Chris Washer
NLP Individual Project Report

Introduction:

News reporting often is influenced by the viewpoint of the author or media organization which affects the language in the story. Changes in the way the public consume news media and the accessibility of news from non-traditional sources have increased the prevalence of political bias in media. While all media should be expected to have bias, hidden bias in news media, and reporting opinions like facts, is an increasing challenge to the public good of open information.

These changes require individuals to become equipped with ways to understand and identify political bias in news media. Awareness and education are vital to spotting media bias in news reporting. But in this project, our team set out to develop a pipeline to help media consumers detect the bias in news they read.

The goal of our project was to compare several text classification models, from classical methods to state-of-the-art methods, to classify news articles as left, center, or right political bias. After identifying the best model to perform the classification, our team created a pipeline to ingest a URL of a news article and determine the article's media bias. After determining the bias, we leveraged a transformer model to create center-bias abstractive summaries of the article to help resolve the media bias.

Our team leveraged prior work done by Baly et al. (2020), by using the left, center, right labeled dataset of news articles. Using the dataset they generated, we built the following models to detect political bias: Logistic Regression, Naïve Bayes, Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) Recurrent Neural Network, and a fine-tuned RoBERTa Transformer. In addition, we deployed a pretrained Pegasus model to perform abstractive text summarization on news articles.

In this report, I will talk about my contributions to the effort. Namely, I will discuss the MLP, CNN, the RoBerta model, and the Pegasus Summarization (and failed attempt to pretrain).

Experimental Design:

Our team took the approach of generating models with increasing complexity in order to find the best option for this text classification task. To support this objective, I initially created the MLP model and trained on our news bias dataset. I created a custom data loader, word2vec embedding, and a simple 3-layer architecture.

For each model, the dataset was split into a training set, a validation set, and a test set. Following training and validation, the test set was used to determine the models' accuracy and F1 score to evaluate the overall ability of the model. The model which had the highest

accuracy and F1 score were deployed onto our Streamlit application to detect political bias in news articles.

The app was designed to ingest a URL of a news article, determine the political bias of the article, and then provide the bias of the summary.

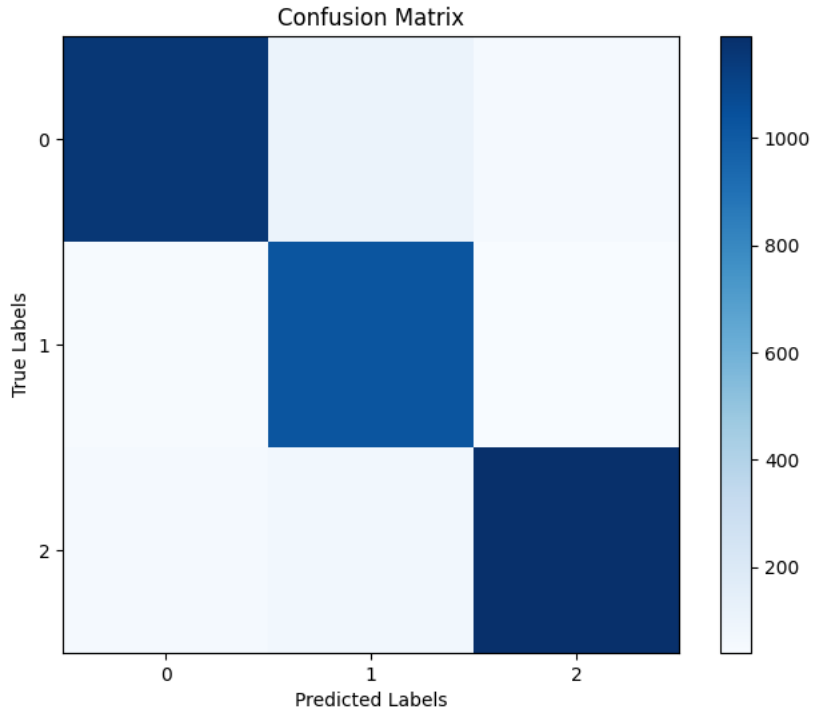
Description of Individual Work

An MLP is a basic feed forward neural network that is useful for basic classification problems. The model built for this project used a pretty basic architecture. The model consisted of an input layer which takes the input tensors with an embedding dimension of 300, based on the word2vec embedding used (Kamath et al). The model also contained two fully connected hidden layers and finally the output layer. This model uses the Rectified Linear Unit (ReLU) activation function, this function enables more complexity. To train the model, I used a cross-entropy loss function and AdamW optimizer. Overall this model achieved accuracy of 58.7% and an F1 score of 0.57.

After, I created the CNN model using a similar code framework as the MLP model. I used the same dataset class and word2vec embedding but added convolutions into the neural network module. The CNN model itself is more complex than the MLP (Kamath et al).. This model starts with a similar input layer. Then the model has multiple convolutional layers, which mathematically learn and extract features from the input depending on the filter size parameter. Some other features included in this model architecture include the pooling, concatenation, and dropout which are applied to the outputs of the convolution layers. This model uses the Rectified Linear Unit (ReLU) activation function in the convolutional layers. To train the model, I used a cross-entropy loss function and AdamW optimizer. Overall, the model performance improved from the MLP with an accuracy of 73% and F1 score of .73.

The third model created was the fine-tuned RoBERTA transformer model. I again used the same dataset class to load the data. However, the preprocessing was done using a RoBERTa Tokenizer with a max length of 512. For this model, I leveraged the pretrained RoBERTa base model which loads the pretrained model weights and configuration. The base model was then fine-tuned using the news bias dataset. To fine tune, the model uses two tensors, inputs and attention masks – which indicate which tokens should be attended to and which should be ignored (Liu et al.). The model is trained with a supervised learning approach to adapt the model to make more accurate predictions for our dataset. The RoBERTa model achieved the best performance with results below in Figure 1.

Figure 1: Confusion matrix from the RoBERTa model.



Test Loss: 0.3893, Test Accuracy: 0.8980
Test Kappa: 0.8469
Test F1 (weighted): 0.8983

Over, for all the models built to address this classification, the performance increased with the complexity of the model. The accuracy and F1 scores of all the models is in Table 1.

Table 1: A summary of all the models is included:

Model	Naïve Bayes	Logistic regression	MLP	CNN	LSTM	RoBERTa
F-1 score	0.50	0.71	0.57	0.73	0.26	.90
Testing Accuracy	0.54%	0.71%	58.47%	73%	36.08%	90%

The last model I deployed was the Pegasus abstractive summarization model. I initially tried to fine tune the generative model, but our data set did not have unbiased summaries that could be used for training. I tried to use an unsupervised training method but after the training the model failed to abstractly summarize the text.

Our goal was still to make news more digestible to the consumer, and having a summary of the news articles was essential to the success of this project. Luckily, we a Pegasus model that was pretrained on a massive, multi-news corpus. While not perfect, a model trained on a massive news corpus will still reduce heavy bias in the language. However, this is an area for future work to improve the fine tuning of this model.

Conclusions:

Overall, our team was successful in evaluating multiple NLP models to determine the best to deploy for our political bias classification. We deployed a fine-tuned transformer model into a Streamlit Application to assess the political bias of a news article in real time. We then summarized the news article to make the news more consumable for the reader.

For this effort, I created three separate models and incorporated the pretrained Pegasus model into a python script for use in the final app.

Citations:

Baly, R., Da San Martino, G., Glass, J., Nakov, P. (2020). We can detect your bias: Predicting political ideology of news articles. *Archive*. *https://arxiv.org/pdf/2010.05338*

Kamath, Cannannore Nidhi, Syed Saqib Bukhari, and Andreas Dengel. "Comparative study between traditional machine learning and deep learning approaches for text classification." *Proceedings of the ACM Symposium on Document Engineering 2018*. 2018.

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020, November). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In International conference on machine learning (pp. 11328-11339). PMLR.

Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).