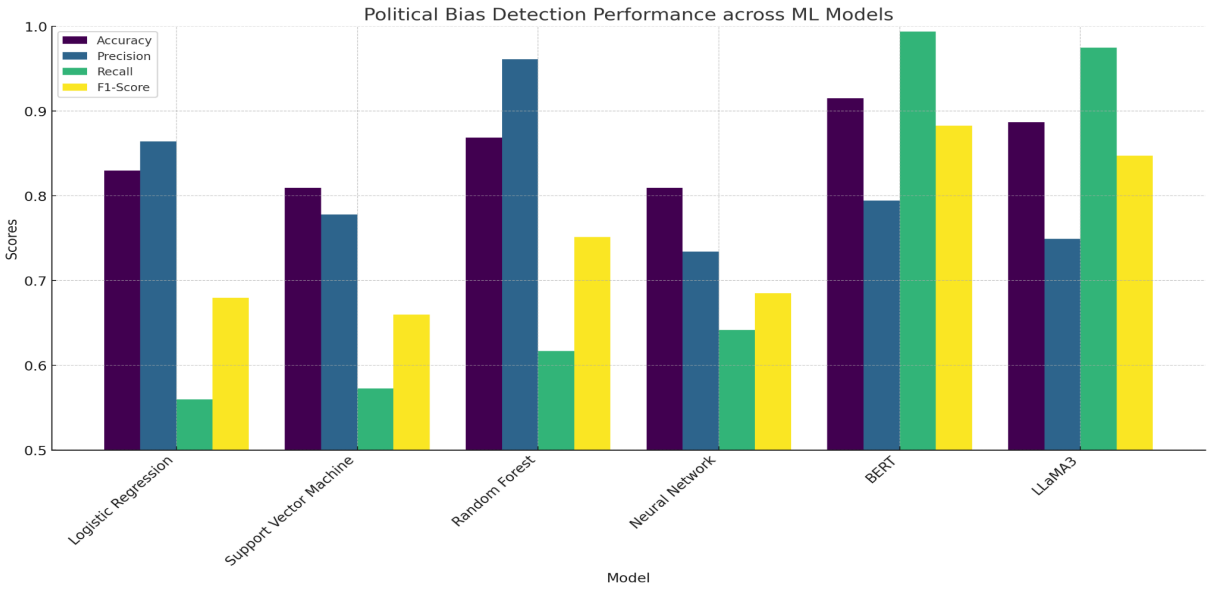


Figure 4 (See appendix for raw data)



After doing this, I evaluated these models on the test split of the data and compared them to the 4 ML models' performance on the same test split of data (fig 4 above). Of the traditional ML models (LR, SVM, RF, NN), the RF model stands out, with the highest accuracy (0.8682) and F1-Score (0.7510) for political bias detection. However, the NLP-based models, BERT and LLaMA3, generally outperform the machine learning models in terms of accuracy (2 - 5%) , recall, and F1(0.09 - 0.13) indicating their clear superior ability to capture the nuances in political bias within the dataset.

The final step that I took was to apply my filter to the output of the best model in the hopes of making it even better. First I had to fine-tune the filter to optimize the threshold (1% of the total word count) of where it would even make a difference to outputs. But from the figures below (fig 5, fig 6), the filter made a minimal improvement in accuracy (91.48% up to 91.68%) and F1 score (0.8823 to 0.8838), which led me to analyze the recall metric. This showed that the recall was already so high (0.9937) that there were almost no false negatives and explained why the filter made a negligible improvement. This also led me to conclude that the few articles that did have a negative sentiment and a political bias X did not mention many terms from that political bias and so my filter wasn't very useful on top of my best NLP model.

Figure 5

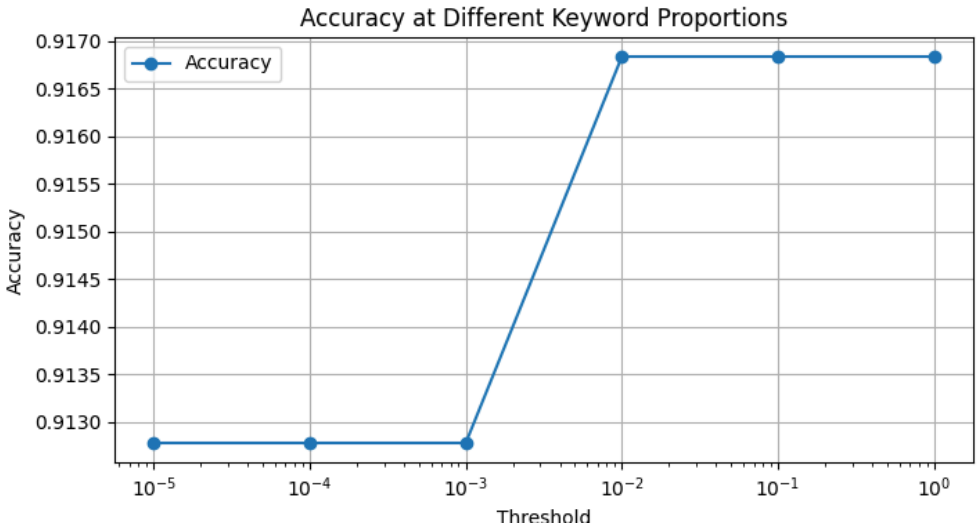
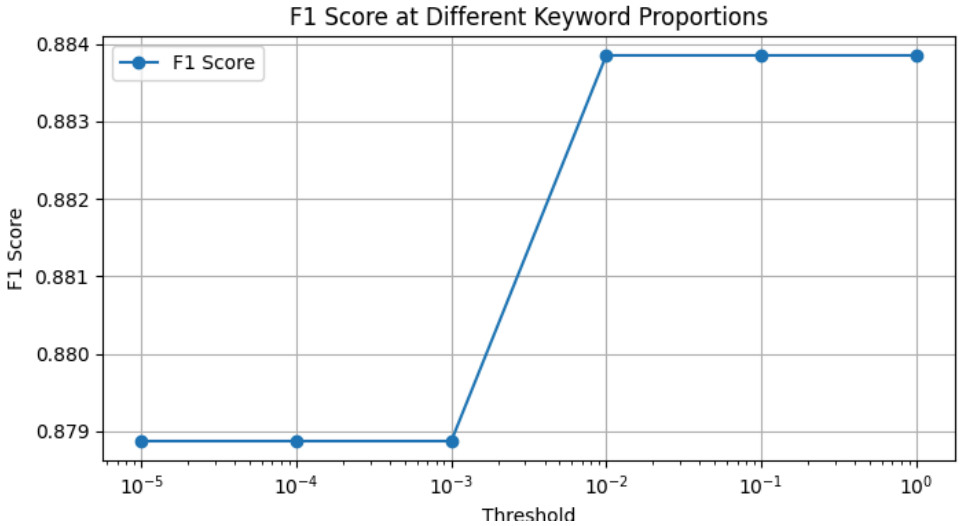


Figure 6



Future Work

While I have made significant progress in developing a tool for political bias detection in this project, there are several areas I want to explore further to enhance the model's effectiveness and broaden its applications. I want to explore attention head analysis within transformer models like BERT and LLaMA3[8]. By examining how different attention heads in these models focus on specific parts of the text, I can gain a better understanding of how they detect bias and find more nuanced patterns. I'm also interested in expanding the scope of bias detection beyond just political bias to recognize other types of bias, such as gender or racial bias. Developing a model that can detect multiple dimensions of bias would make it a more

[8] https://arxiv.org/pdf/2310.16270

comprehensive tool for analyzing media content, providing a deeper understanding of how different forms of bias interplay in news articles. This could also be combined with the attention head analysis so I can use the attention head impacts to recognize which type of bias is most prevalent in an article or other news source. Another approach would be to use data-centric federated learning[9] to build a more robust model. This would involve training multiple models - each on different news sources or different authors - and then taking a weighted average of these models to create a model that accounts for news source bias.

Also, I think there is space to do a more thorough hyperparameter optimization. While I did some tuning in this project, exploring different token counts and learning rates could help fine-tune the model's performance. Training on a more varied and balanced dataset that includes a wider range of news sources, authors, and sentiments could also improve the model's ability to generalize and reduce any unintended biases. Ideally, this would all lead to a system that can detect bias in real-time as new articles or social media posts are published and could be adjusted by the user to be more sensitive to certain types of bias or to prioritize certain sources. This could involve creating a pipeline that continuously ingests and analyzes new content, flagging biased material as it appears.

Personal Analysis

Although I was familiar with the basics of machine learning and had some experience with traditional models, working with transformers required a deeper understanding of how these models process and interpret text. I had to familiarize myself with concepts like attention mechanisms, tokenization, and the challenges of fine-tuning large language models (LLMs). I also needed to better understand the nuances of bias detection. While I was aware of the general concept of bias in media, I had to learn how to translate this into a measurable and detectable form using ML. This involved not only selecting the right models but also carefully considering the metrics and strategies for evaluating their performance.
In combination with the class itself, I spent a lot of time learning how to use libraries such as HuggingFace and PyTorch and a large amount of time in the data gathering and pre-processing stage. While I was already familiar with handling numerical data, the unique challenges from working with text data—such as cleaning, tokenizing, and managing input lengths—required me to spend time learning new approaches - which went hand in hand with what I was learning in the class as well. I also spent a lot of time reading research papers in this area. While there is a solid amount of work being done in bias recognition in NLP, most of it focuses on either coming up with new metrics or finding ways to mitigate the bias as opposed to optimizing the actual process of bias recognition and letting the people use this information as they choose. This really stood out to me and drove my interest in building such a tool. All in all, this project fueled a much deeper interest in the field of bias detection and I really look forward to continuing to do research in this area if possible.

[9] https://blog.openmined.org/federated-learning-types/

Bibliography

1. Statista. (n.d.). *Fake news*. https://www.statista.com/topics/3251/fake-news/
2. Khan, A., Suryotomo, H., & Sujaini, H. (2023). *Sentiment analysis using deep learning techniques*. *Expert Systems with Applications*. https://www.sciencedirect.com/science/article/pii/S0957417423021437
3. He, Y., Yang, Z., Yin, L., & Yu, Y. (2022). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv. https://arxiv.org/pdf/2208.05777
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2022). *Attention is All You Need: A Comprehensive Guide*. arXiv. https://ar5iv.labs.arxiv.org/html/2210.03404
5. AllSides. (n.d.). *Media Bias Chart*. https://www.allsides.com/media-bias/media-bias-chart
6. Hadasu. (n.d.). *CNN articles after basic cleaning*. Kaggle. https://www.kaggle.com/datasets/hadasu92/cnn-articles-after-basic-cleaning
7. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv. https://arxiv.org/abs/1810.04805
8. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., & Jégou, H. (2023). *Transformers in Vision: A Survey*. arXiv. https://arxiv.org/pdf/2310.16270
9. OpenMined. (n.d.). *Federated Learning Types*. OpenMined Blog. https://blog.openmined.org/federated-learning-types/
10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2019). *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP)*. ACL Anthology. https://aclanthology.org/W19-4809.pdf

Appendix

Political Bias Model Comparison Metrics:

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.8296	0.8641	0.5597	0.6794
Support Vector Machine (SVM)	0.8093	0.7778	0.5723	0.6594
Random Forest	0.8682	0.9608	0.6164	0.7510
Neural Network	0.8093	0.7338	0.6415	0.6846
BERT	0.9148	0.7940	0.9937	0.8827
LLaMA3 Prompt-Guard	0.8864	0.7488	0.9748	0.8470