Political Bias Detection in News Articles Using Deep Learning

Anderson Hsiao
phsiao@umass.edu

Arnav Nidumolu
anidumolu@umass.edu

Atharva Kale
aukale@umass.edu

Shriram Giridhara
sgiridhara@umass.edu

1 Introduction

Media bias is present in many online articles that we read, but it can be difficult for the average person to notice its presence. This can lead to readers being misled by biased information and is one of the causes of political extremism. By using a machine learning model to detect political bias in online media, readers can be automatically and swiftly informed of what biases could be present in a given article, even if no experts have read or rated it yet.

The input for our model is text from a news article in the form of paragraphs, and we use neural networks to output a number between 0 and 1, with a 0 meaning that the article is reliable (not biased) and a 1 meaning that the article is unreliable (biased).

2 Related Work

Spinde et al used a Transformer-based multitask learning model to detect bias by word choice from a smaller dataset. Their method led to a 3% performance boost compared to existing methods, while being more cost-efficient when it comes to using cloud computing resources (Spinde et al., 2021).

Aires et al introduced a method for using graphs to group websites together based on political biases. Starting off with 20 labeled "seed" websites, they scraped through the hyperlinks in their articles to generate a directed and weighted graph. Grouping by edge weights, they found two or three big clusters of mostly left-leaning or mostly right-leaning websites, with the remaining groups being small ones that were either very extremist, very centrist, or unknown (Patricia Aires et al., 2019).

3 Dataset and Features

The dataset used in this project is pulled from Harvard Dataverse, an online data repository where researchers can publish and access research data. This dataset, titled NELA-GT-2019, was specifically released by its publishers to help contribute to the study of misinformation in news articles (Horne, 2020). It contains nearly 1.12 million labeled articles, published between January 1, 2019 and December 31, 2019, from over 260 news sources. The original dataset contained several indicators of whether an article source is biased or not; however, the label of interest for this project was the aggregated label, which categorized each news source into one of three categories: 0 (reliable), 1 (mixed), or 2 (unreliable). Every article was labeled with the aggregated label corresponding to its source. For example, the news network ABC News has an aggregated label of 0, meaning that all the articles in the dataset published under ABC News are assigned a label of 0 and are considered reliable.

Due to constraints on computing resources, we utilized only 2,000 articles from the dataset to feed into the model. An 80/5/15 percentage-based split was used for the training, validation, and testing sets for our model respectively.

4 Methods

4.1 BERT Preprocessing

BERT (Bidirectional Encoder Representations from Transformers) is an open-source pre-trained model that understands the nuances of the English language. It can comprehend the context by processing any given word in relation to all other surrounding words rather than iterating through them one at a time. So as to better tailor it to the purpose of this project, we fine-tuned BERT on our data to find semantics within news-related context and embed the sentences in each article into vectors that can be passed into a neural network.

4.2 Model Architectures

We fed the vectors returned by BERT into a linear model first, followed by a convolutional neural network and BiLSTM to compare their performances.

4.2.1 Feed-Forward Layer

The basic progression for our linear model was that the BERT embedding layer would initially preprocess the data and embed the sentences into vectors to pass them in for dropout. This would randomly set outgoing weights to 0 with a 50% probability, which we defined, and help prevent overfitting. The resulting output of 768 dimensions is passed into a fully connected linear layer which outputs the weights for the 2 classes representing the model's prediction.

4.2.2 Long Short-Term Memory Layer

After extracting embeddings from BERT, they are passed into the LSTM layer. As fine-tuning the pretrained BERT is difficult due to its sheer size, an LSTM is better equipped to be fine-tuned with the article paragraphs. Just like BERT, LSTM can process sequences of data and maintain both encoders and decoders. However, it lacks parallel computation, making it slower than BERT; for our dataset, LSTM allows for more useful embeddings, being more easily fine-tuned across the training process. We utilized a Bidirectional LSTM with a hidden size of 256. The last cell of the LSTM is then leveraged by the succeeding layers to extract or filter the bias label.

4.2.3 Convolutional Layer

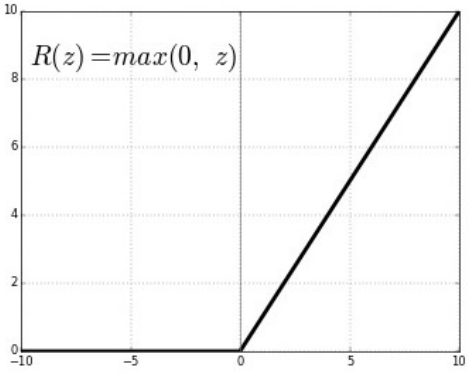
The CNN model's progression is similar to that of the linear model to an extent. However this time, BERT's output vectors are passed into a set of convolutional layers, which use filters to identify and extract specific attributes that contribute to an article being labeled as biased. In this case, we used 64 filters of three kernel sizes: 10, 50, and 100. We also included the ReLU activation function in between, which transforms the convoluted output by activating only certain neurons at a time. The resulting output is max pooled, reducing the number of parameters and size of the batch, and then dropout is applied. Finally, the output is passed into a fully connected linear layer that outputs the model's predicted label.

4.3 Metrics

4.3.1 Loss Function: Cross Entropy Loss

$$Loss = -\sum_{i=1}^{\substack{output \\ size}} y_i \cdot log\ y_i$$

4.3.2 Activation Function: ReLU



4.3.3 Optimizer: AdamW

$$x_t \leftarrow x_{t-1} - \alpha \frac{\beta_1 m_{t-1} + (1-\beta_1)\left(\nabla f_t + w x_{t-1}\right)}{\sqrt{v_t} + \epsilon}$$

5 Experiments/Results/Discussion

5.1 Experiments and Results

5.1.1 Hyper-Parameters

Learning Rate: 2e-5 (Mosbach et al., 2021)
Batch Size: 8 Articles/Batch (Sun et al., 2020)
Number of Filters for CNN: 64

5.1.2 Performance Metrics

Model Type	Accuracy	F1 Score
BERT + Feed Forward	85.5%	86.86%
BERT + CNN	86.2%	87.27%
BERT + BiLSTM + CNN	87.9%	88.15%
BERT + BiLSTM	87.9%	88.24%

Using 297 samples and testing the BERT + BiLSTM model, we observed the following Confusion Matrix:

	Positive	Negative	Total
Positive	126	12	138
Negative	24	135	159
Total	150	147	297

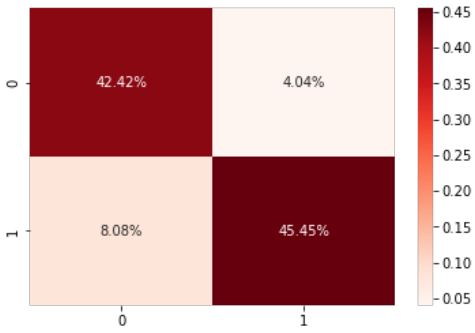


Figure 1: Confusion Matrix for BiLSTM

Area Under ROC Curve: 88.11

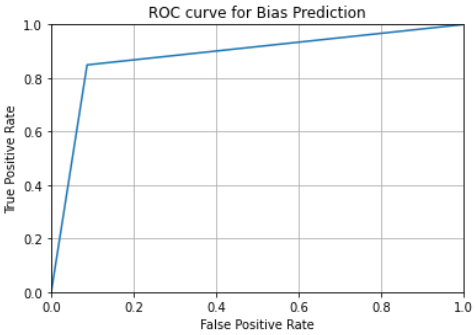


Figure 2: ROC Curve for BiLSTM

5.2 Discussion

5.2.1 Learning Process

As we began our search for an effective dataset for bias detection, we encountered several sources with differing data columns and bias measure. We decided to choose a political-based dataset by Harvard University, with over 2 million samples, which contained links to articles and a clear label for each data point (Biased, Mixed, Unbiased). Through this search for datasets, we learnt to critically differentiate between numerous datasets aimed towards a specific use-case, analyzing data discrepancy and the presence of noise.

In addition, we grasped a way to carefully select a proper neural network pipeline; experimenting between Transformers, CNN's, LSTM's, and Feed Forward Networks, we were able to appropriately pick the best architecture for our dataset. Further, we understood the differences between Loss Functions, deciding to use Cross Entropy for best accuracy. Utilizing other research papers, we were also able to tune hyperparameters such as the learning rate, batch size, and number of CNN filters. In addition, we became aware of the role of optimizers in ensuring a proper training of Neural Networks, leveraging AdamW with default constants. Overall, we gained exposure to the intricacies of Pytorch and the process of building custom models, along with researching various datasets and hyperparameters.

5.2.2 Challenges Encountered

During the development of this project, we found it challenging to build a model that runs under the constraint of limited computational resources. With the first network pipeline defined for the project, we tried running it on the Google Colab Hosted Runtime with GPU acceleration and found the model exceeding 12GB of the RAM provided. The BERT transformer was found to be the most consuming component. Considering these limits, the team needed to reduce the batch size to a lower number (finally reaching a value of 8).

Another challenging task was to choose the right nets for text classification. This goal, being inherently difficult, was even more complicated than the objective of political bias classification, an ambiguous term. Consequently, the team found it difficult to achieve an accuracy metric of over 90

On the flip side, although we are not domain experts in determining bias, we found it impressive that our models were able to achieve an accuracy of 88% on the test set along with 85% on the validation set.

5.2.3 Differences in Layers

While each algorithm has its own advantages, only some of them proved suitable for our project's purpose. BERT worked best for preprocessing as it handles the vector embeddings of each sentence using self-attention and positional embedding. This allows the model to train on the data in relation to surrounding context as opposed to individually going through each sentence without parallel computations, which is what an LSTM-based model would be doing.

In terms of the network itself, a CNN suits our purpose well as it filters the embedding vectors using kernels, identifying "biased" features and gaining insights into the data. A linear feed-forward network would not be as accurate due to its having to compute functions for determining the bias label given BERT's embeddings. Similarly, a recurrent neural network is not suited for paragraph classification as it can only consider one sentence at a time, which means it cannot "re-

member" past computations, effectively ignoring the context of the text.

5.2.4 Future Work

Through this project, we were able to define a pipeline that could be used for any text classification problem by switching out the dataset underlying it. We would like to explore the model's capabilities with detecting other types of implicit information in text, e.g. hate speech, toxicity, misinformation, emotions, and more. We were also certainly limited by computation bandwidth available to us. If we have access to higher computational bandwidth, we want to extract a larger amount of articles from the given dataset and train the model on a bigger batch size (i.e. 16 or 32).

6 Conclusion

In this project, we used neural networks to detect political biases from news articles. We tried four different neural networks, and found that BERT + BiLSTM performed the best at $\sim 88\%$ accuracy. Our takeaway is that BERT is an effective method of restructuring sentences into data that we can pass into a neural network. We also learned that CNNs are not the best choice for our specific application, providing only negligible changes in accuracy.

GitHub Repo:
https://github.com/atharvakale343/BiasAware

References

Horne, B. (2020). NELA-GT-2019.

Mosbach, M., Andriushchenko, M., and Klakow, D. (2021). On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines.

Patricia Aires, V., G. Nakamura, F., and F. Nakamura, E. (2019). A link-based approach to detect media bias in news websites. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 742–745, New York, NY, USA. Association for Computing Machinery.

Spinde, T., Krieger, J., Ruas, T., Mitrović, J., Götz-Hahn, F., Aizawa, A., and Gipp, B. (2021). Exploiting transformer-based multitask learning for the detection of media bias in news articles. *Proceedings of the iConference*, 2022.

Sun, C., Qiu, X., Xu, Y., and Huang, X. (2020). How to fine-tune bert for text classification?